



Fondamenti di Informatica

AA 2018/2019

Eng. Ph.D. Michela Paolucci

DISIT Lab <http://www.disit.dinfo.unifi.it>

Department of Information Engineering, DINFO

University of Florence

Via S. Marta 3, 50139, Firenze, Italy

tel: +39-055-2758515, fax: +39-055-2758570

michela.paolucci@unifi.it



UNIVERSITÀ
DEGLI STUDI
FIRENZE

DINFO
DIPARTIMENTO DI
INGEGNERIA
DELL'INFORMAZIONE

DISIT
DISTRIBUTED SYSTEMS
AND INTERNET
TECHNOLOGIES LAB



UNIVERSITÀ
DEGLI STUDI
FIRENZE

DINFO
DIPARTIMENTO DI
INGEGNERIA
DELL'INFORMAZIONE

DISIT
DISTRIBUTED SYSTEMS
AND INTERNET
TECHNOLOGIES LAB

DISIT Lab



- Researchers: 20
- Main research topics:
 - **Big Data, Smart Cities and Distributed Systems**
 - **Smart City models and applications**
 - **Collective Awareness Platforms**
 - **Creative industry, creativity, elearning**
 - **Smart cloud**
 - **Smart manufacturing: Industry 4.0, Factory 4.0, e-factory**
 - **Smart Retail:** user behavior analysis, engagement, ..
 - **Autonomous drivers:** operators, high speed trains, driverless



Distributed Systems and Internet Technologies Lab
Distributed Data Intelligence and Technologies Lab
Department of Information Engineering (DINFO)
University of Florence



UNIVERSITÀ
DEGLI STUDI
FIRENZE
DINFO
DIPARTIMENTO DI
INGEGNERIA
DELL'INFORMAZIONE

<http://www.disit.dinfo.unifi.it>

qualsiasi tipo deep search

HOME ABOUT RESEARCH INNOVATION CORSI E TESI COME FARE EVENTI MIO PROFILO

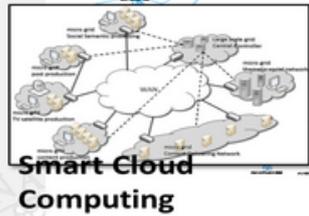
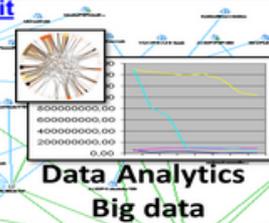
root Uscire

Mostra Modifica Log Translate Devel

DISIT LAB OVERVIEW

<http://www.disit.dinfo.unifi.it>

Text and Web Mining



DISIT lab and research group is active since 1994. It is one of the most active ICT labs of the University of Florence, metropolitan Tuscany area. DISIT successfully developed a relevant number of International and National research, development and innovation projects. DISIT provides an infrastructure for cloud and distributed computing.

Fondamenti di Informatica, Univ. Firenze, Michela Paolucci, 2017/2018

CONTENUTI

- [Ultime Attività](#)
- [In primo piano](#)
- [Più visti](#)
- [Most Viewed \(last 500\)](#)
- [Most Viewed All \(last 500\)](#)
- [Ultimi caricati](#)
- [Più votati](#)
- [Mie collezioni pubblicate](#)
- [Miei contenuti](#)
- [Carica un nuovo contenuto](#)

ROOT

- ▶ Gruppi
- Cerca Utenti
- Contenuti ed attività non lette relative ai tuoi gruppi
- Crea la matrice di tassonomia
- Forum
- Invite a colleague
- Issues
- Keyword cloud
- Messaggi e Sottoscrizioni
- Mio MatchMaking

Con chi lavoriamo



Società Italiana degli Autori ed Editori



CITTÀ METROPOLITANA
DI FIRENZE



Consiglio Nazionale
delle Ricerche



consorzio nazionale
interuniversitario
per le telecomunicazioni



Disit Lab Roadmap on Smart Cities

2013

Km4City 1.1

- Tuscany Map
- Services
- AVM
- Sensors
- Parking
- Cultural Heritage
- Enrichment cities
- Event in the city
- Digital Locations
- Fresh places

- <http://servicemap.km4city.org>
- <http://log.disit.org>
- <http://www.disit.org/fodd>
- <http://www.disit.org/tv>
- <http://smartds.km4city.org>

- Weather
- Cultural Heritage
- Energy recharge pillar
- Wi-Fi
- Events in the city

2015

Km4City 1.4

- Embed
- More API
- iBeacon

API

Twitter Vigilance

2016

REPLICATE H2020
2016-2021- Started



2015



Km4City 1.5

- SmartDS
- Km4City App

RESOLUTE H2020
2015-2018 - Started



Sii-Mobility SCN
2016-2018 - Started
Km4City 1.6.2

2016

REPLICATE H2020
2016-2021- Started



12/2017

WEEE
2017-2020

6/2017

- waste
- Territorial areas and paths
- Health, Bike sharing
- Statistics, Energy, ICT, ...
- E-vehicles

- Risk analysis
- Environmental, water
- Data Licensing models
- Energy Meters
- Fi-Ware compliant

- More Sensors, IOE, IOT
- Dashboard Builder
- Territorial areas and paths
- User Engagement
- Mobility and transport
- Resilience Decision Support

GHOST SIR
2016-2019 - Started



- Suggestions on demand
- User Behaviour Analysis
- Trajectories and OD



Snap4City

- IOT/IOE
- Monitoring
- Smart City IOT integration
- Living Lab

5G

- Smart City vs IOT, Industria 4.0

Traffair EC project as CEF
11/2018

Already Planned up to 2021

Smart City

- **Snap4city:** EC project selected by **SELECT4Cities** pre-commercial procurement for the R&D of large-scale IoE/IoT platforms for urban Innovation
- **Traffair:** EC project as CEF
- **MOSAIC**, regional project with ALSTOM, Tages, LiberoLogico
- **Weee Life:** project of the EC with regione Toscana
- **Feedback project:** co-funded by the Tuscany Region
- **JOIN project:** co-Founded by the Tuscany Region
- **REPLICATE** H2020, SCC1, EC flagship
 - <http://replicate-project.eu/>
- **5G – Prato Wind Open Fiber**
- **RESOLUTE** H2020, EC
 - <http://www.resolute-eu.org>
- **Sii-Mobility** SCN MIUR:
 - <http://www.sii-mobility.org>
- **Km4City:** <http://www.km4city.org>
- **Coll@bora** Social Innovation, MIUR:
 - <http://www.disit.org/5479>
- **Trace-it**
- **Raiss** **projects:**
<http://www.disit.org/5501>



Trace-IT
RAISSS



Snap4City

*Snap4City has been proposed in response to **Select4Cities** challenges and PCP – Pre-Commercial Procurement Project (<http://www.select4cities.eu/>).*

UNIVERSITÀ DEGLI STUDI FIRENZE | DINFO | DISIT
Team University of Florence - Department of Information Engineering (lead contractor) - Italy
www.unifi.it
EFFECTIVE KNOWLEDGE SRL - Italy | UNIVERSITÀ DEGLI STUDI DI MILANO - Italy

SELECT
for Cities

Home Challenge Winners Outputs Blog Contact

Search Site

Welcome to SELECT for Cities Knowledge Hub

Benefit from experience and lessons learned from the ongoing SELECT for Cities Pre-Commercial Procurement Project to develop a data-driven, Internet-of-Everything (IoE) platform for large-scale urban app co-creation



About Internet-of-Everything



About SELECT's City Platforms

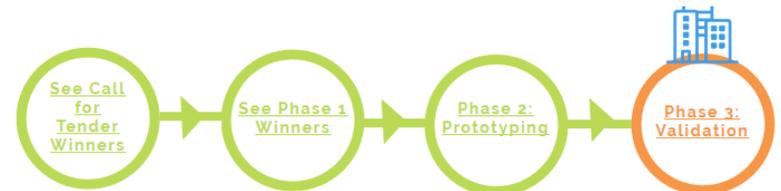


About Smart Procurement

Smart Cities involved:

- Antwerp
- Copenhagen
- Helsinki

Winners of each Phase in our competition to create large scale city innovation labs



City marks the current phase.

Snap4City



- LOCAL GOVERN
- STAKEHOLDERS
- CITY USERS
- IN-HOUSE
- ENERGY OPERATORS
- MOBILITY OPERATORS
- COMMERCIAL OPERATORS
- SECURITY OPERATORS
- INDUSTRIES
- RESEARCHERS
- START-UPS
- ASSOCIATIONS



- GDPR
- SECURITY
- PRIVACY
- ASSESSMENT
- AUDITING

- OPEN HW/SW
- IOT EDGE
- IOT BUTTONS
- IOT DEVICES

IOT APPLICATIONS • INSTANT APPS



PERSONAL DATA • DATA DRIVEN • REAL TIME - ANY PROTOCOL & FORMAT

DASHBOARDS & APPLICATIONS



CONTROL ROOM • KPI • BUSINESS INTELLIGENCE • DECISION SUPPORT • WHAT-IF ANALYSIS

MOBILE & WEB APPLICATIONS



DEV KIT • SUGGESTIONS • USER ENGAGEMENT • COWORKING • SMART APPLICATIONS

MICROSERVICES & ADVANCED SMART CITY API

LIVING LAB - DEV TOOLS - COWORKING



IOT DIRECTORY • SERVICE MAP • RESOURCE MANAGER • DATA GATE • R STUDIO • ETL

BIG DATA ANALYTICS



SMART SOLUTIONS • PREDICTIONS • ANOMALY DETECTION • TRAFFIC RECONSTRUCTION • ORIGIN-DESTINATION MATRIX • SOCIAL MEDIA ANALYSIS • OFFER & DEMAND ANALYSIS • RISK & RESILIENCE ANALYSIS

SMART MICRO-APPLICATIONS - OPEN TO EXTERNAL SERVICES



SMART PARKING • ROUTING • PERSONAL ASSISTANT • INFOMOBILITY • ALERTING • INFO TRIAGE

KM4CITY DATA AGGREGATOR, KNOWLEDGE BASE OF THE CITY

OPEN DATA

GIS+MAP DATA

PROPRIETARY DATA

PERSONAL DATA

IOT / IOE

INDUSTRY 4.0

SOCIAL MEDIA





Horizon 2020
European Union Funding
for Research & Innovation

**REnaissance of PLaces
with Innovative Citizenship
And Technology**

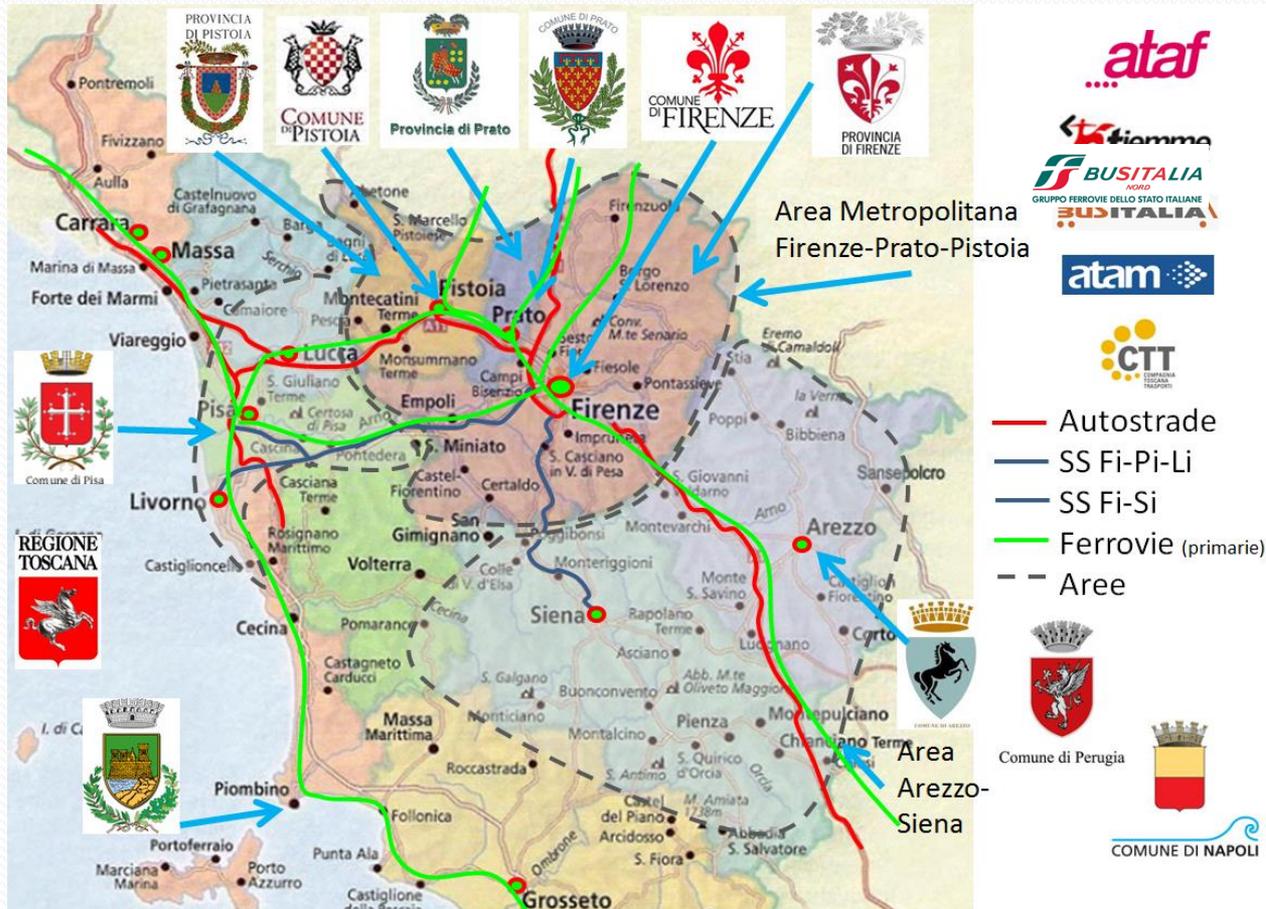


- *demonstrate Smart City technologies in energy, transport and ICT in districts in:*
 - *San Sebastian, Florence and Bristol,*
 - *follower cities of Essen, Nilufer and Lausanne*
- *Cities are the customer: considering local specificities*
- *Solutions must be replicable, interoperable and scalable.*
 - *Integrated Infrastructure: deployment of ICT architecture, from internet of things to applications*
 - *Low energy districts*
 - *Urban mobility: sustainable and smart urban services*

- 1 (coordinator) FOMENTO DE SAN SEBASTIAN FSS SPAIN
- 2 AYUNTAMIENTO DE SAN SEBASTIAN SAN SEBASTIAN SPAIN
- 3 COMUNE DI FLORENCE FLORENCE ITALY
- 4 BRISTOL COUNCIL BRISTOL UNITED KINGDOM
- 5 STADT ESSEN ESSEN GERMANY
- 6 NILUFER BELEDIYESI NILUFER TURKEY
- 7 VILLE DE LAUSANNE LAUSANNE SWITZERLAND
- 8 IKUSI ANGEL IGLESIAS, S.A. IKUSI SPAIN
- 9 ENDESA ENERGÍA, S.A. ENDESA SPAIN
- 10 EUROHELP CONSULTING, S.L. EUROHELP SPAIN
- 11 ILUMINACION INTELIGENTE LUIX, S.L. LUIX SPAIN
- 12 FUNDACION TECNALIA RESEARCH & INNOVATION TECNALIA SPAIN
- 13 EUSKALTEL, S.A. EUSKALTEL SPAIN
- 14 COMPAÑIA DEL TRANVÍA DE SAN SEBASTIÁN DBUS SPAIN
- 15 CONSIGLIO NAZIONALE DELLE RICERCHE CNR ITALY
- 16 ENEL DISTRIBUZIONE, SPA ENEL ITALY
- 17 MATHEMA, SRL MATHEMA ITALY
- 18 SPES CONSULTING SPES ITALY
- 19 TELECOM ITALIA, SPA TELECOM ITALY
- 20 UNIVERSITA DEGLI STUDI DI FLORENCE UNIFI ITALY: DINFO.DISIT, DIEF
- 21 THALES ITALIA, SPA THALES ITALY
- 22 ZABALA INNOVATION CONSULTING ZABALA SPAIN
- 23 TECHNOMAR TECHNOMAR GERMANY
- 24 UNIVERSITY OF BRISTOL UOB UNITED KINGDOM
- 25 UNIVERSITY OF OXFORD UOXF UNITED KINGDOM
- 26 BRISTOL IS OPEN, LTD BIO UNITED KINGDOM
- 27 ZEETTA NETWORKS ZEETTA UNITED KINGDOM
- 28 KNOWLE WEST MEDIA CENTRE, LGB KWMC UNITED KINGDOM
- 29 TOSHIBA RESEARCH EUROPE, LTD TREL UNITED KINGDOM
- 30 ROUTE MONKEY, LTD ROUTE MONKEY UNITED KINGDOM
- 31 ESOTERIX SYSTEMS, LTD ESOTERIX UNITED KINGDOM
- 32 NEC LABORATORIES EUROPE, LTD NEC UNITED KINGDOM
- 33 COMMONWHEELS CAR CLUB CIC CO-WHEELS UNITED KINGDOM
- 34 UNIVERSITY OF THE WEST OF ENGLAND UWE UNITED KINGDOM
- 35 ESADE BUSINESS SCHOOL ESADE SPAIN
- 36 SISTELEC SOLUCIONES DE TELECOMUNICACION, S.L. SISTELEC SPAIN

Sii-Mobility

- Experimentations and validation in Tuscany <http://www.Sii-Mobility.org>
- Integration with present central station and subsystems
- DISIT lab, Università di Firenze, is the coordinator



ataf

tiemme

BUSITALIA
GRUPPO FERROVIE DELLO STATO ITALIANE

atam

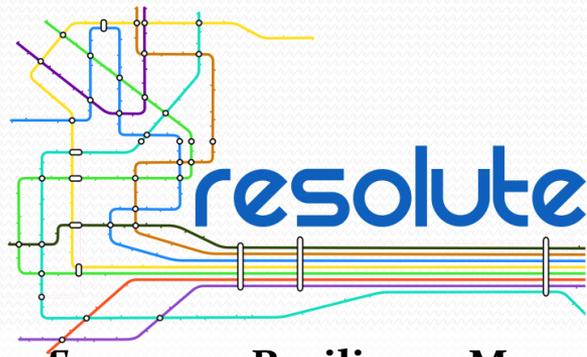
CTT
SISTEMA REGIONALE

— Autostrade
— SS Fi-Pi-Li
— SS Fi-Si
— Ferrovie (primarie)
- - - Aree

Comune di Perugia

COMUNE DI NAPOLI

ECM; Swarco Mizar;
Inventi In20; Geoin;
QuestIT; Softec;
T.I.M.E.; LiberoLogico;
MIDRA (autostrade,
motorola); ATAF;
Tiemme; CTT Nord;
BUSITALIA; A.T.A.M.;
Effective Knowledge;
eWings; Argos
Engineering; Elfi;
Calamai & Agresti;
Project; Negentis



<http://www.resolute-eu.org>

- **Develop European Resilience Management Guidelines (ERMG)**
 - Develop a conceptual framework for creating/maintaining Urban Transport Systems
- Enhance resilience through improved support of human decision making processes, particularly by training professionals and civil users on the ERMG and the RESOLUTE system
- **Operationalize and validate the ERMG by implementing the RESOLUTE Collaborative Resilience Assessment and Management Support Systems (CRAMSS) for Urban Transport Systems** addressing Road and Urban Rail Infrastructures
 - **Pilots in Florence and Athens**
- Adoption of the ERMG at EU and Associated Countries level

University of Florence: DISIT lab DINFO (Proj coordinator), DISIA and DST	UNIFI	IT
THALES	THALES	IT
ATTIKOMetro	ATTIKO	GR
Comune di Firenze	CDF	IT
Centre for Research and Technology Hellas	CERTH	GR
Fraunhofer-Gesellschaft zur Förderung der angewandten Forschung e.V.	FHG	DE
HUMANIST	HUMANI ST	FR
SWARCO Mizar	SWMIZ	IT
Associação para o Desenvolvimento da Investigação no Instituto Superior de Gestão	ADI-ISG	PT
<i>Consorzio Milano Ricerche</i>	CMR	IT

What is enabling and providing smart services

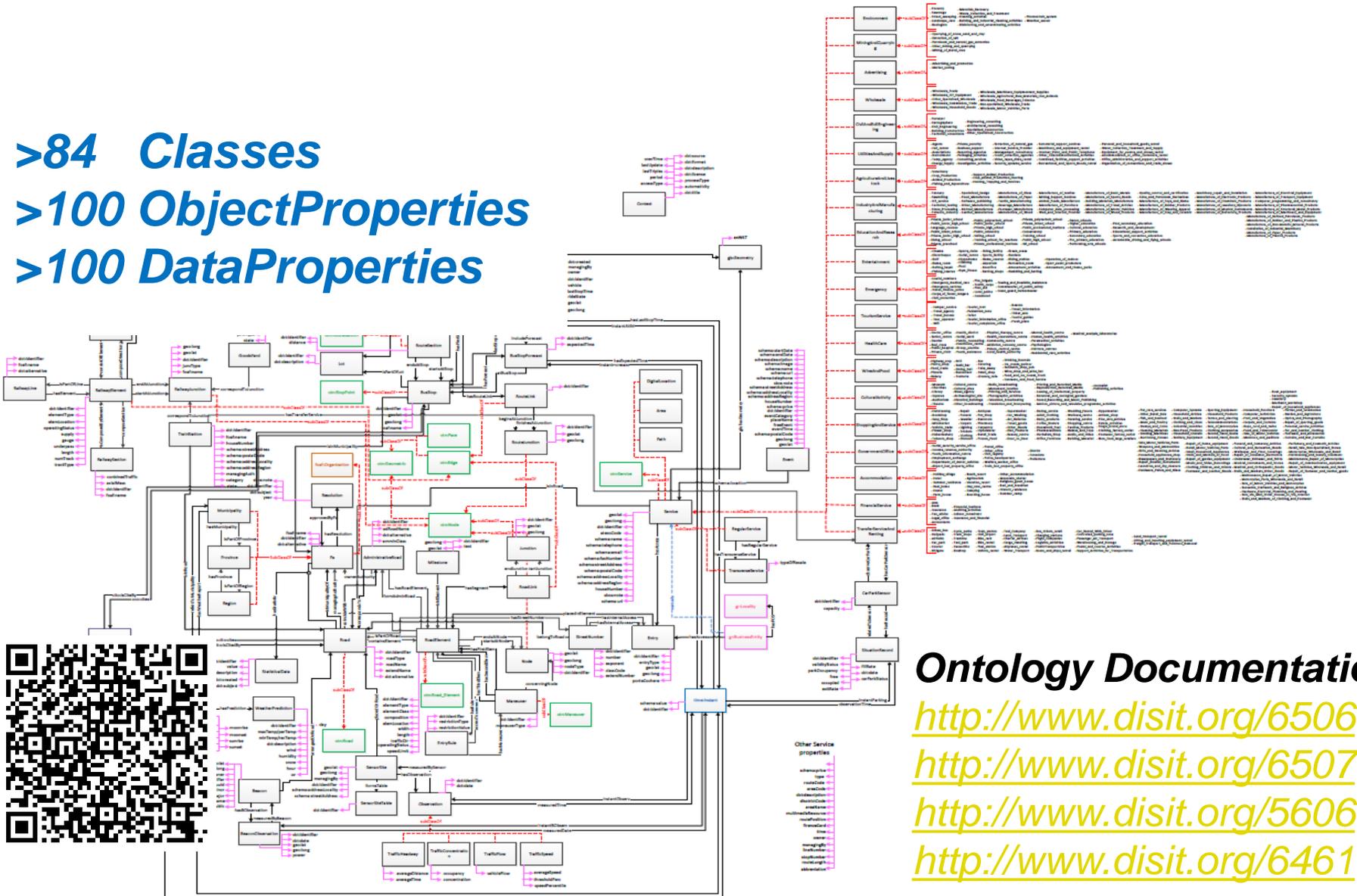
- **Smart Parking, in Tuscany**
- **Smart First Aid in Tuscany**
- **Smart Fuel pricing in Tuscany**
- **Smart search for POI and public transport srv.**
- **Public Transportation in Tuscany**
- **Routing in Tuscany, multimodal routing**
- **Social Media Monitoring and acting**
- **Traffic events and Resilience in Florence**
- **Bike Sharing in Pisa and Siena**
- **Recharge stations for e-vehicles**
- **Entertainment Events in Florence**
- **Traffic Sensors in Tuscany**
- **Weather forecast/condition in Tuscany**
- **Pollution and Pollination in Tuscany**
- **Origin Destination Matrices, trajectories**
- **People Monitoring Assessment in the City, in Florence via WiFi**
- **People Monitoring, in Tuscany via App**



**All Point of Interests, cultural activities, IOT, ...
Over than 1.2 Million of complex events per day!**

Smart-city Ontology km4city

>84 Classes
>100 ObjectProperties
>100 DataProperties



Km4City: Knowledge Base



- **Multiple DOMAINS**
- **Geospatial reasoning**
- **Temporal reasoning**
- **Metadata**
- **Statistics**
- **Risk and Resilience**
- **Licensing**
- **Open and Private Data**
- **Static and Real time**
- **IOT/IOE**



- Street-Guide
- Mobility and transport
- Points of interest
- Citations from strings
- Sensors, IOT, ...
- Energy
- Administration

Big Data Tools



LOD
and
reasoners

Ontology Documentation:

<http://www.disit.org/6506>

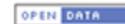
<http://www.disit.org/6507>

<http://www.disit.org/5606>

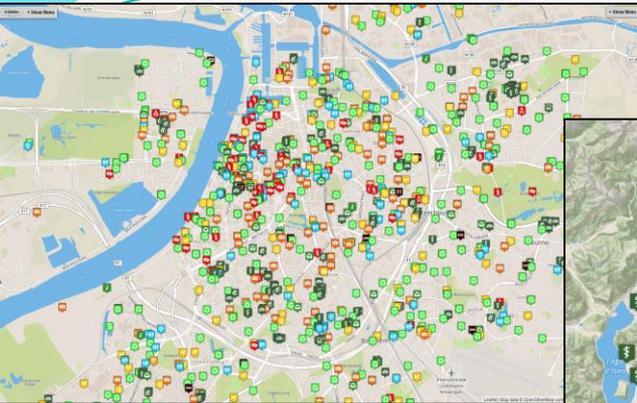
<http://www.disit.org/6461>



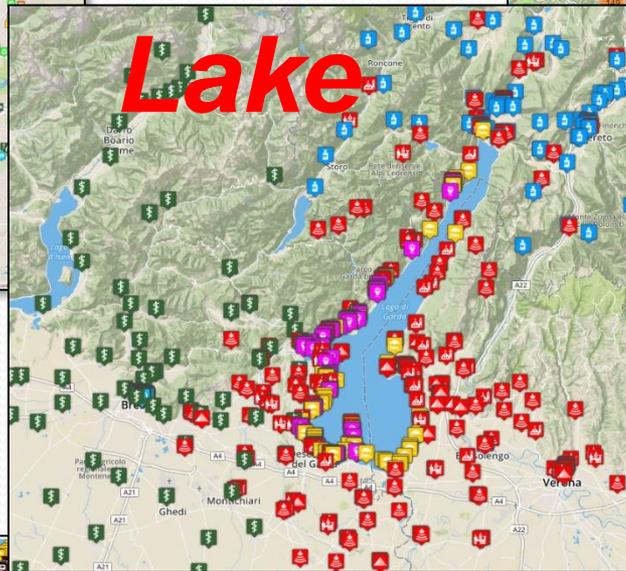
Schema: <http://www.disit.org/km4city/schema>
RDF version: <http://www.disit.org/km4city.rdf>



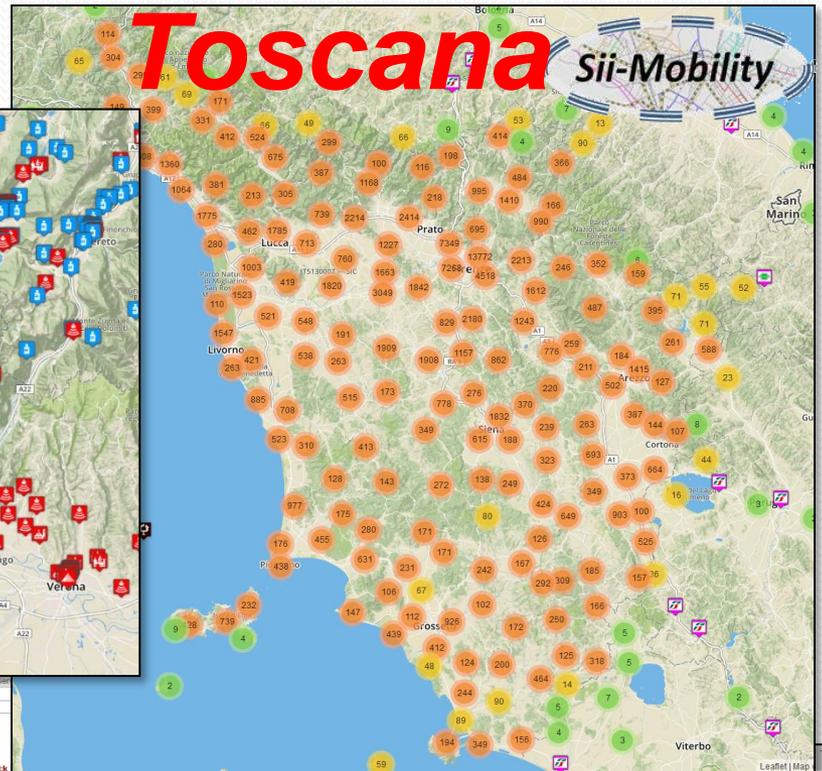
Antwerp Km4City in ...



Garda Lake



Toscana



Sii-Mobility

Sardegna

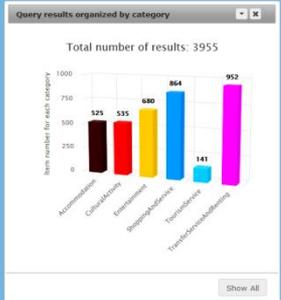


Public transport Municipalities Text Search Address Search Events

Select an agency:
- Select an Agency -
- Select a line -
- Select a route -
- Select a Bus Stop -

Position of selected Buses

Actual Selection:
Count: 99,230/9,11899
Address: Casteddu/Cagliari
Path from here Path to here



Filter:
search text into categories

Services Categories

- Accommodation
- Advertising
- AgricultureAndLivestock
- CivilAndEngineering
- CulturalActivity
- EducationAndResearch
- Emergency
- Entertainment
- Environment
- FinancialService
- GovernmentOffice
- HealthCare
- IndustryAndManufacturing
- MiningAndQuarrying
- ShoppingAndService
- TourismService
- TransportServiceAndRenting
- UtilitiesAndSupply
- Wholesale
- WineAndFood

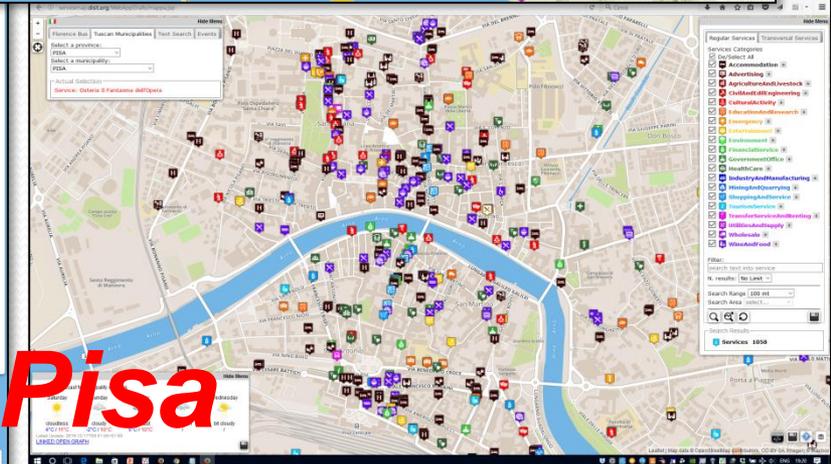
Filter:
search text into service

N. results: No Limit

Search range (visible area)
Search Area: select...

Search Results

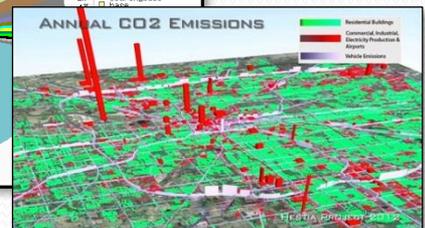
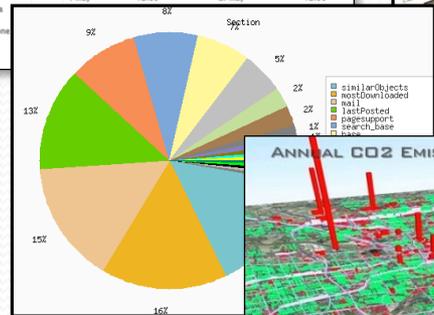
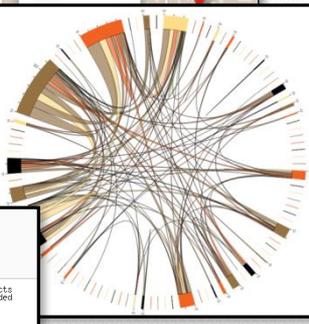
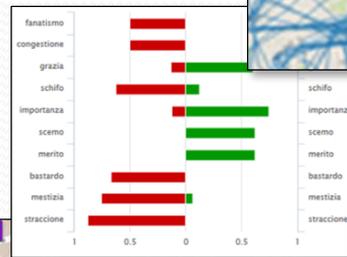
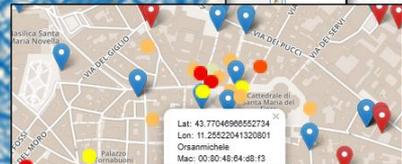
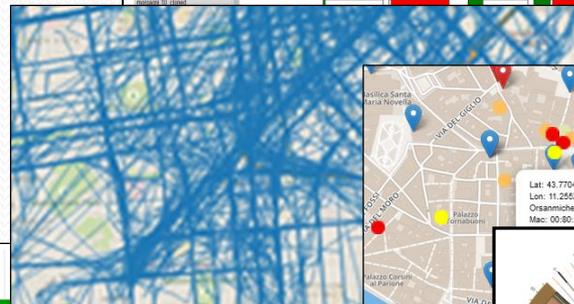
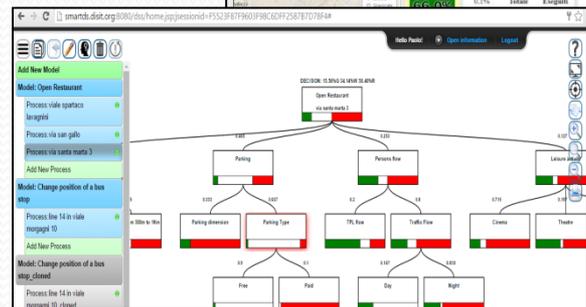
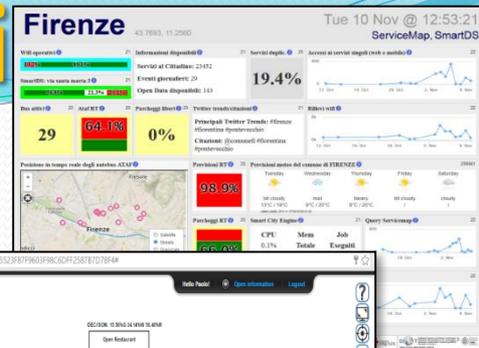
Services: 3690 of 3690 available



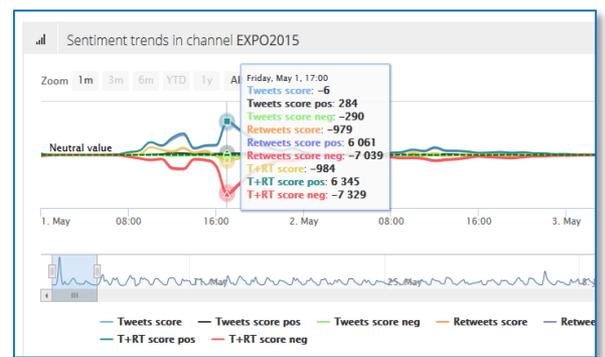
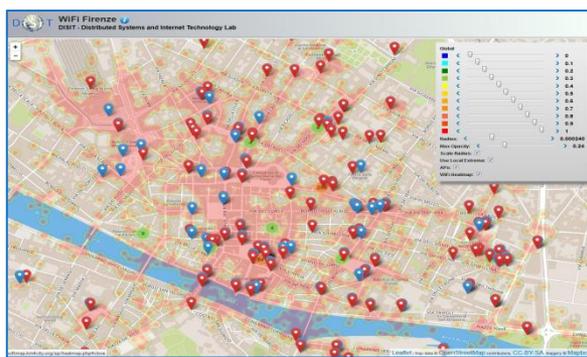
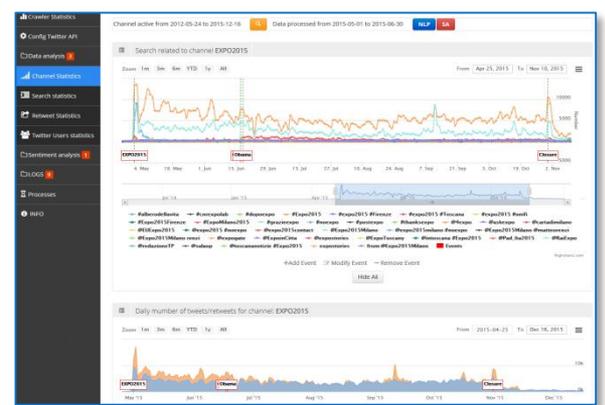
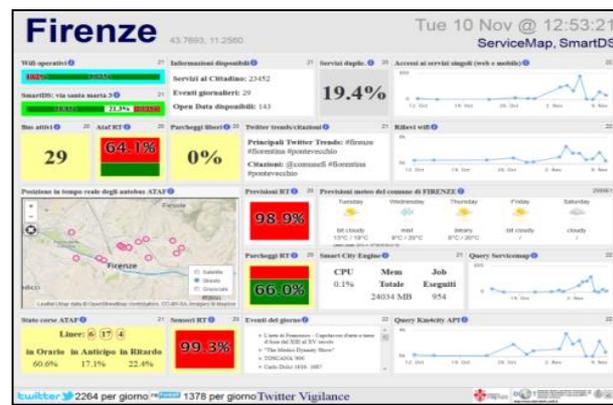
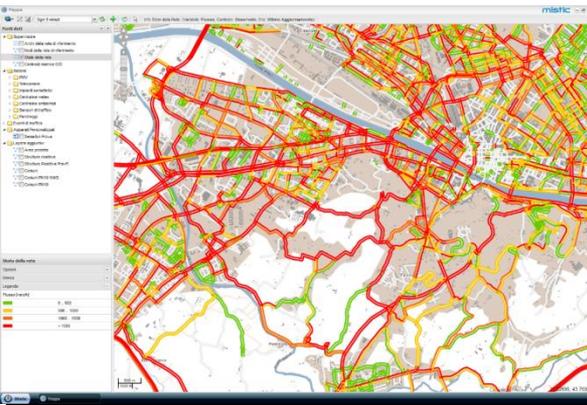
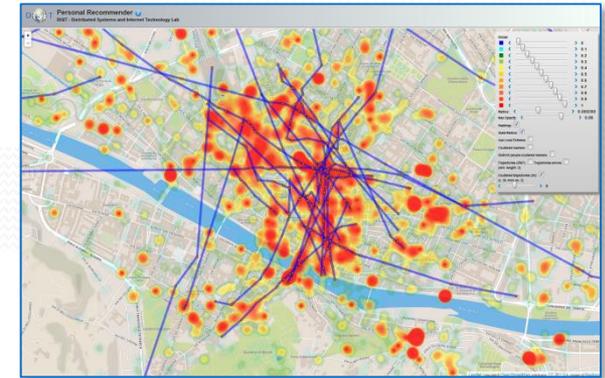
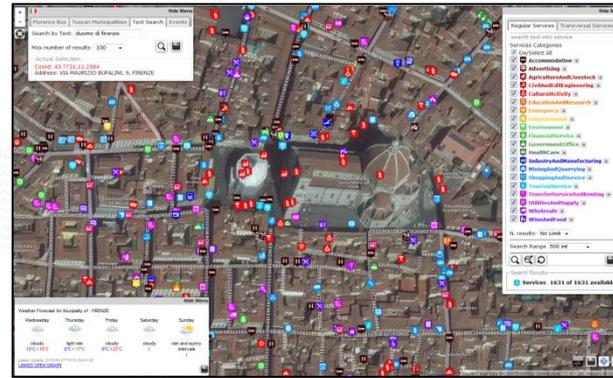
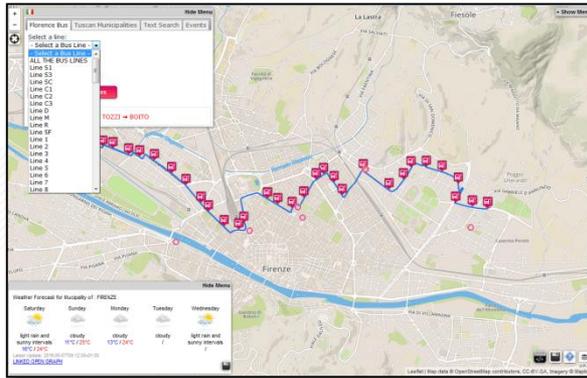
Pisa

Decisioni supportate dai dati periodiche ed in tempo reale

- **Condivisione e Integrazione Dati multi-dominio: *semantica e bigdata***
- **Dati → Smart City Engine → Control Room**
- **analisi: monitoraggio, flussi e comportamenti, sondaggi, mining, correlazioni, cause – effetti, etc.**
 - Per il miglioramento di servizi correnti
 - Per reagire ad eventi, incremento della resilienza,
 - Per la creazione servizi innovativi
 - ...



Smart City Dashboards



Smart City Dashboard



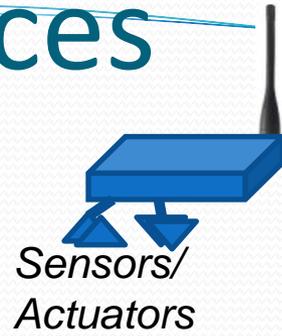
<https://main.snap4city.org/view/index.php?iddashboard=OTM5>

<https://main.snap4city.org/view/index.php?iddashboard=OTQz>

<https://main.snap4city.org/view/index.php?iddashboard=OTQy>

IOT Devices

IOT Edge Devices

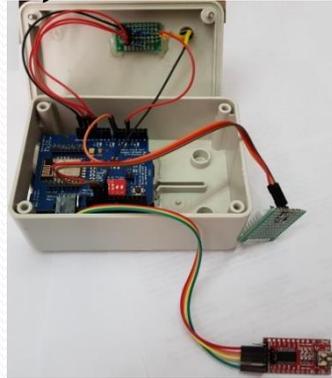


*SigFOX
Any and
Arduino*

*Arduino, Wi-Fi,
NGSI*

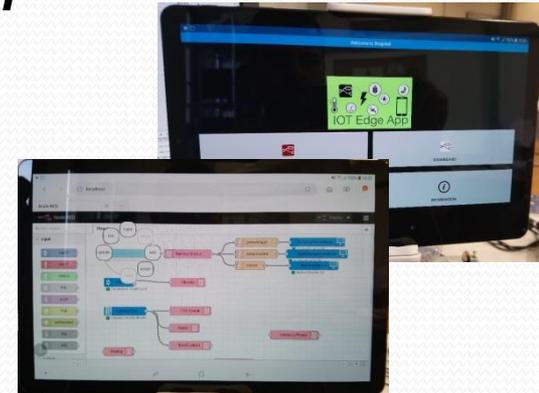
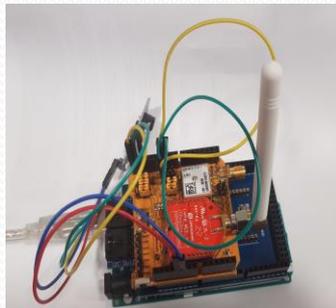
*IOT Edge
NodeRED:
Raspberry*

*IOT Edge
NodeRED
:
Android,
LINUX,
Windows*

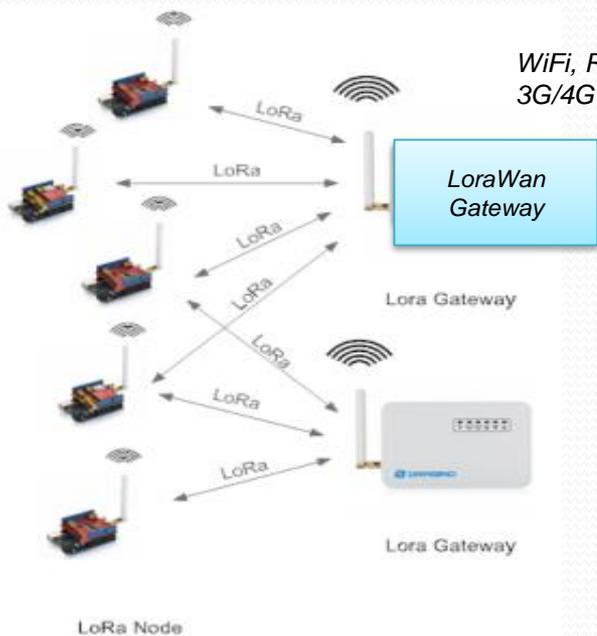
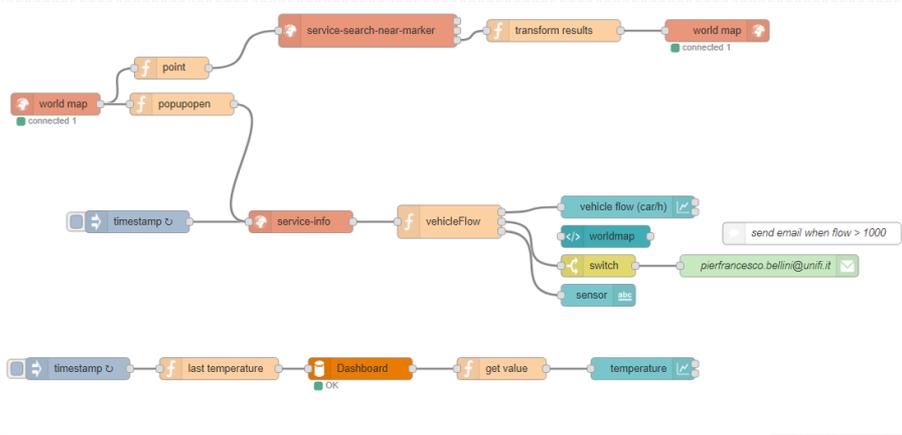


*Snap4All IOT
Button ESP*

*LoraWAN +
Arduino +
I2C, NGSI*



Internet of Things



IoT Applications

Snap4City

User: rootooladmin1, Org: DISIT
Role: RootAdmin, Level: 7

- Dashboards
- My Dashboards
- Notifier
- IoT Applications**
- My Personal Data
- IoT Directory and Devices
- Knowledge and Maps
- Micro Applications
- External Services
- Data Set Manager: Data Gate
- Resource Manager: Process Loader
- Development Tools
- Management
- Settings
- User Management and Auditing
- Help and Contacts
- Documentation and Articles
- My Profile
- Snap4City portal
- Km4City portal
- DISIT Lab portal

IOT Applications

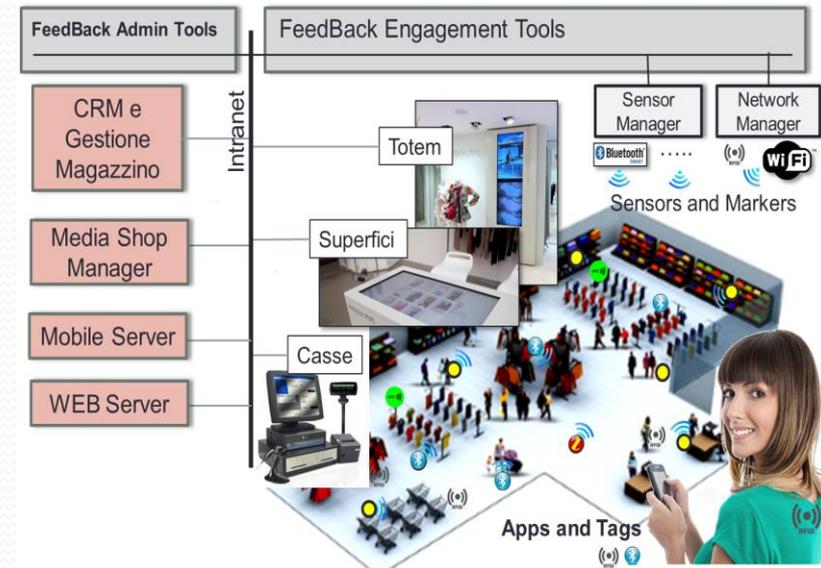
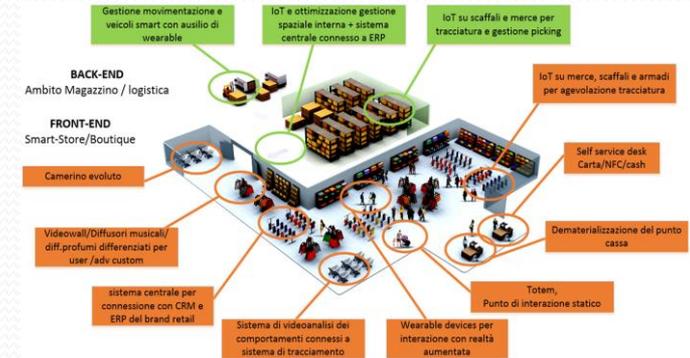
Prev 1 2 3 ... 9 Next

Filter

<p>2018-09-14T04:44</p>  <p>IOT Edge App</p> <p>owner: badii</p> <p>Management</p>	<p>2018-09-21T03:19</p>  <p>IOT Edge App</p> <p>owner: panesi</p> <p>Management</p>	<p>2018-10-19T16:07</p>  <p>IOT Edge App</p> <p>owner: pb3</p> <p>Management</p>	<p>2018-10-19T17:17</p>  <p>IOT Edge App</p> <p>owner: pb3</p> <p>Management</p>
<p>2018-10-22T11:57</p>  <p>IOT Edge App</p> <p>owner: semolarudy</p> <p>Management</p>	<p>application</p>  <p>IOT Application</p> <p>owner: tester5</p> <p>Management</p>	<p>Bib APP</p>  <p>IOT Application</p> <p>owner: semolarudy</p> <p>Management</p>	<p>ChargingStations</p>  <p>IOT Application</p> <p>owner: comunedashres</p> <p>Management</p>
<p>Deprecated - SiiMobilityControlRoom</p>  <p>IOT Application</p> <p>owner: badii</p> <p>Management</p>	<p>SamsungGalaxyS4Barcode</p>  <p>IOT Edge App</p> <p>owner: badii</p> <p>Management</p>	<p>esercitazione</p>  <p>IOT Application</p> <p>owner: tester2</p> <p>Management</p>	<p>lot-App</p>  <p>IOT Application</p> <p>owner: tester14</p> <p>Management</p>

Smart Retail and Human monitoring/engaging

- **Feedback Project, from Feb 2017**
 - Flexible Advanced Engagement Exploiting User Profiles and Product/Production Knowledge
 - VAR, PatriziaPepe (Tessilform), DISIT, Effective Knowledge, SICE
 - Keywords: retail, GDO, ...
- **Goals and drivers:**
 - adaptive user engagement, customer experience
 - Advanced user profiling, user behavior analysis
 - Predictive models for engagement
 - IOT and instrumentation
 - Integrated incity customer experience



Smart Manufacturing

- **Riduzione costi e incremento efficienza**
 - Automazione della manutenzione e della produzione
 - Navigazione indoor – Outdoor integrata
 - Ottimizzazione flussi per utensili, pezzi e materiali
- **Progetti Regionali con PMI e GI**
 - Frontman (Novicrom)
 - Green Capacity (ALTAIR)
 - Smart bed (Materassificio Montalese)

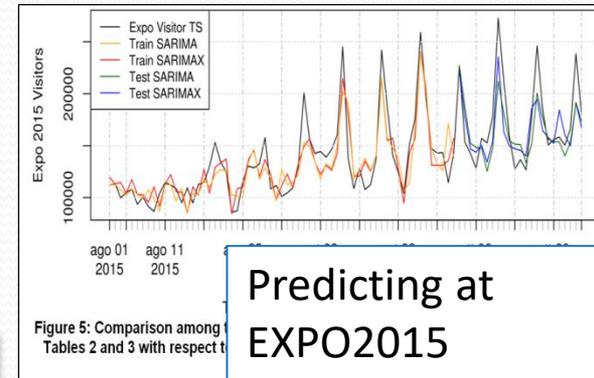


Analysing: Predicting, detection

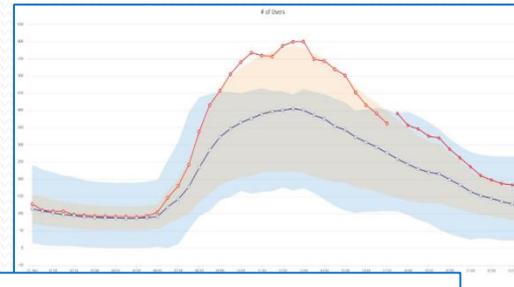
- Aiming at managing
 - Appreciation
 - User relationships
 - quality of service
 - workload
 - early warning/detection
 - Dysfunction
 - Habitudes



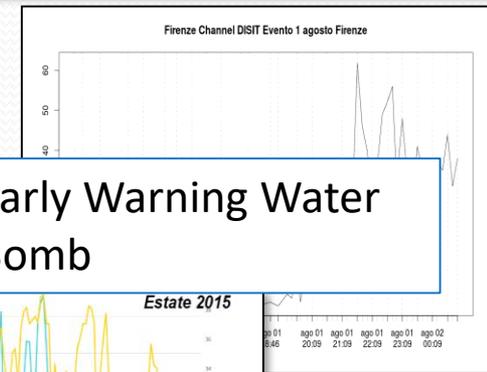
Predicting City Users on Areas



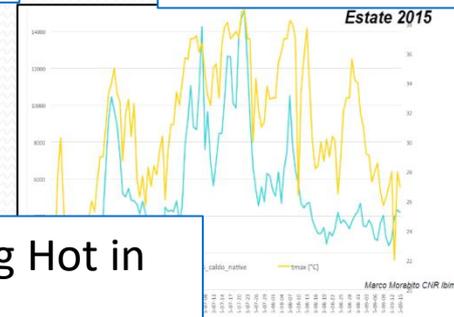
Predicting at EXPO2015



User behaviour



Early Warning Water Bomb



Early Warning Hot in Tuscany

Traffic Flow Tools



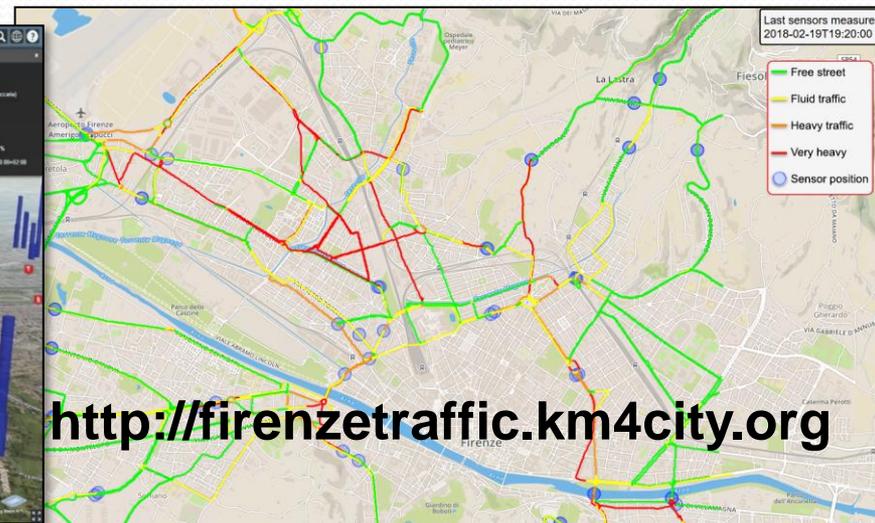
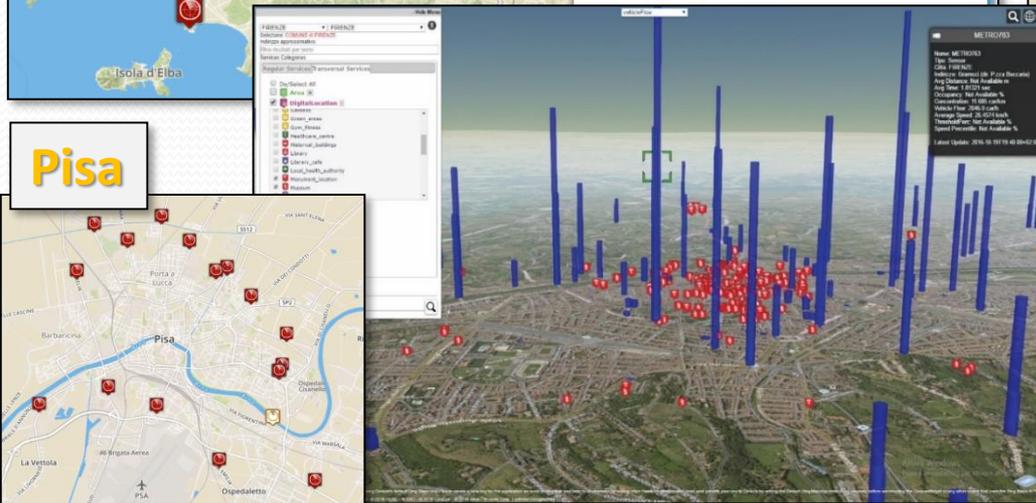
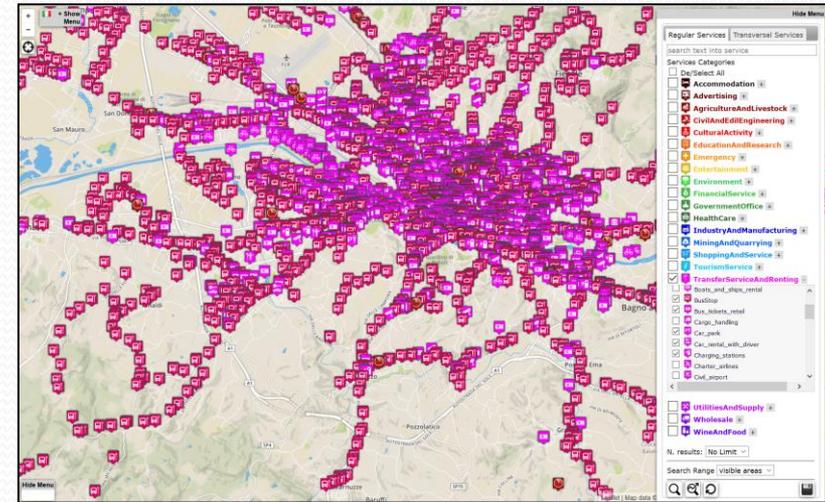
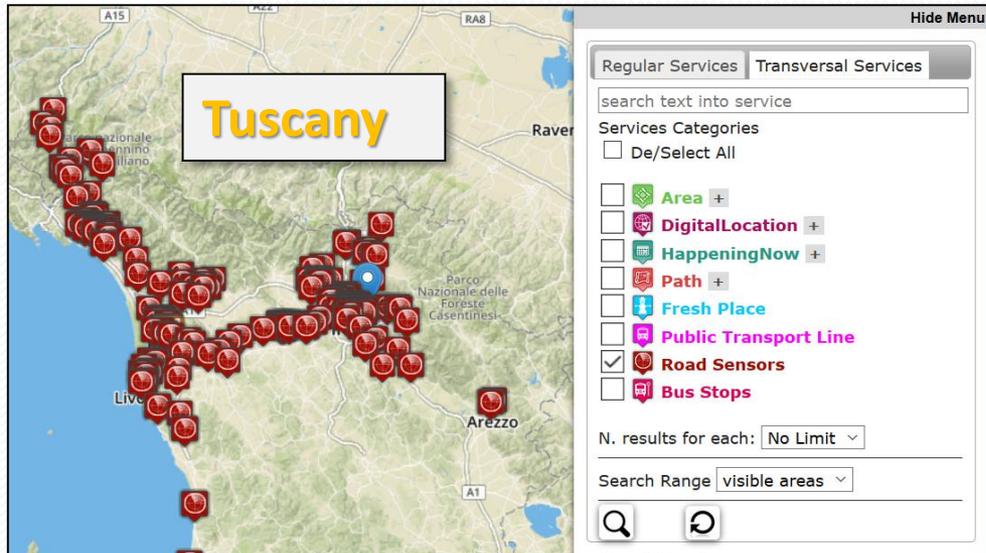
UNIVERSITÀ
DEGLI STUDI
FIRENZE

DINFO
DIPARTIMENTO DI
INGEGNERIA
DELL'INFORMAZIONE

DISIT
DISTRIBUTED SYSTEMS
AND INTERNET
TECHNOLOGIES LAB



- Spire and Virtual Spires (cameras), Bluetooth, ..
- Specifically located: along, around, ..



Toscana Traffico

Thu 1 Nov 14:15:47

Traffic Events (9m)

TEMPORARY TRAFFIC LIGHTS

05/11/2018 00:00:00 5

ORD. 2018-002375 - ISTITUZIONE DI SENSO UNICO ALTERNATO REGOLATO DA IMPIANTO SEMAFORICO MOBILE E/O MOVIERI, OLTRE ALLA LOCALIZZATA LIMITAZIONE DI VELOCITA' A 30 KM/H IN PROSSIMITA' DEL CANTIERE, PER IL RESTRINGIMENTO DELLA CARREGGIATA, SULLA S.P. N.56 'DEL BROLO E POGGIO ALLA CROCE' AL KM 16+600 CIRCA, IN LOCALITA' CAPANNUCCIA E SULLA S.P. N.34 'DI ROSANO', PER INTERVENTI PUNTUALI, DAL KM 3+440 AL KM 5+500 CIRCA, IN LOCALITA' VALLINA, NEL COMUNE DI BAGNO A RIPOLI (FI), DAL GIORNO 05/11/2018 AL GIORNO 16/11/2018 CON ORARIO 08:00/17:00.

Multi Map

METRO740

Last update: 2018-11-01 14:11:00-01:00

Description	Value	Buttons					
		Last value	Last 4 hours	Last 24 hours	Last 7 days	Last 30 days	
Avg Time	59.28916	Last value	Last 4 hours	Last 24 hours	Last 7 days	Last 30 days	
Concentration	10.46809	Last value	Last 4 hours	Last 24 hours	Last 7 days	Last 30 days	
Vehicle Flow	984.0	Last value	Last 4 hours	Last 24 hours	Last 7 days	Last 30 days	
Average	47.0	Last value	Last 4 hours	Last 24 hours	Last 7 days	Last 30 days	

Average Speed - 30 days (9m)

Selector

- Air Quality
- Bus Stops
- Cycle Paths Geometry
- Cycle Paths Pins
- Hot places heatmap
- Meteo Stations
- Parkings
- Recharging Stations - Normal
- Recharging Stations - Fast
- Traffic Sensors
- Traffic Flow Density

Florence Events (9m)

- CAMBIO DELLA GUARDIA A PALAZZO VECCHIO
- PER GRANDI E PUCCINI
- QUANTOUR EBENE, ALEXANDER LONQUICH
- "GLI STRUMENTI DI GALILEO" - A TUTTA SCIENZA!

<https://main.snap4city.org/view/index.php?iddashboard=MTE5MQ==>



Smart Cloud - Computing

- **Projects:** <http://www.disit.org/5501>
 - ICARO: <http://www.disit.org/5482>
 - Social Museum and Smart Tourism
- **Tools:** <http://www.disit.org/5489>
 - Smart Cloud Engine and reasoner <http://www.disit.org/6544>
 - Cloud ontology and tools: <http://www.disit.org/5604>
 - Configuration analysis and checker
 - Service Level Analyzer and control
 - Cloud Simulation, ICLOS
 - Cloud Monitoring, SM



<http://www.cloudicaro.it>

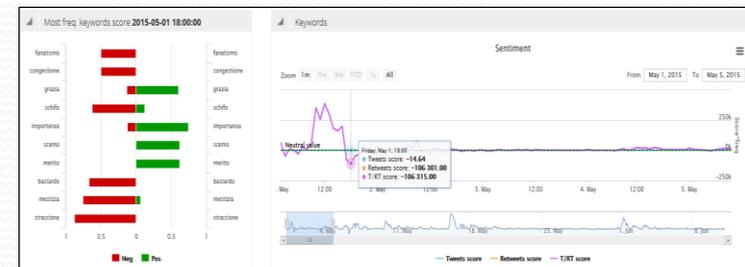
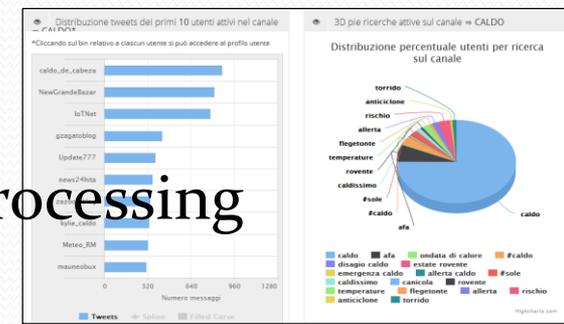
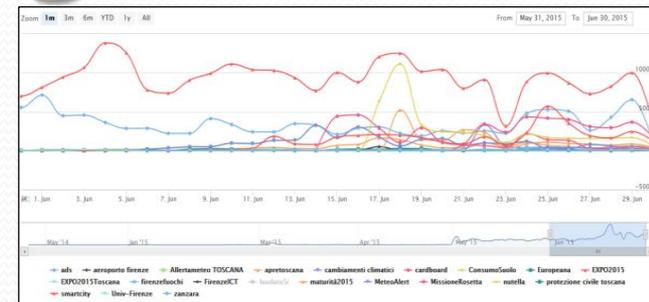


<http://www.disit.org/6588>



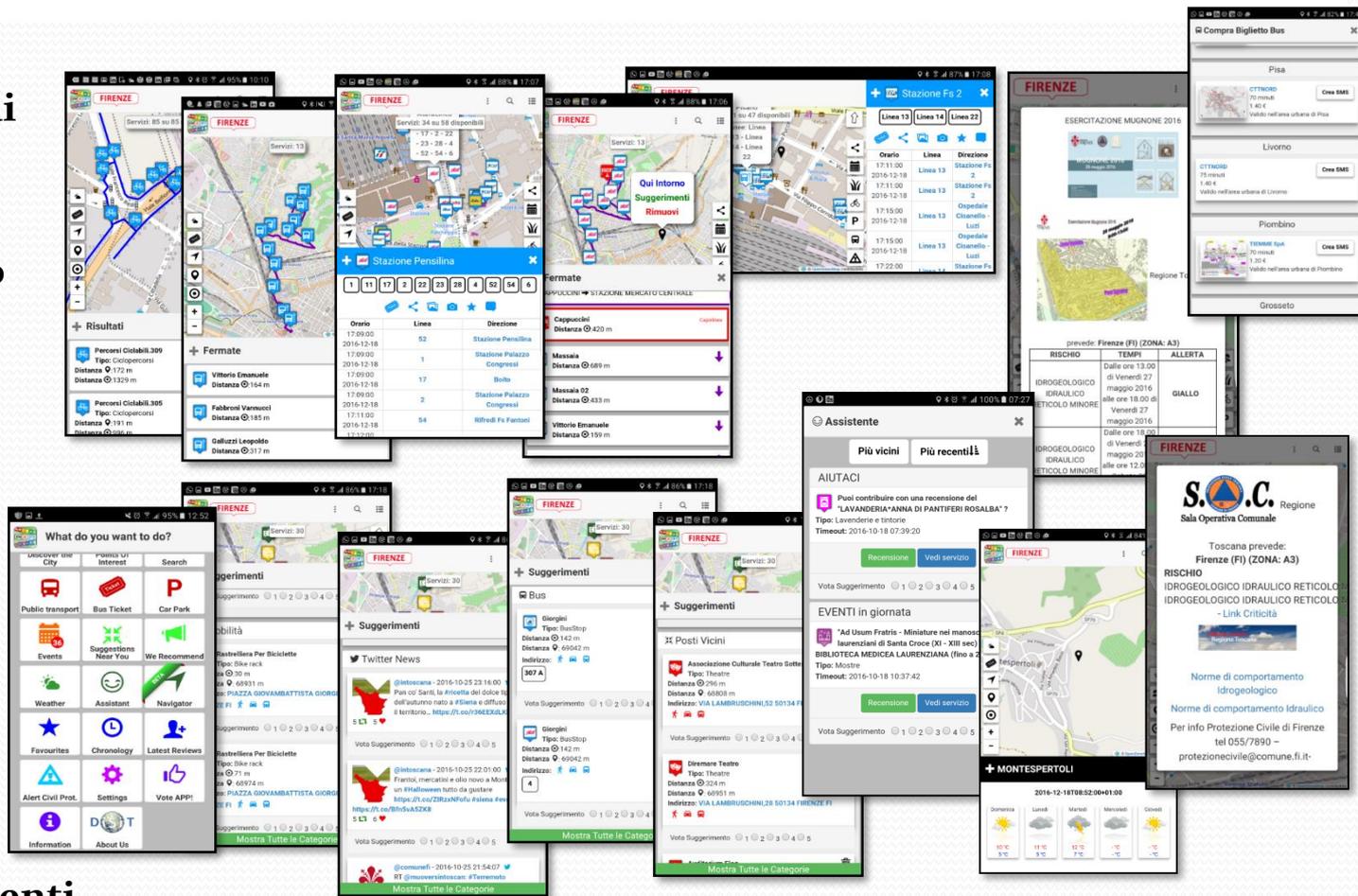
Text and Web Mining

- **Projects:** <http://www.disit.org/5501>
 - OSIM: <http://www.disit.org/5482>
 - SACVAR: <http://www.disit.org/5604>
 - Blog/Twitter Vigilance
- **Tools:** <http://www.disit.org/5489>
 - Text and web mining, Natural Language Processing
 - Service localization
 - Web Crawling
 - Competence analysis
 - Blog Vigilance, sentiment analysis



Toscana dove cosa,

- Tutta la Toscana
- Personalizzabile
- Profilata per tipo di utente
- Trasporto pubblico
- Traffico, percorsi, navigazione
- Parcheggi liberi
- Costi benzina
- Suggerimenti
- Assistenza
- Protezione civile
- Meteo
- Biglietti bus
- Punti di Interesse
- Contributi degli utenti



<http://www.km4city.org/app>

Introduzione al corso

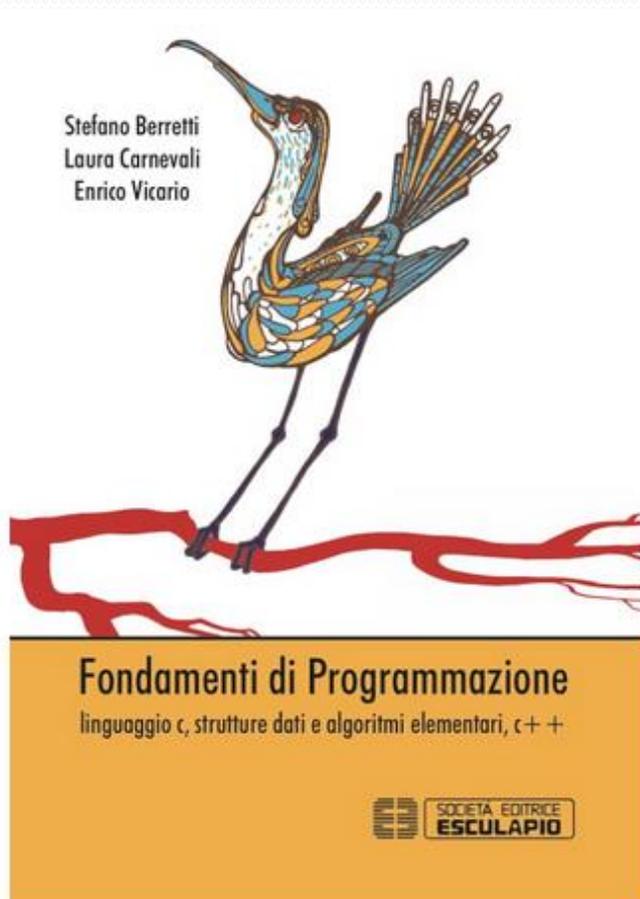


UNIVERSITÀ
DEGLI STUDI
FIRENZE

DINFO
DIPARTIMENTO DI
INGEGNERIA
DELL'INFORMAZIONE

DISIT
DISTRIBUTED SYSTEMS
AND INTERNET
TECHNOLOGIES LAB

Libro di testo



- Stefano Berretti, Laura Carnevali, Enrico Vicario
- Fondamenti di programmazione. Linguaggio C, strutture dati, algoritmi elementari, C++
- Editore: Esculapio
- ISBN: 9788893850513
- **NOTA: Queste slide integrano e non sostituiscono quanto scritto sul libro di testo.**

Programma del corso

Rappresentazione

- Introduzione al linguaggio C
 - Tipi, variabili e costanti
 - Operatori ed espressioni
 - Istruzioni
- Rappresentazione dei dati
 - Numeri
 - Interi senza segno
 - Caratteri
 - Interi con segno
- Definizione di un linguaggio
 - Sintassi di un linguaggio
 - Grammatica
 - Albero sintattico
 - Il metalinguaggio BNF
 - Semantica e sintassi
- Il linguaggio C
 - Tipi variabili e costanti
 - Operatori ed espressioni
 - Puntatori
 - Array
 - Istruzioni
 - Funzioni
 - Dati strutturati

Strutture dati e algoritmi elementari

- Liste
 - Rappresentazione in forma sequenziale
 - Rappresentazione in forma collegata con array e indici
 - Rappresentazione collegata con puntatori
 - Iterazione
- Cenni sugli alberi:
 - Alberi
 - Alberi binari di ricerca
 - Visita in forma ricorsiva
 - Ricerca
 - Inserimento ordinato
- Algoritmi di ordinamento su vettori
 - Sequential-sort

Applicazione degli algoritmi

- Excel
 - Operazioni/Formule di base
 - Filtri, Somma, Media, Max e Min, creazione formule semplici, ...
 - Analisi dati: Tabelle e Grafici
 - Macro



Pagina del Corso

<http://www.disit.org/drupal/?q=node/7020>

Qui trovate:

- AVVISI
- Slide del corso
- Approfondimenti

Modalità d'esame

– alcune linee guida -

- L'esame si compone di una prova scritta e una orale.
- La prova scritta è svolta su carta A4.
- Si accede alla prova orale solo se la parte di programmazione è corretta e funzionante
- La prova orale può essere sostenuta a partire dalla settimana seguente alla prova scritta, non oltre la prova scritta successiva.
- La prova orale inizia con la discussione dell'elaborato, e prosegue con l'approfondimento di tutti i contenuti del programma del corso.

Orario del Corso e Ricevimento

Insegnamento: FONDAMENTI DI INFORMATICA (A-L) (FonDiInf(A-L))

Ingegneria FIRENZE - A.A. 2018/2019 - 2° periodo

	Monday	Tuesday	Wednesday	Thursday	Friday	Saturday
08:15						
09:15						
10:15						
11:15	(Auditorium A - C.D.M.)					
12:15	(Auditorium A - C.D.M.)					
14:00					(Auditorium A - C.D.M.)	
15:00					(Auditorium A - C.D.M.)	
16:00					(Auditorium A - C.D.M.)	
17:00						
18:00						

Docenti: PAOLUCCI MICHELA

Legenda: C.D.M.: Centro Didattico Morgagni

- Il ricevimento si svolge su appuntamento contattando la docente via email:
 - michela.paolucci@unifi.it

Outline

- **Rappresentazione**

- Un frammento del linguaggio C ←

- Tipi, variabili e costanti
 - Operatori ed espressioni
 - Istruzioni

- Rappresentazione dei dati:

- Numeri
 - Interi senza segno
 - Caratteri
 - Interi con segno

Algoritmo (1)

- Il concetto di Algoritmo è uno dei concetti basilari della matematica
- Un algoritmo descrive un procedimento per risolvere una classe di Problemi in un numero finito di passi descritti in modo rigoroso
- Formalizzare un algoritmo significa trascrivere l'algoritmo da una scrittura informale ad una scrittura prettamente formale e, a volte, anche matematica
- Un algoritmo è un elenco finito di istruzioni, di passi, che devono essere eseguiti per arrivare alla soluzione finale del problema

Algoritmo (2)

- Esempi di algoritmi:
 - Per fare una torta c'è bisogno di una ricetta che deve essere seguita punto per punto
 - Per entrare in una Banca è necessario seguire una serie di linee guida
 - etc.
- Gli Algoritmi fanno uso di Dati in ingresso e sono in grado di produrre dei risultati che costituiscono la soluzione del problema in questione
- Se l'Algoritmo non produce risultati, serve a poco

DATI  **ALGORITMO**  **RISULTATI**

Algoritmo (3)

- Algoritmo formalizzato con la notazione matematica:

‘Calcolo delle radici di un polinomio di 2° grado’

Dato il polinomio $ax^2+bx+c=0$, si possono calcolare le radici x_1 e x_2 con il seguente algoritmo:

$$\Delta = b^2 - 4*a*c$$

$$x_{1,2} = (-b \pm \sqrt{\Delta}) / 2*a \quad \text{supponendo il } \Delta \geq 0 ;$$

Algoritmo (4)

- I dati costituiscono le informazioni che vengono fornite dall'esterno, dall'utente
- L'algoritmo acquisisce dei dati dall'esterno e comunica i risultati all'ambiente esterno
- I risultati sono il prodotto dell'algoritmo
- L'algoritmo descrive come i risultati vengono prodotti in base ai dati
- L'algoritmo è composto di passi e di relazioni tra passi
- L'algoritmo può essere eseguito dall'uomo o da macchine automatiche, oppure dall'uomo con l'ausilio di macchine automatiche
- L'algoritmo deve essere comprensibile per l'esecutore (uomo o macchina)
- Lo schema di esecuzione degli algoritmi è costituito da alcune regole che indicano l'ordine di esecuzione delle istruzioni che lo costituiscono

Algoritmo (5)

- Esempi di Algoritmi:
 - Andare a fare la spesa
 - Preparare una torta
 - Entrare in banca
 - Andare a iscriversi all'Università
 - Data una serie di numeri trovare il maggiore
 - Dati due numeri trovare il MCD
 - Dati due numeri trovare il mcm
 - ...

Elaboratore (1)

- L'Elaboratore è quella 'macchina' in grado di eseguire gli algoritmi forniti dall'uomo sulla base di dati diversi ma in modo automatico
- Una volta fornito un algoritmo dall'esterno (realizzato dall'uomo), spesso l'elaboratore è in grado di eseguirlo più velocemente dell'uomo
- Sull'elaboratore l'uomo può eseguire programmi diversi tra loro per tipologia e formulazione MA è necessario che venga usato un linguaggio di programmazione compatibile con l'elaboratore stesso
- L'elaboratore è in grado di operare solo in base a istruzioni redatte in un formato per lui eseguibile

Elaboratore (2)

- Un algoritmo deve essere descritto attraverso un linguaggio generalizzato costituito da strutture linguistiche definite
- Solitamente le proposizioni che vengono usate per gli algoritmi contengono una descrizione di:
 - i) operazioni che devono essere eseguite
 - ii) oggetti sui quali si devono effettuare le operazioni per ottenere i risultati voluti
- L'elaboratore è flessibile perché permette di eseguire algoritmi diversi semplicemente cambiando il programma (algoritmo)



Elaboratore (3)

- Un algoritmo, per essere eseguibile, deve essere comprensibile per chi lo esegue
- Gli elaboratori sono in grado di eseguire istruzioni SE queste sono opportunamente codificate
- Un insieme di istruzioni ordinate secondo un certo schema, che corrisponde all'implementazione di un algoritmo, può essere chiamato **Programma**
- Con programmi diversi verranno usati dati diversi e/o uguali in ingresso e producono risultati diversi e/o uguali ma sempre coerenti con il programma
- L'**Elaboratore** è una apparecchiatura AUTOMATICA, DIGITALE o ELETTRONICA capace di effettuare trasformazioni su dei dati in ingresso

Informatica (1)

- ‘Scienza che studia l’informazione e, più specificamente, l’elaborazione dei dati e il loro trattamento automatico per mezzo di computer’, Garzanti Linguistica
- ‘Scienza pratica che si occupa del trattamento dell’informazione mediante procedure automatizzabili’, Wikipedia
- ‘Scienza che studia l’elaborazione delle informazioni e le sue applicazioni; più precisamente l’i. si occupa della **rappresentazione**, dell’organizzazione e del trattamento automatico della informazione. Il termine i. deriva dal fr. *informatique* (composto di INFORMATion e automatIQUE, «informazione automatica») e fu coniato da P. Dreyfus nel 1962.’, Treccani

Informatica (2)

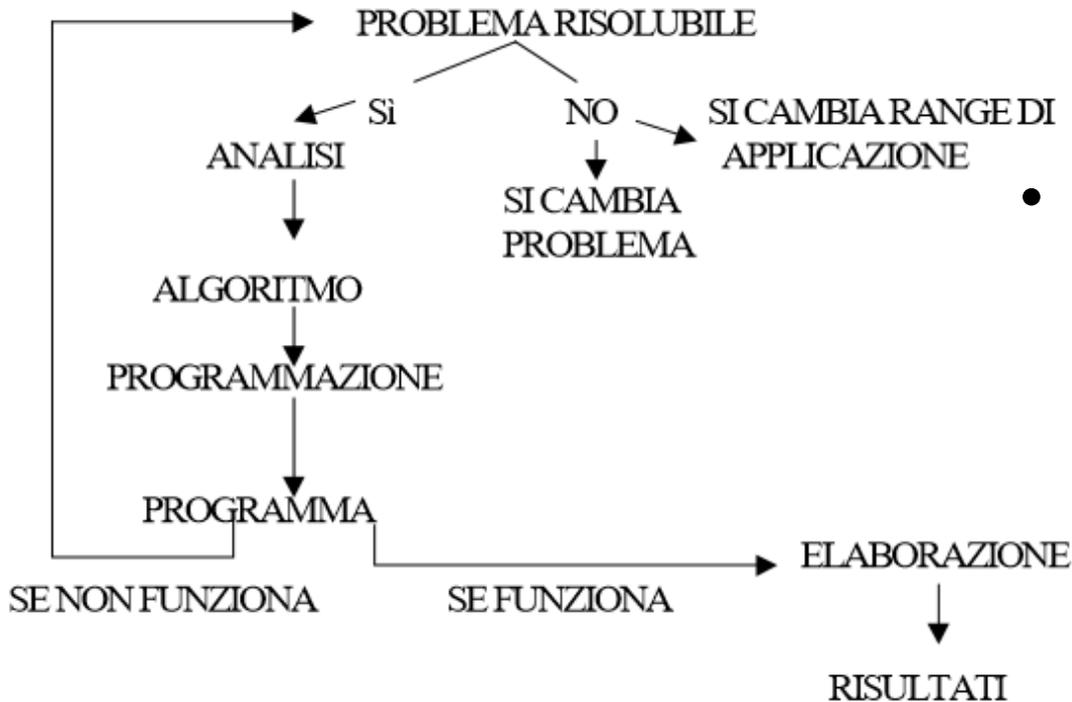
- L'informatica può essere anche definita come lo studio degli algoritmi che descrivono e trasformano l'informazione
- Per passare da un problema a un programma è necessario applicare una serie di procedimenti, principalmente:
 - Analisi
 - ProgrammazioneL'insieme di queste attività serve per risolvere un problema tramite l'elaboratore

Problema → **ALGORITMO** → **Programma**

Informatica (3)

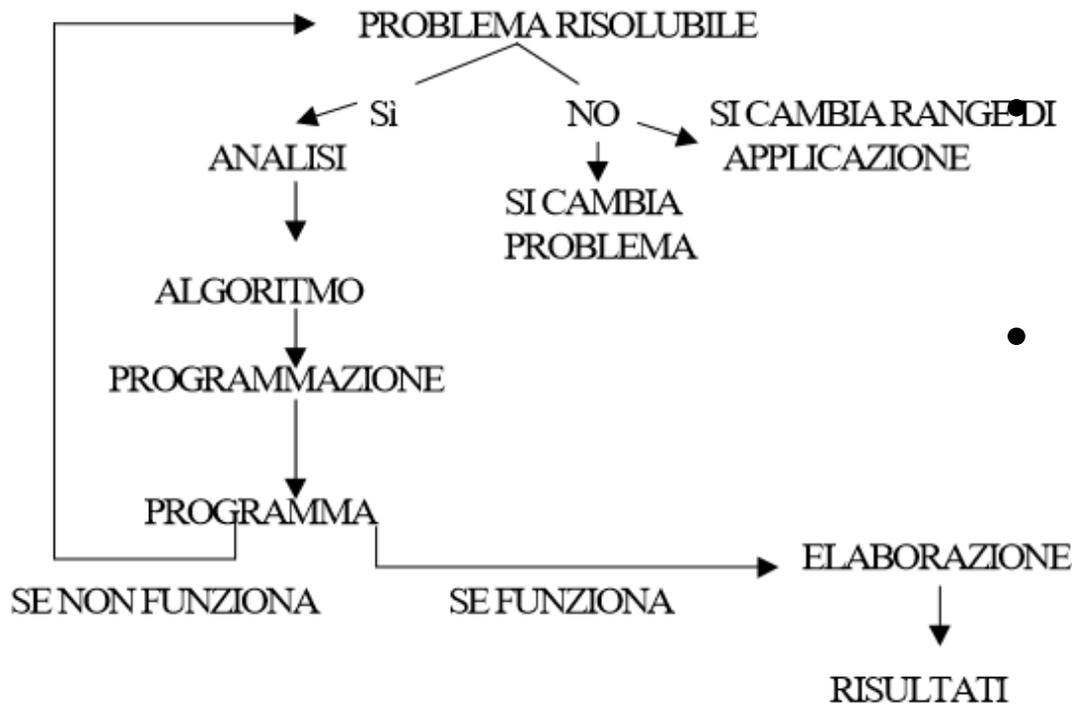
- Lo scopo dell'Analisi è quello di definire un algoritmo, cioè definire un elenco finito di istruzioni che determinano una serie di operazioni per ottenere una soluzione al problema
- Prima ancora di formulare l'algoritmo, si deve analizzare il problema e definire:
 - Se è risolvibile o meno
 - Se i dati rientrano nel dominio stabilitoTutto in base ai presupposti e ai dati iniziali

Informatica (4)



- Se il problema è risolvibile, si passa alle operazioni di analisi e alla definizione dell'algoritmo
- SE NON è risolvibile, ci sono due alternative:
 - Si cambia completamente il problema
 - Si riduce il range di applicazione e si controlla nuovamente la risolubilità del problema

Informatica (4)



- Una volta definito l'Algoritmo si passa alla Programmazione
- La programmazione ha come scopo la definizione di un programma ottimale
- Non è detto che il programma funzioni, in questo caso è necessario:
 - tornare al problema iniziale
 - modificarlo se necessario
 - ripetere il resto delle operazioni

Programma (1)

- Un Programma è una traduzione dell'algoritmo in un linguaggio comprensibile dall'elaboratore (Calcolatore) e deve essere in grado di:
 - Produrre il risultato voluto
 - Oppure dichiarare che la soluzione non è stata trovata
- Un programma è registrato nella memoria di un Elaboratore
- Cambiando il programma, si può mettere l'elaboratore in grado di risolvere problemi di natura diversa
- E' assolutamente necessario che la SINTASSI di ciascuna riga di codice sia assolutamente corretta
- Partiamo dalle basi... Come **rappresentare** i dati?

Rappresentazione (1)

- Il problema della rappresentazione consiste nel dare ad un algoritmo una forma tale che possa essere eseguita da un elaboratore
- In questo ambito assume un ruolo fondamentale l'uso di un linguaggio di programmazione
- In questo corso sarà affrontato il problema avendo come riferimento il linguaggio C



Rappresentazione (2)

- E' necessario capire come riuscire a formalizzare in maniera logica (Algoritmo) i concetti che vogliamo trasformare in programma;
- Questa operazione è una delle più importanti e complessa da effettuare
- Da questa fase dipende poi tutta la scrittura del programma
- Due strumenti fondamentali per formalizzare i problemi, per poi poterli trasformare in programmi leggibili e risolvibili dall'elaboratore sono:
 - Algebra di Boole (utile per le operazioni sui dati)
 - Diagrammi di Flusso o Flow Chart (utili per rappresentare in maniera ordinata i nostri processi logici).

Algebra di Boole (1)

- I fondamenti dell'Algebra Booleana sono stati delineati dal matematico inglese George Boole, nato a Lincoln nel 1815 e morto nel 1864, in un lavoro pubblicato nel 1847 riguardante l'analisi della logica e in particolare l'algebra della logica.
- L'Algebra di Boole è stata fondamentale nel campo della elettronica digitale, in particolare nella progettazione e realizzazione dei circuiti elettronici
- L'Algebra di Boole è fondamentale nell'informatica

Algebra di Boole (2)

- L'Algebra di Boole non è legata esclusivamente alla programmazione, ma fa parte della vita quotidiana
- Pensiamo ad esempio a quando vogliamo uscire di casa:
 - ***se piove non possiamo uscire;***
- Questo processo mentale, ovvero quello di verificare se piova o meno, può assumere, di fatto, due stati:
 - o è vero (e quindi non usciamo)
 - o è falso (e quindi usciamo).
- Oppure quando andiamo al cinema:
 - ***Se compriamo il biglietto, ci fanno entrare;***
 - Vero: ho comprato il biglietto ed entro
 - Falso: niente biglietto e niente ingresso al cinema

Algebra di Boole (3)

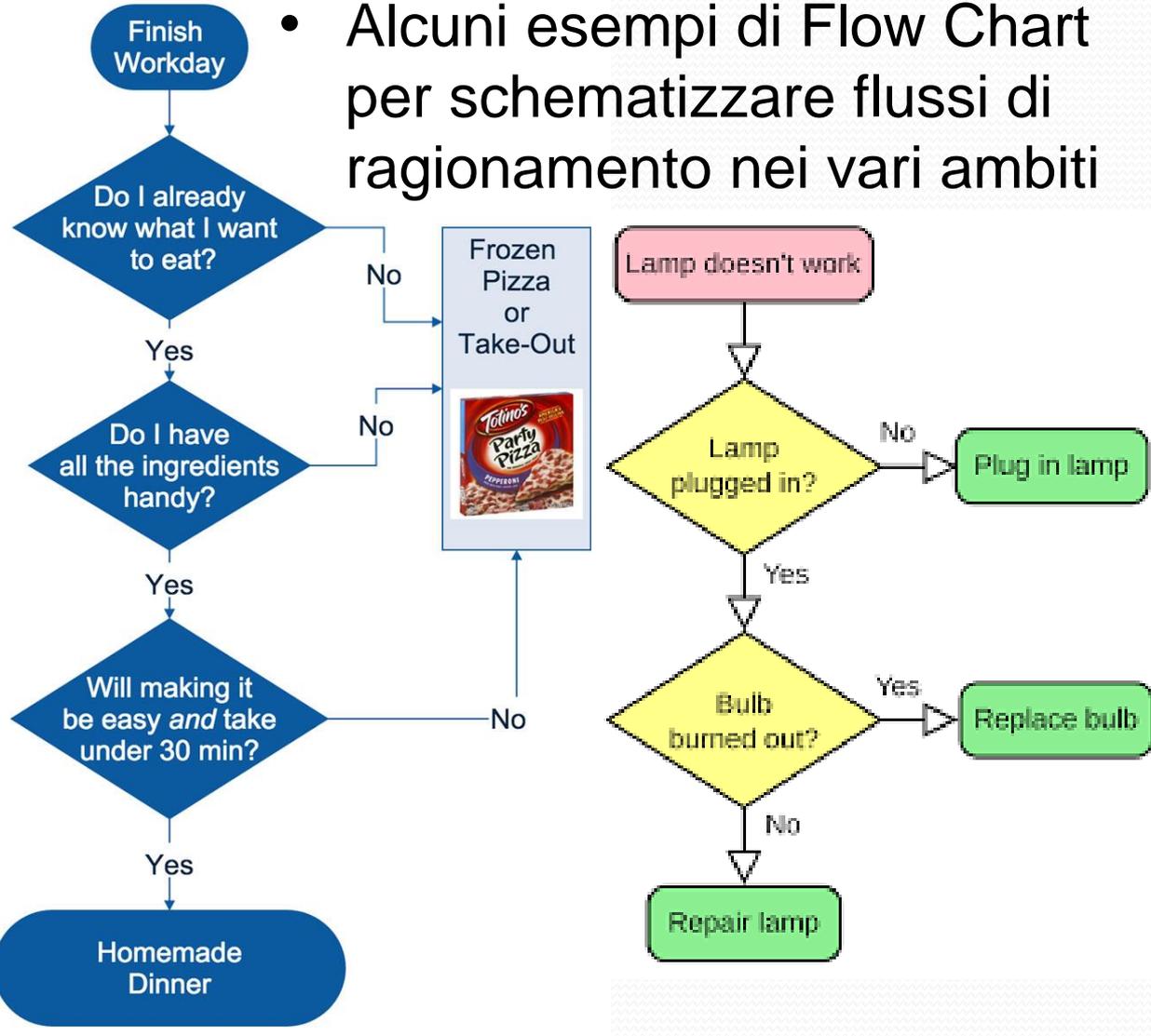
- Si può capire che se questo tipo di logica risulta utile nella vita quotidiana, lo è ancora di più per un computer
- L'Algebra di Boole, a differenza di quella tradizionale, è un'algebra **binaria**, ovvero le variabili possono assumere soltanto due stati
- Esempi:
 - 1/0;
 - v/f (vero, falso);
 - l/h (low, high);
 - t/f (true, false);
 - on/off; (acceso/spento)

Introduzione ai Flow Chart (1)

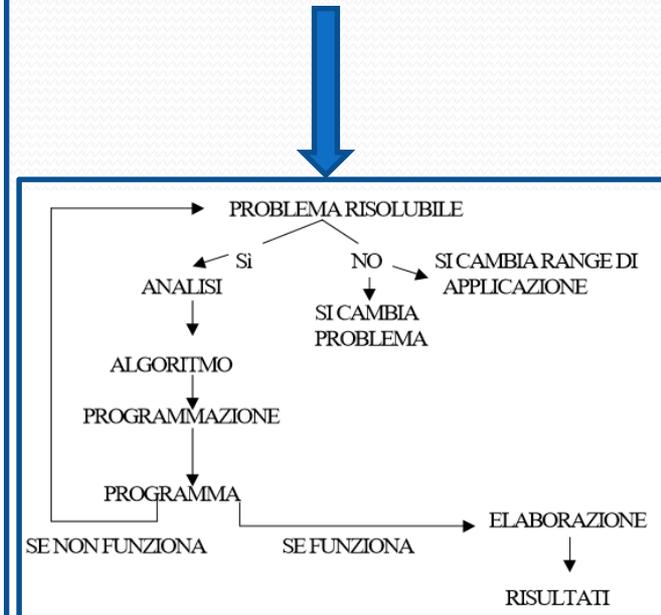
- I **Flow Chart** sono dei disegni/schemi che rappresentano graficamente un ragionamento rappresentandolo come un algoritmo
- Una rappresentazione schematica del problema permette:
 - una comprensione immediata del funzionamento del percorso logico che sta alla base della risoluzione del problema
 - un controllo accurato sulla funzionalità e la casistica del ragionamento

Introduzione ai Flow Chart (2)

- Alcuni esempi di Flow Chart per schematizzare flussi di ragionamento nei vari ambiti



- Ricordano lo schema visto per la formalizzazione di un algoritmo e di un programma



...Tornando al C

- Il nostro scopo è quello di analizzare, formalizzare e risolvere problemi avendo come riferimento il linguaggio C
- E' necessario quindi capire quali sono le prime basi di partenza per poi poter usare anche Algebra di Boole e Flow Chart per formalizzare i problemi
- Come possiamo quindi formalizzare i nostri problemi?
- Quali sono gli strumenti che il C ci fornisce?
- Come si possono rappresentare i dati che poi useremo per formalizzare i problemi?

Outline

- **Rappresentazione**
 - Un frammento del linguaggio C
 - Tipi, variabili e costanti ←
 - Operatori ed espressioni
 - Istruzioni
 - Rappresentazione dei dati:
 - Numeri
 - Interi senza segno
 - Caratteri
 - Interi con segno

Rappresentazione: Tipi di dati nel C (1)

- Per formalizzare un problema e scrivere un programma capace di risolverlo, è necessario gestire una serie di dati
- Il linguaggio C rappresenta i dati secondo **tipi**
- Il TIPO: è caratterizzato dai valori che può rappresentare e dalle operazioni che possono essere effettuate su tali valori
- I Tipi elementari del C sono:
 - int
 - float
 - char



Rappresentazione: Tipi di dati nel C (2)

- Alcuni esempi:
 - **int** (interi):
 - Devo tenere in considerazione il numero di ruote di una macchina → 4
 - Devo chiedere l'età ad un cliente → 30
 - **float**
 - Devo scrivere il risultato di una divisione per due di un numero dispari → $5/2 = 2.5$
 - **char**
 - Devo scrivere l'iniziale del mio nome → 'M'
 - Devo fare un programma per leggere da un file memorizzando un carattere alla volta → 'a'
 -

Rappresentazione: Tipi di dati nel C (3)

- I tipi booleani, introdotti con L'Algebra di Boole, non esistono in C
- Vengono sostituiti dai numeri **interi senza segno**, in base alla seguente convenzione:
 - **0** → codifica un **FALSO**
 - Qualunque valore **diverso da 0** → codifica un **VERO**

Rappresentazione: le Costanti (1)

- **COSTANTE:** è un valore che NON varia nel corso della computazione
- Si usa quando nel formalizzare e risolvere il problema si ha la necessità che ad un elemento usato nella computazione sia associato ('assegnato') sempre lo stesso valore
- Si usa quando NON deve essere possibile modificare tale elemento neanche per errore nel contesto della computazione in esame

Rappresentazione: le Costanti (2)

- Esempi
 - il numero di ruote di una macchina avrà sempre valore 4
 - Il mio codice fiscale NON potrà cambiare
 - Il mese di Aprile avrà sempre 30 giorni
- Se voglio calcolare la circonferenza e l'area di un cerchio supponendo di usare le prime 10 cifre del pigreco:
 - Associa alla costante '**pigreco**' tali cifre che NON dovranno MAI cambiare nel corso del mio programma
 - Poi effettua i vari calcoli usando la costante '**pigreco**'

Usare le Costanti in C (1)

- In C per formalizzare il concetto di costante si fa uso della direttiva '#define'
- Ad esempio, Se nel mio programma scrivo (una sola volta e solitamente a INIZIO del file):

```
#define MAXLIMIT 100
```

```
#define PIGRECO 3.1415926535
```

- Significa che:
 - Nessun tipo di calcolo può modificare le costanti
 - Al nome PIGRECO associo il valore 3.1415926535, ovvero uso solo le prime 10 cifre dopo la virgola (approssimazione)
- Tutte le volte che il programmatore scrive 'PIGRECO' nel programma, automaticamente il compilatore sostituirà il valore assegnato tramite la direttiva #define, che in questo caso è: 3.1415926535



Usare le Costanti in C (2)

- Se si vuole calcolare circonferenza e l'area di un cerchio di raggio $r=2$ cm, con le prime 10 cifre del π , in C potremmo usare allora le costanti:

```
#define PIGRECO 3.1415926535
```

```
#define R 2
```

```
//calcolo dell'area e della circonferenza usando 'PIGRECO' e 'R'
```

```
...
```

```
circonferenza = 2*PIGRECO*R;
```

```
area = PIGRECO*R*R;
```

- In seguito, approfondiremo le righe di codice viste sopra per ora si noti che:
 - Chi scrive il programma associa all'inizio il valore alle costanti
 - Il compilatore sostituisce tale valore tutte le volte che 'incontra' la costante ('**PIGRECO**' o '**R**')
 - Notare l'uso delle MAIUSCOLE per definire le costanti



Usare le Costanti in C (3)

- Supponiamo di voler calcolare la somma del numero di ruote di 2 macchine e 3 moto. E' possibile:
 - definire come costanti il numero di ruote di ogni tipo di veicolo (che ovviamente NON devono cambiare nel corso della computazione)
 - effettuare il calcolo

```
#define RMACCHINA 4
```

```
#define RMOTO 2
```

...

```
totale_ruote = 2*(RMACCHINA) + 3*(RMOTO);
```

- Il compilatore sostituisce i valori impostati all'inizio dal programmatore, tramite la direttiva **#define**
- Quindi verrà eseguita la seguente espressione:
 - $totale_ruote = 2*(4) + 3*(2) = 14$



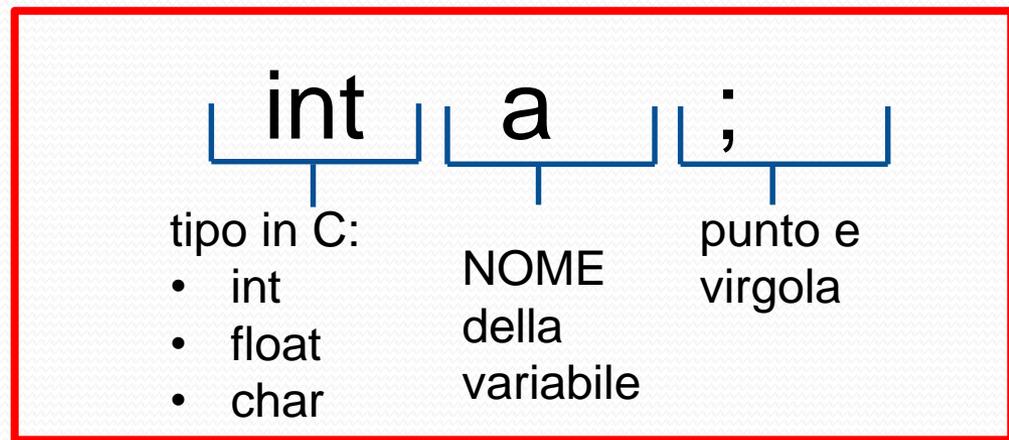
Concetto di VARIABILE (1)

- Quando si scrive un programma si ha la necessità di usare anche altre tipologie di elementi oltre alle COSTANTI. Ad esempio cosa succede se:
 - Si deve scrivere la data attuale su una pagina web?
 - Si devono visualizzare le previsioni del tempo per domani?
 - Si vuole scrivere l'algoritmo per calcolare il codice fiscale di una persona?
- In questi casi avrò bisogno di ricorrere al concetto di **Variabile**
- **VARIABILE**: è una locazione di memoria che contiene il valore di un tipo con le seguenti caratteristiche:
 - Il valore può cambiare nel tempo
 - Il tipo (int, char, float) è invariante



Concetto di VARIABILE (3)

- Il nome è associato alla VARIABILE tramite DICHIARAZIONE che:
 - Specifica il **TIPO**
 - Riserva lo SPAZIO in memoria
- Esempi di dichiarazioni:
 - **int** count;
 - **float** sum;
 - **char** c;



Concetto di VARIABILE (2)

- Il **nome** di una variabile deve rispettare le seguenti regole:
 - È una qualunque sequenza di caratteri alfabetici, numerici
 - È possibile usare il segno di underscore '_'
 - Il primo carattere non può essere un numero
 - La sequenza può avere una lunghezza arbitraria, il compilatore in realtà NON distingue tra nomi che non differiscono sui primi 32 caratteri

- Esempi:

int a1;

int totale_ruote;

float area;

float circonferenza;

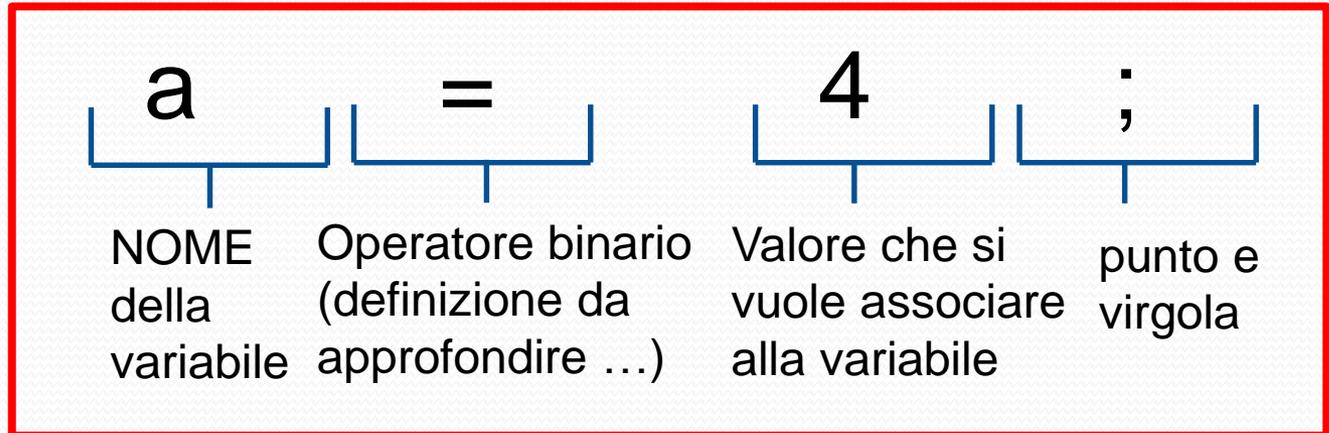


Dichiarazione e assegnazione (1)

- Si è detto che si effettua una **Dichiarazione** di una variabile quando, si associa al nome della variabile il tipo, in modo che il compilatore possa ‘creare’ la variabile associandogli un determinato spazio in memoria:
 - **int** a;
 - **float** numero_con_virgola;
 - **char** c1;
- Si parla invece di ‘Assegnazione’ quando si vuole associare, alla variabile un determinato valore:
 - a = 4;
 - numero_con_virgola = 2.5;
 - c1 = ‘L’;

Dichiarazione e assegnazione (2)

- Si parla invece di **'Assegnazione'** quando si vuole associare, alla variabile un determinato valore:
 - `a = 4;`
 - `numero_con_virgola = 2.5;`
 - `c1 = 'L';`



- Si calcola il valore scritto a destra dell'uguale (in questo caso ho il valore 4)
- Si assegna alla variabile (memorizzazione) che sta a sinistra dell'uguale, il valore calcolato che sta a destra

Esempi di assegnazione

...

```
float a; //dichiarazione di a
```

```
float b; //dichiarazione di b
```

...

```
a = 6.96; //assegnazione: a ha valore 6.96
```

```
b = a; //assegnazione: b ha lo stesso valore di a,  
//ovvero b ha valore 6.96
```

...

```
a = 4.23; // assegnazione: a ha valore 4.23
```

```
//notare che b ha sempre valore 6.96
```

```
//l'assegnazione agisce su una variabile alla volta
```

Differenza tra Variabile e Riferimento

- Una **VARIABILE** ha una propria identità che deriva dalla locazione di memoria ad essa associata
- E' poi possibile fare **RIFERIMENTO** alla variabile in vari modi come ad esempio attraverso il nome



Uso delle variabili in C (1)

- Se si vuole calcolare circonferenza e l'area di un cerchio di raggio $r=2$ cm, con le prime 10 cifre del π , in C potremmo usare allora le costanti:

```
#define PIGRECO 3.1415926535
```

```
#define R 2
```

```
...
```

```
//calcolo dell'area e della circonferenza usando 'PIGRECO' e 'R'
```

```
float area; //DICHIARAZIONE della variabile area
```

```
float circonferenza; //DICHIARAZIONE della variabile circonferenza
```

```
circonferenza = 2*PIGRECO*R; //assegnazione
```

```
area = PIGRECO*R*R; //assegnazione
```

- Si noti che quando si dichiarano le variabili (in questo caso float area e circonferenza):
 - Si 'dice' al compilatore con che 'tipo' di elemento si sta lavorando
 - Si definisce con precisione quanto spazio in memoria il compilatore deve riservare per le variabili usate

Uso delle variabili in C (2)

- Vogliamo calcolare la somma del numero di ruote di 2 macchine e 3 moto, possiamo:
 - definire come costanti il numero di ruote di ogni tipo di veicolo (che ovviamente NON devono cambiare nel corso della computazione)
 - effettuare il calcolo

```
#define RMACCHINA 4
```

```
#define RMOTO 2
```

...

```
int totale_ruote; // dichiaro totale_ruote come una variabile di  
                // tipo int
```

```
totale_ruote = 2*(RMACCHINA) + 3*(RMOTO); //assegnazione  
                //totale_ruote vale 14
```

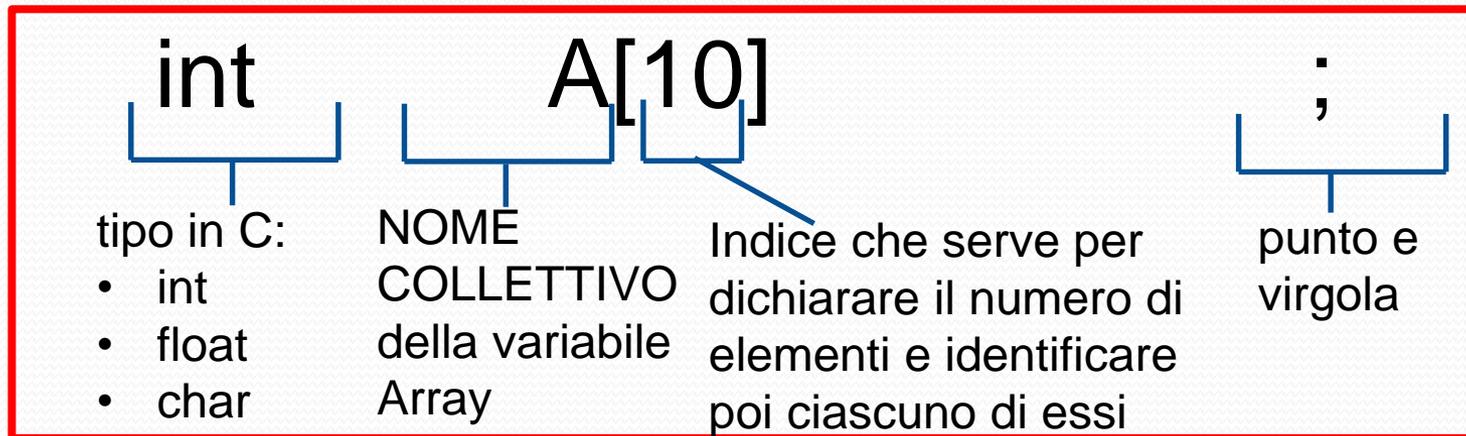
- In questo caso comunico al compilatore di riservare per la variabile **totale_ruote** uno spazio in memoria corrispondente a quello per un tipo intero

Concetto di **VARIABILE ARRAY** (1)

- Spesso ci si trova a dover risolvere problemi in cui non si deve trattare una variabile sola alla volta, come i casi precedenti (numero di ruote in una macchina, circonferenza di un cerchio, data di oggi, etc.)
- Ad esempio potrei avere problematiche del tipo:
 - Gestire 100 alunni in una classe
 - Creare una agenda telefonica
 - ...
- Quindi, cosa succede quando ho un insieme di variabili con caratteristiche comuni?
- Si fa ricorso al concetto di **ARRAY**

Concetto di VARIABILE ARRAY (2)

- **VARIABILE ARRAY:** è un insieme di variabili dello stesso tipo. Tali variabili possono essere referenziate tramite:
 - Un **nome collettivo**
 - Un **indice** che le identifica



- Nota sulle VARIABILI ARRAY:
 - Sono memorizzate in locazioni contigue della memoria

Concetto di VARIABILE ARRAY (3)

- **VARIABILE ARRAY**: è un insieme di variabili dello stesso tipo. Tali variabili possono essere referenziate tramite:
 - Un **nome collettivo**
 - Un **indice** che le identifica
- Le VARIABILI ARRAY sono memorizzate in locazioni contigue della memoria (il compilatore alloca un blocco di memoria)
- Dichiarazione di un array:
tipo nome_variabile[numero_elementi_array];
- Esempio di dichiarazione:
float A[100]; //dichiaro 100 variabili di tipo floating point

Concetto di VARIABILE ARRAY (4)

- Dichiarazione di un array:

tipo **nome_variabile**[**numero_elementi_array**];

- Esempio di dichiarazione:

float **A**[**100**]; //dichiaro 100 variabili di tipo floating point

- Si noti che:

- il primo elemento ha INDICE **0**
- L'ultimo elemento ha indice: **numero_elementi_array-1**

- Quando si usano i vari elementi dell'array si ha che:

A[**0**]; //indica il primo elemento di A, ovvero la prima
//**locazione** dell'Array A

A[**99**]; // indica l'ultimo elemento dell'array A, ovvero
// l'ultima **locazione** dell'array A

A[**100**]; // ERRORE! NON esiste

- Gli elementi di un Array (es: A[0]) si possono usare come le altre variabili definite in precedenza



Esempio: Uso degli ARRAY (1)

- Supponiamo di avere una agenda di 100 numeri telefonici e di voler stampare il terzo numero memorizzato nella agenda.
- Soluzione:
 - Fare uso di un array di 100 elementi (i 100 numeri hanno tutti caratteristiche simili)
 - Fare uso della **funzione** *printf* per stampare il numero di interesse
- Nota:
 - Torneremo in futuro a definire il concetto di funzione in C, per ora basta sapere che alcune operazioni sono state definite da altri programmatori e raccolte in 'librerie' che possiamo utilizzare in base alle necessità
 - Vedremo in dettaglio come fare...

Esempi d'uso della funzione printf (2)

- Ecco alcuni esempi d'uso della funzione **printf**
- Questa funzione è definita nella libreria C <stdio.h>, il cui uso verrà approfondito in futuro

- Esempio1, se scrivo:

```
printf("Hello World! ");
```

- Verrà stampata la frase "Hello World!"

- Esempio 2, se si scrive:

...

```
int anniSara; //dichiarazione
```

```
anniSara = 7; //assegnazione
```

```
printf("Ciao mi chiamo Sara e ho %d anni!", anniSara);
```

- Verrà stampata la seguente frase:
 - 'Ciao mi chiamo Sara e ho 7 anni!'
- NOTA: il compilatore sostituisce al %d la variabile di tipo

```
int anniSara
```

Esempi d'uso della funzione printf(3)

- Esempio 3, se si scrive:

```
char carattere; //dichiarazione
```

```
carattere = 'M'; //assegnazione
```

```
printf("Iniziale del mio nome: %c", carattere);
```

- Verrà stampata la seguente frase:
 - 'Iniziale del mio nome: M'
- NOTA: il compilatore sostituisce al %c la variabile di tipo char **carattere**

Esempi d'uso della funzione printf(4)

- Esempio 4, se si scrive:

```
float numero; //dichiarazione
```

```
numero = 2.5; //assegnazione
```

```
printf("Cinque diviso due: %f", numero);
```

- Verrà stampata la seguente frase:
 - 'Iniziale del mio nome: M'
- NOTA: il compilatore sostituisce al %f la variabile di tipo float **numero**

Esempio: Uso degli ARRAY (2)

- Supponiamo di avere una agenda di **100** numeri telefonici e di voler stampare il **terzo numero memorizzato** nella agenda.
- Soluzione:
 - Fare uso di un array di 100 elementi (i 100 numeri hanno tutti caratteristiche simili)
 - Fare uso della **funzione printf** per stampare il numero di interesse

```
int agenda[100];
```

```
... //supponiamo di riempire l'agenda, si approfondirà poi...
```

```
agenda[2] = '055999'; //esempio di assegnazione
```

```
//il primo elemento dell'array ha indice 0, quindi il terzo elemento  
//avrà indice 2
```

```
printf('Terzo numero memorizzato nella Agenda: %d', agenda[2]);
```

- Come risultato avrò la stampa della frase:
 - 'Terzo numero memorizzato nella agenda: 055999'
- Nota: **agenda[2]** è un elemento dell'array e viene trattato come una **variabile di tipo int**

Costanti, variabili e Algebra di Boole (1)

- Una volta chiari i concetti di costante e variabile, è possibile vedere altri esempi d'uso legati all'Algebra di Boole
- L'Algebra di Boole è uno degli strumenti più importanti per formalizzare i problemi e per poi poterli trasformare in programmi leggibili
- In C Non esiste un tipo Booleano, si è detto però che è possibile usare gli interi

Costanti, variabili e Algebra di Boole (2)

- Tornando agli esempi visti, uno è legato al tempo ‘Se piove non possiamo uscire’, ovvero è legato al processo mentale di verificare se piova o meno
- In un programma possiamo fare uso della variabile:
int oggi_piove;
 - che avrà valore:
 - **0**, FALSO se NON sta piovendo
 - **Diverso da zero** (ma sempre di tipo **int**), cioè VERO se sta piovendo. Solitamente si usa il valore **1**

Outline

- **Rappresentazione**
 - Un frammento del linguaggio C
 - Tipi, variabili e costanti
 - Operatori ed espressioni ←
 - Istruzioni
 - Rappresentazione dei dati:
 - Numeri
 - Interi senza segno
 - Caratteri
 - Interi con segno

Operatori ed Espressioni

- Una espressione è la combinazione di costanti e riferimenti a variabili tramite operatori
- Gli operatori sono classificati per 'tipo di operazione':

Aritmetici	Divisione	/
	Modulo	%
	Somma	+
	Differenza	-
	Prodotto	*
Relazionali	Minore	<
	Maggiore	>
	Maggiore o uguale	>=
	Minore o uguale	<=
	Uguaglianza	==
	Diverso	!=
Logici	Congiunzione	&&
	Negazione	!
	Disgiunzione	

Operatori Aritmetici

- L'operatore modulo (%), in C così come in altri linguaggi, fornisce il resto della divisione intera

Aritmetici	Divisione	/
	Modulo	%
	Somma	+
	Differenza	-
	Prodotto	*
	Incremento	++
	Decremento	--

- L'incremento ++ e il decremento -- aggiungono e sottraggono, rispettivamente, una unità al proprio operando
 - Sono Operatori Unari
- Gli operatori incremento e decremento agiscono in modo leggermente diverso in base alla posizione che hanno rispetto alla variabile:
 - Forma prefissa
 - Forma postfissa

Esempi di Operatori Aritmetici (1)

- Calcolare circonferenza e area di un cerchio di raggio $r=2$ cm, con le prime 10 cifre del π :

```
...
#define PIGRECO 3.1415926535
#define R 2
...
float area; //DICHIARAZIONE della variabile area
float circonferenza; //DICHIARAZIONE della variabile
circonferenza
area = 2*PIGRECO*R; // uso dell'operatore * Prodotto
circonferenza = PIGRECO*R*R //uso dell'operatore *
//Prodotto
...
```

Esempi di Operatori Aritmetici (2)

- Somma del numero di ruote di 2 macchine e 3 moto:

...

```
#define RMACCHINA 4
```

```
#define RMOTO 2
```

...

```
int totale_ruote; // dichiarazione
```

...

```
totale_ruote = 2*(RMACCHINA) + 3*(RMOTO); /* uso
```

Degli operatori aritmetici

Somma e Prodotto */

Esempi di Operatori Aritmetici (3)

```
int a; // serie di dichiarazioni di variabili di tipo int
int b;
int somma;
...
a=5; // assegnazioni
b=3;
...
somma = a + b; // assegnazione e uso dell'operatore
                // Somma +

//la variabile somma ha valore 8 che è un int
```

Esempi di Operatori Aritmetici (4)

```
float a; // serie di dichiarazioni di variabili di tipo int
```

```
float b;
```

```
float divisione;
```

```
...
```

```
a=9; // assegnazioni
```

```
b=6;
```

```
...
```

```
divisione = a / b;           // assegnazione e uso dell'operatore  
                             // Divisione
```

```
// la variabile divisione ha valore 1.5 che è un float
```

Operatori Aritmetici: incremento (1)

- **Postfisso** (modifica il valore della variabile **DOPO** l'utilizzo del valore nell'espressione):
 - Il valore dell'espressione `b++` è il valore `b`
 - `b` stesso è incrementato di uno

```
int a;
```

```
int b;
```

```
b = 6;
```

```
a = b++; //decremento postfisso -> PRIMA uso b con il  
//vecchio valore, ovvero 6 e POI incremento b  
//QUINDI: a ha valore 6 e b ha valore 7
```

Operatori Aritmetici: incremento (2)

- **Prefisso** (modifica il valore della variabile a cui sono applicati **PRIMA** che se ne usi il valore nell'espressione):
 - Il valore dell'espressione `++b` è il valore di `b` incrementato di 1
 - `b` stessa è incrementata di uno

```
int a;
```

```
int b;
```

```
b = 6;
```

```
a = ++b; //decremento prefisso -> PRIMA INCREMENTO b
```

```
    //POI uso b nella espressione
```

```
    //QUINDI: a ha valore 7 e b ha valore 7
```

Operatori Aritmetici: incremento (3)

```
int a;  
int b;  
b = 5;
```

```
a = ++b; //incremento prefisso -> PRIMA INCREMENTO b  
        //POI uso b nella espressione  
        //QUINDI: a ha valore 6 e b ha valore 6
```

```
a = b++; //incremento postfisso -> PRIMA uso b con il  
        //vecchio valore, ovvero 6 e POI incremento b  
        //QUINDI: a ha valore 6 e b ha valore 7
```

Operatori Aritmetici: incremento (4)

- **Postfisso** (modifica il valore della variabile **DOPO** l'utilizzo del valore nell'espressione):
 - Il valore dell'espressione $a++$ è il valore a
 - a stesso è incrementato di uno
- Esempio:

```
int a = 5;
```

```
int b = 30 / a++; /* a++ ha valore 5  
                  b ha valore 30/a++ cioè 30/5, cioè  
                  ha valore 6  
                  a ha valore 5+1 = 6  
                  */
```

NOTA: nella espressione, PRIMA USO la variabile **a** con il suo valore iniziale (5) e POI la incremento

Operatori Aritmetici: incremento (5)

- **Prefisso** (modifica il valore della variabile a cui sono applicati **PRIMA** che se ne usi il valore nell'espressione):
 - Il valore dell'espressione $++a$ è il valore di a incrementato di 1
 - a stessa è incrementata di uno

`int a = 5;`

`int b = 30 / ++a; /* ++a ha valore 6`

`b ha valore 30/++a cioè 30/6, cioè`
`ha valore 5`

`a ha valore 6 */`

NOTA: nella espressione **PRIMA** incremento il valore della variabile **a** (che assume il valore iniziale +1, cioè 6) e **POI** la uso

Operatori Aritmetici: decremento (1)

- **Postfisso** (modifica il valore della variabile **DOPO** l'utilizzo del valore nell'espressione):
 - Il valore dell'espressione `b--` è il valore `b`
 - `b` stesso è decrementato di uno

```
int a;
```

```
int b;
```

```
b = 5;
```

```
a = b--; //decremento postfisso -> PRIMA uso b con il  
//vecchio valore, ovvero 5 e POI decremento b  
//QUINDI: a ha valore 5 e b ha valore 4
```

Operatori Aritmetici: decremento (2)

- **Prefisso** (modifica il valore della variabile a cui sono applicati **PRIMA** che se ne usi il valore nell'espressione)
 - Il valore dell'espressione `--a` è il valore di `a` decrementato di 1
 - `A` stessa è decrementata di uno

```
int a;  
int b;  
b = 5;
```

```
a = --b; //decremento prefisso -> PRIMA DECREMENTO b  
        //POI uso b nella espressione  
        //QUINDI: a ha valore 4 e b ha valore 4
```

Operatori Aritmetici: decremento (3)

```
int a;  
int b;  
b = 5;
```

```
a = --b; //decremento prefisso -> PRIMA DECREMENTO b  
        //POI uso b nella espressione  
        //QUINDI: a ha valore 4 e b ha valore 4
```

```
a = b--; //decremento postfisso -> PRIMA uso b con il  
        //vecchio valore, ovvero 4 e POI decremento b  
        //QUINDI: a ha valore 4 e b ha valore 3
```

Operatori Aritmetici: decremento (4)

- **Prefisso** (modifica il valore della variabile a cui sono applicati **PRIMA** che se ne usi il valore nell'espressione)
 - Il valore dell'espressione `--a` è il valore di `a` decrementato di 1
 - `A` stessa è decrementata di uno

```
int a = 4;
```

```
int b = 12 / --a; /* --a ha valore 3
```

```
    b ha valore 12/--a cioè 12/3, cioè 4
```

```
    a ha valore 4-1 = 3
```

```
*/
```

NOTA: nella espressione **PRIMA** decremento **a** (che assume il valore iniziale -1, cioè 3) e **POI** la uso

Operatori di assegnazione composti (1)

- Operatore di assegnazione semplice (=)
 - $a = 4;$
- Operatori composti ($+=$, $-=$, $*=$, $/=$, ...):

valore $+=$ incremento; // valore = valore + incremento;

//ESEMPIO INCREMENTO:

```
int a;
```

```
a = 6;
```

```
a += 1;           // Equivale a scrivere: a = a + 1;
```

```
                // a ha valore 7
```

Operatori di assegnazione composti (2)

- Operatore di assegnazione semplice (=)
 - $a = 4;$
- Operatori composti ($+=$, $-=$, $*=$, $/=$, ...):

valore -= incremento; // valore = valore - incremento;

//ESEMPIO DIFFERENZA

```
int a;  
a = 6;  
a -= 1;           // Equivale a scrivere: a = a - 1;  
                 // a ha valore 5
```

Operatori di assegnazione composti (3)

- Operatore di assegnazione semplice (=)
 - $a = 4;$
- Operatori composti ($+=$, $-=$, $*=$, $/=$, ...):

valore $*=$ numero; // valore = valore * numero;

//ESEMPIO PRODOTTO

```
int a;  
a = 6;  
a *= 3;           // Equivale a scrivere: a = a * 3;  
                 // a ha valore 18
```

Operatori di assegnazione composti (4)

- Operatore di assegnazione semplice (=)
 - $a = 4;$
- Operatori composti ($+=$, $-=$, $*=$, $/=$, ...):

valore /= numero; // valore = valore / numero;

//ESEMPIO DIVISIONE

```
int a;  
a = 6;  
a /= 3;           // Equivale a scrivere: a = a / 3;  
                  // a ha valore 2
```

Operatori di assegnazioni composti e incremento/decremento (1)

```
int a;  
a = 3; //assegnazione tramite un valore intero  
a = a+1; //assegnazione tramite calcolo di una espressione  
//Le espressioni seguenti sono equivalenti, a vale 4
```

```
//operatore di assegnazione composto  
//valore += incremento; // valore = valore + incremento;  
a += 1; //a = a + 1; a ha valore 4+1 = 5
```

```
//incremento  
a++; //prima utilizzo a nella espressione in cui si trova e  
//poi la incremento di 1, in ogni caso come risultato  
//finale a avrà valore : a+1, ovvero 6
```

Operatori Relazionali (1)

Relazionali	Minore	<
	Maggiore	>
	Maggiore o uguale	>=
	Minore o uguale	<=
	Uguaglianza	==
	Diverso	!=

- Gli operatori relazionali e di uguaglianza (o disuguaglianza) confrontano il primo operando con il secondo per testare la validità della relazione in esame

Operatori Relazionali (2)

- Gli operatori relazionali sono operatori **binari**:

`<valore1> <operatore di relazione> <valore2>`

espressione relazionale

Risultato

- Il Risultato di una espressione relazionale è:
 - 0 se è falsa (false)
 - diverso da 0, (1) se la relazione testata è vera (true)
- Il **tipo** del risultato è **int**

Esempi di Operatori Relazionali (1)

- Maggiore $>$ e Minore $<$:

```
int a;
```

```
int b;
```

```
a=6; //assegnazione, a ha valore 6
```

```
b=a+1; //assegnazione, b ha valore 7
```

```
//Se scrivo  $a > b$ , ho come risultato un false, che  
//corrisponde al valore 0
```

```
//Se scrivo  $a < b$ , ho come risultato un true,  
//che corrisponde al valore diverso da zero (1)
```

Esempi di Operatori Relazionali (2)

- Maggiore o uguale \geq e Minore o uguale \leq :

int a, b, c;

a=6; //a ha valore 6

b=a+1; //b ha valore 7

c = 7; //c ha valore 7

//Se scrivo $a \geq b$, ho come risultato false, 0

//Se scrivo $a > b$, ho come risultato false, 0

//Se scrivo $a \leq b$, ho come risultato true, diverso da 0 (1)

//Se scrivo $c \leq b$, ho come risultato true, diverso da 0 (1)

//Se scrivo $c \geq b$, ho come risultato true, diverso da 0 (1)

Esempi di Operatori Relazionali (3)

- Maggiore o uguale \geq e Minore o uguale \leq :

int a, b, c;

a=6; //a ha valore 6

b=a+1; //b ha valore 7

c = 7; //c ha valore 7

//Se scrivo ...a \geq b+2... , ho come risultato false, 0

//Se scrivo a+2 >b, ho come risultato false, 0

//Se scrivo a \leq b, ho come risultato true, diverso da 0 (1)

//Se scrivo c \leq b, ho come risultato true, diverso da 0 (1)

//Se scrivo c \geq b, ho come risultato true, diverso da 0 (1)

Operatore di Uguaglianza (==) e Operatore di assegnazione (=) (1)

- Attenzione!! E' necessario fare distinzione tra il concetto di assegnazione (in C si usa '=') e l'operatore relazionale di Uguaglianza (in C '==')

//Concetto di assegnazione

```
int a;           //dichiarazione della variabile a
a = 5+3;        // assegnazione l'elaboratore calcola il valore
                //dell'espressione che sta a destra del simbolo
                //'=' e ne assegna il valore alla variabile a
```

//NOTA: L'operatore di assegnazione opera sempre da destra verso sinistra

Operatore di Uguaglianza (==) e

Operatore di assegnazione (=) (2)

- Attenzione!! E' necessario fare distinzione tra il concetto di assegnazione (in C si usa '=') e l'operatore relazionale di Uguaglianza (in C '==')

// Operatore Relazionale di uguaglianza

<valore1> <operatore di relazione> <valore2>

- Quando si scrive: ... a==b ... l'elaboratore effettua un confronto tra le due variabili a e b e rende come risultato:
 - 0 Se l'affermazione è falsa, ovvero se a è diversa da b
 - Diverso da 0 (solitamente 1), se l'affermazione è vera

Esempi di Operatori Relazionali (3)

- Operatore di Uguaglianza e Diverso:

```
int a;
```

```
int b;
```

```
a=6;
```

```
b=a+1; //assegnazione b ha valore 7
```

```
//Se scrivo a==b, ho come risultato un false
```

```
//ovvero a==b, ha valore 0
```

```
//Se scrivo a!=b, ho come risultato un true,
```

```
//ovvero a!=b, ha valore diverso da 0, solitamente 1
```

Operatori Logici (1)

- Sono introdotti dall'Algebra di Boole

- I tre operatori di base sono:
 - AND (&&), OR (||) e NOT (!)

Logici	Congiunzione	&&
	Negazione	!
	Disgiunzione	

- notazioni “linguistica” matematica aritmetica
- prodotto logico a **AND** b $a \wedge b$ $a \cdot b$
- somma logica a **OR** b $a \vee b$ $a + b$
- negazione **NOT** a $\neg a$ \bar{a}
- vero true T 1
- falso false \perp 0
- La notazione linguistica è usata in molti linguaggi di programmazione
- la notazione aritmetica enfatizza la similarità tra l'algebra booleana e l'algebra dei numeri.

Operatori: NOT (1)

- **NOT** A è una funzione logica che inverte il valore logico del suo operando
- È un operatore unario

A	NOT A
F	V
V	F

- In C, L'operatore NOT riceve in ingresso un valore (A) e lo interpreta come:
 - Falso, se 0
 - Vero, se diverso da 0
- In C, L'operatore NOT restituisce:
 - 0, se NOT A è falso
 - Diverso da 0, se NOT A è vero

Operatori: NOT (2)

- In C NOT A, si scrive !A

<i>A</i>	NOT A
<i>F</i>	<i>V</i>
<i>V</i>	<i>F</i>

- Esempio in C:
 - ! espressione
 - Restituisce il valore booleano **true/Vero (1)** se l'espressione è **false/Falsa**
 - Restituisce il valore booleano **false/Falso (0)** se l'espressione è **true/Vero**

Operatori: NOT – esempi (1)

```
int a, b, c;
```

```
a=6;
```

```
b=a+1; //assegnazione: b ha valore 7, a ha sempre valore 6
```

```
c=7;
```

```
// !(a==b)   rende true (1), poiché a è diverso da b  
// !(a<b)    rende false (0), poiché a è minore di b  
// !(c==b)   rende false, poiché c ha lo stesso valore di b  
//!(c>=b)   rende false, poiché c ha lo stesso valore di b  
//!(c<=b)   rende false, poiché c ha lo stesso valore di b  
//!(a>=b)   rende true, poiché a è maggiore di b  
//!a        rende true, perché a vale 6 e viene interpretato  
//come tale perché diverso da zero
```

Operatori: NOT – esempi (2)

```
int a;
```

```
int b;
```

```
a=6;
```

```
b=a+1; //assegnazione: b ha valore 7, a ha sempre valore 6
```

```
//Attenzione a NON confondere l'operatore Logico NOT
```

```
//con l'operatore Relazionale DIVERSO
```

- NOT:
 - $!(a > b)$ rende true/vero, poiché l'espressione dentro le parentesi è falsa
 - $!a$ rende false, poiché a è diverso da zero ovvero è considerato true/vero
- DIVERSO:
 - $a != b$ rende true/vero

Operatori: AND (1)

- A **AND** B è una funzione logica che è vera solo se entrambi gli operandi A e B sono veri
- L'operatore AND è binario

<i>A</i>	<i>B</i>	<i>A AND B</i>
<i>F</i>	<i>F</i>	<i>F</i>
<i>F</i>	<i>V</i>	<i>F</i>
<i>V</i>	<i>F</i>	<i>F</i>
<i>V</i>	<i>V</i>	<i>V</i>

- In C, L'operatore AND riceve in ingresso due valori (A e B) e interpreta ciascuno di essi come:
 - Falso, se 0
 - Vero, se diverso da 0
- In C, L'operatore AND rende:
 - 0, se l'espressione valutata (si veda tabella) è falsa
 - diverso da 0, se l'espressione valutata (tabella) è vera

Operatori: AND (2)

- In C, A **AND** B, si scrive: A && B

Ovvero: <espressione> && <espressione>

<i>A</i>	<i>B</i>	<i>A AND B</i>
<i>F</i>	<i>F</i>	<i>F</i>
<i>F</i>	<i>V</i>	<i>F</i>
<i>V</i>	<i>F</i>	<i>F</i>
<i>V</i>	<i>V</i>	<i>V</i>

- A && B rende:
 - Restituisce vero/true (diverso da 0) se entrambe le espressioni valutate sono vere/true (diverse da 0)
 - false (0), in caso contrario, ovvero se almeno una delle due espressioni valutate è falsa

Operatori: AND - esempi

```
int a, b, c;
```

```
a=6;
```

```
b=a+1; //assegnazione b ha valore 7
```

```
c=7;
```

```
//... ((c==b) && (a<b) )... rende true... ((true) && (true))
```

```
//... ((c!=b) && (a<b) )... rende false... ((false) && (true))
```

```
//... ((c==b) && (c>b)) ... rende false... ((true) && (false))
```

```
//... ((c==b) && (c<b)) ... rende false... ((true) && (false))
```

```
//... ((a<=b) && (c==b)) ... rende true... ((true) && (true))
```

```
//... a && b... rende true perché sia a che b sono diverse
```

```
// da 0
```

Operatori: OR

- A **OR** B è una funzione logica che è vera solo se almeno uno dei due operandi A o B è vero

<i>A</i>	<i>B</i>	<i>A OR B</i>
<i>F</i>	<i>F</i>	<i>F</i>
<i>F</i>	<i>V</i>	<i>V</i>
<i>V</i>	<i>F</i>	<i>V</i>
<i>V</i>	<i>V</i>	<i>V</i>

- In C, L'operatore OR riceve in ingresso due valori (A e B) e interpreta ciascuno di essi come:
 - Falso, se 0
 - Vero, se diverso da 0
- In C, L'operatore OR rende:
 - 0, se l'espressione valutata (si veda tabella) è falsa
 - diverso da 0, se l'espressione valutata (tabella) è vera

Operatori: OR

- In C, $A \text{ OR } B$, si scrive $A \parallel B$

Ovvero: $\langle \text{espressione1} \rangle \parallel \langle \text{espressione2} \rangle$

<i>A</i>	<i>B</i>	<i>A OR B</i>
<i>F</i>	<i>F</i>	<i>F</i>
<i>F</i>	<i>V</i>	<i>V</i>
<i>V</i>	<i>F</i>	<i>V</i>
<i>V</i>	<i>V</i>	<i>V</i>

- In C, $A \parallel B$ rende:
 - il valore booleano **true (diverso da 0)** se almeno uno dei due operandi è **true (diverso da 0)**
 - Restituisce il valore booleano **false (0)** se entrambi gli operandi sono **false (0)**

Operatori: OR - Esempi

```
int a, b, c;
```

```
a=6;
```

```
b=a+1; //assegnazione b ha valore 7
```

```
c=7;
```

```
//... ((c==b) || (a<b) )... rende true... ((true) && (true))
```

```
//... ((c!=b) || (a<b) )... rende true... ((false) && (true))
```

```
//... ((c==b) || (c>b)) ... rende true... ((true) && (false))
```

```
//... ((c==b) || (c<b)) ... rende true... ((true) && (false))
```

```
//... ((a<=b) || (c==b)) ... rende true... ((true) && (true))
```

```
//... (a>b) || (c<b) ... rende false perché nessuna delle
```

```
//due espressioni valutate è vera
```

```
//... a || b... rende true perché sia a che b sono diverse
```

```
// da 0
```

Concetto di side-effects



Assegnazione e side-effects

- L'assegnazione (=) è un operatore
 - Esempi:
 - `a=2;`
 - `name='Fabio';`
- L'assegnazione del risultato di una espressione ad una variabile è a sua volta una espressione
 - Esempi:
 - `area = lato * lato;`
 - `result = a+(b*c);`
- Il calcolo di una espressione restituisce un valore (output) e produce un effetto sui dati (side-effects)

Operatori Aritmetici e side effects, Esempi

- $a = b + (c=125);$
 - assegna ad a il valore di b aumentato del valore restituito dalla espressione $c=125$ (side effect)
 - restituisce il valore assegnato ad a (output)
 - INOLTRE:
 - l'espressione $c=125$ a sua volta:
 - assegna il valore 125 a c (side effect)
 - restituisce il valore 125 (output)
- $a = b + 125;$
 - assegna ad a il valore di b aumentato di 125 (side effect)
 - restituisce il valore che è assegnato ad a

Operatori Relazionali, Esempio

- $a > b$;
 - restituisce 1 SE l'espressione è vera
 - restituisce 0 SE l'espressione è falsa
- Questo comportamento vale per:
 - Operatori relazionali
 - Operatori logici

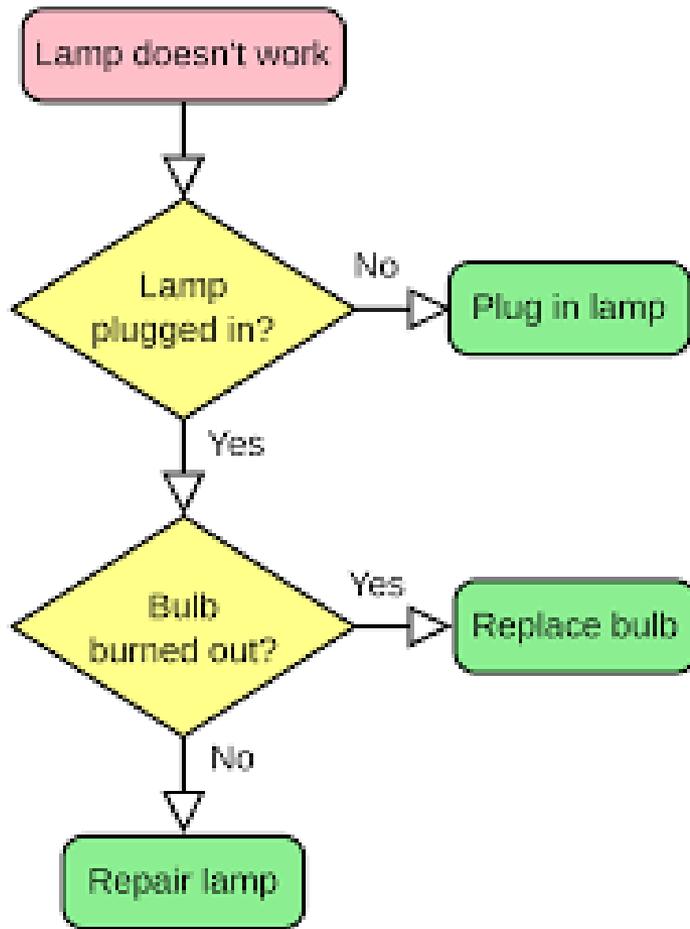


Outline

- **Rappresentazione**
 - Un frammento del linguaggio C
 - Tipi, variabili e costanti
 - Operatori ed espressioni
 - Istruzioni ←
 - Rappresentazione dei dati:
 - Numeri
 - Interi senza segno
 - Caratteri
 - Interi con segno

Uso dei Flow Chart

Flow Chart



- I Flow Chart servono per schematizzare flussi di ragionamento nei vari ambiti
- Servono come strumento per formalizzare/visualizzare gli algoritmi
- Gli algoritmi per poter essere 'processabili' da un elaboratore devono essere tradotti in un programma
- I passi finiti degli algoritmi sono tradotti in Istruzioni in un programma

Concetto di Istruzione

- ‘In informatica, elemento della programmazione con cui si richiede al computer, attraverso un codice prestabilito, l’esecuzione di una determinata operazione (leggere dati in ingresso, effettuare un calcolo, una selezione, ecc.)’, Treccani
- ‘Con il termine istruzione, in informatica, si intende il comando impartito ad un esecutore (processore) utilizzando un linguaggio ad esso comprensibile’, wikipedia

Blocco di istruzioni (Compound)

- ‘Un sottoinsieme auto consistente di istruzioni adiacenti la cui esecuzione complessiva equivale all'esecuzione di una singola macroistruzione complessa viene denominato **blocco di istruzioni.**’ Wikipedia
- Un **blocco di istruzioni** può essere considerato l'equivalente di una istruzione
 - Ogni istruzione può essere scomposta in istruzioni più semplici finché non si arriva ad una istruzione che non può essere ulteriormente scomposta (approccio top-down).
 - Ogni istruzione può essere raggruppata all'interno di un'istruzione più complessa (compound), fino ad arrivare al livello dell'intero programma (approccio bottom-up).

ISTRUZIONI e Flow Chart (1)

- Le ISTRUZIONI servono per '**dirigere**' il flusso della esecuzione di un programma:
 - In C i dati sono elaborati attraverso i side-effects prodotti dalle espressioni
 - Il ruolo delle istruzioni è quello di determinare la sequenza con cui le istruzioni devono essere eseguite
- Uno strumento fondamentale per descrivere le istruzioni è il flow chart (diagramma di flusso)
- FLOW CHART: sono diagrammi che servono per evidenziare il flusso delle istruzioni



ISTRUZIONI e Flow Chart (2)

- La più semplice **istruzione** è una espressione seguita da un punto e virgola:

- $a = b + 125;$

- Altri esempi:

int a; //dichiarazione

int b; //dichiarazione

int c,d; //dichiarazione

//NOTA: Si ricorda che l'assegnazione è a sua volta una //espressione

a=6; //assegnazione: a ha valore 6

b=a+1; //assegnazione: b ha valore 7

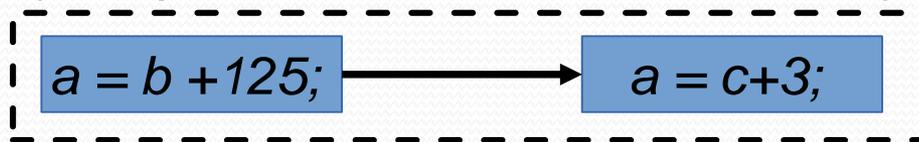
a=++b; //assegnazione: a vale 8, b vale 8

Espressioni e Flow charts (2)

- Nel rappresentare i flow chart si fa uso di:
 - **Rettangoli:** entro cui si inseriscono le espressioni
 $a = b + 125;$
 - **Linee continue e frecce:** per collegare le varie componenti



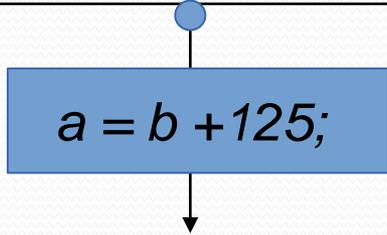
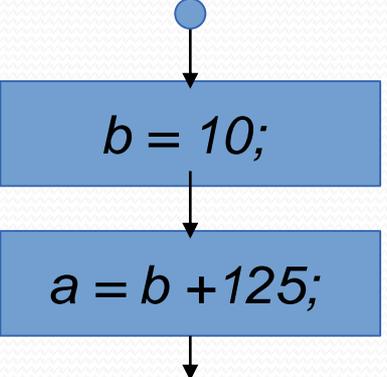
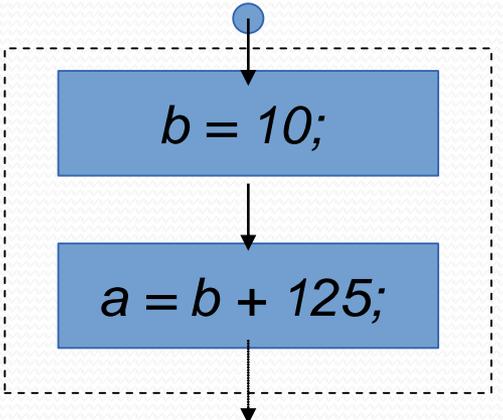
- **Linee tratteggiate:** per effettuare raggruppamenti (compound o blocchi di istruzioni)



- **Rombi:** per esprimere le condizioni (che vedremo)



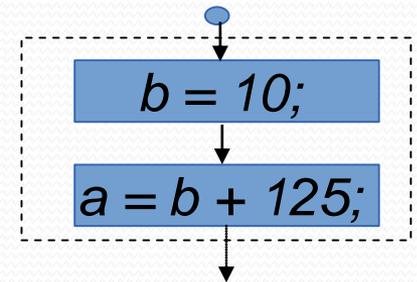
Espressioni e Flow charts (2)

Linguaggio C	Flow chart	NOTE
CASO A) a= b+125;		Istruzione semplice
CASO B) b =10; a = b + 125;		Sequenza di 2 istruzioni
CASO C) { b =10; a = b + 125; }		Istruzione Compound

Concetto di compound

- L'esecuzione del compound consiste nella esecuzione della istruzione racchiusa tra parentesi graffe (caso C visto in precedenza) e che è costituita da una sequenza di istruzioni

```
{  
  b = 10;  
  a = b + 125;  
}
```



- L'utilità del compound sarà più chiara successivamente
- Il vantaggio è quello di raccogliere una serie di istruzioni in blocchi in modo che possano essere trattati in maniera unitaria
- **Le parentesi condizionano la SINTASSI vincolando l'ordine di associazione dei termini**

Condizione – istruzione if (1)

- Le istruzioni condizionali permettono di decidere direzioni diverse nel flusso di esecuzione in base al valore restituito da una espressione
- Istruzione **if**
 - Condiziona l'esecuzione di una istruzione detta **CORPO**, al risultato restituito da una espressione detta **GUARDIA**



Condizione – istruzione if (1)

- Le istruzioni condizionali permettono di decidere direzioni diverse nel flusso di esecuzione in base al valore restituito da una espressione
- Istruzione if
 - Condiziona l'esecuzione di una istruzione detta CORPO, al risultato restituito da una espressione detta GUARDIA

Sintassi:

```
if(GUARDIA) {  
    CORPO  
}
```

- Esempio:

```
if(a>b){           // (a>b) → guardia  
    b =10;         // istruzioni che si trovano dentro le  
    a = b + 125;  // graffe → corpo
```

```
}
```

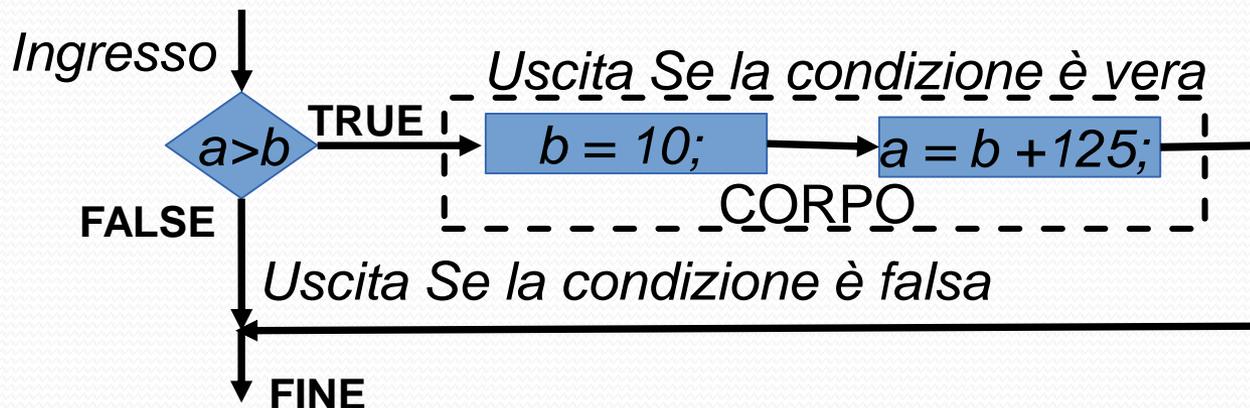


Condizione – istruzione if (1)

```
if(GUARDIA) {  
    CORPO  
}
```

- Esempio:

```
if(a>b){           // (a>b) → guardia  
    b = 10;        // istruzioni che si trovano dentro le  
    a = b + 125;  // graffe → corpo  
}
```



Condizione – istruzione if (2)

- Esempio:

```
if(a>b){           // (a>b) → guardia
    b =10;         // istruzioni che si trovano dentro le
    a = b + 125;  // graffe → corpo
}
```

- Sequenza di azioni:

- Si valuta l'espressione (a>b), ovvero la guardia in cui compare l'operatore Aritmetico 'maggiore di'
- SE la guardia è vera (true), ovvero SE la guardia restituisce un valore diverso da 0, allora viene eseguito il corpo
- SE la guardia è falsa (false), ovvero SE la guardia restituisce un valore uguale a 0, allora il corpo NON viene valutato dall'elaboratore

Condizione - istruzione if e Flow Chart

Linguaggio C	Flow chart	Note
<pre>if(a>b){ b =10; a = b + 125; }</pre>	<pre>graph TD Start(()) --> Cond{a > b} Cond -- TRUE --> P1[b = 10;] P1 --> P2[a = b + 125;] P2 --> Join(()) Cond -- FALSE --> Join Join --> End(())</pre>	Presenza di compound
<pre>if(a>b) b =10; a = b + 125;</pre>	<pre>graph TD Start(()) --> Cond{a > b} Cond -- TRUE --> P1[b = 10;] P1 --> P2[a = b + 125;] P2 --> Join(()) Cond -- FALSE --> Join Join --> End(())</pre>	

Esempi istruzione if e operatore incremento (1)

```
...
int a, b;
a = 6;
...
b = a++; //b ha valore 6, a ha valore 7
if (a>b) {                               // (a>b) → guardia
    b = 10;                               /* in questo caso è VERO che a>b,
                                           quindi l'elaboratore esegue il corpo*/
    a = b + 125;
    printf("SE a è maggiore di b, Stampo il valore di a %d e
           quello di b %d", a, b);
}
...
printf("DA qui PASSO SEMPRE!");
...
```

Esempi istruzione if e operatore incremento (2)

```
...
int a, b;
a = 6;
b = ++a; //a ha valore 7, b ha valore 7
if (a>b) {                               // (a>b) → guardia
    b = 10;                               /* in questo caso NON è VERO che a>b,
                                           quindi l'elaboratore NON esegue il corpo */
    a = b + 125;
    printf("SE a è maggiore di b, Stampo il valore di a %d e
           quello di b %d", a, b);
}
...
printf("DA qui PASSO SEMPRE! IN questo caso passo SOLO da
qui");
...
```

Indentazione (1)

- E' importante scrivere programmi:
 - Sintatticamente e semanticamente validi: **interpretabili dal compilatore** (aspetto che sarà approfondito)
 - **Leggibili**: è necessario fare in modo che il codice che si sta scrivendo sia comprensibile da chi legge il programma (noi stessi, un nostro collega, etc.)
 - Se si scrive un codice con una struttura chiara sarà più semplice risolvere/trovare eventuali errori logici che si stanno facendo
- Una delle regole di base per scrivere un programma leggibile è *l'indentazione* affiancata dall'uso dei *commenti*

Indentazione (2)

- **L'indentazione** consiste nell'inserire spazi o tabulazioni (che solitamente vengono ignorati dal compilatore) per mettere in luce eventuali gerarchie dei cicli o delle funzioni
- Ad esempio, torniamo alla nostra condizione if:

```
if(a>b){                               // (a>b) → guardia
    b =10;                               /* le istruzioni che si trovano
                                         dentro le graffe costituiscono il corpo */
    a = b + 125;   //graffe → corpo
}
```

- Notiamo subito che:
 - Sintassi: if(guardia){corpo}
 - Comprensibilità del codice:
 - Le tabulazioni e gli 'a capo', sottolineano quali sono le espressioni che fanno parte del corpo
 - I commenti ci aiutano a capire la logica di funzionamento

Indentazione (3)

- Esempio:

```
int a; //dichiarazione di a
```

```
int b; //dichiarazione di b
```

```
a = 5; //assegnazione di a
```

```
b = a+1; // assegnazione di b
```

```
//Algoritmo: «Se  $a > b$  dividi a per 2 e somma b al risultato»
```

```
if(a>b){//indentazione corretta
```

```
    a = a/2;
```

```
    a = a+b;
```

```
}
```

```
if(a>b){a=a/2; a = a+b;} //indentazione poco chiara
```

Indentazione (4)

- Esempio:

```
int a; //dichiarazione di a
int b; //dichiarazione di b
a = 5; //assegnazione di a
b = a+1; // assegnazione di b
```

```
if(a>b)           // (a>b) → guardia
    b =10;        /* l'elaboratore esegue questa riga, se a è maggiore
                   di b. In questo caso, date le dichiarazioni e
                   assegnazioni fatte, il valore di b NON viene
                   modificato. b resta uguale a 6. */
a=4;              // cambio invece il valore di a
```

- In questo caso l'indentazione (ovvero la tabulazione davanti a 'b=10;') permette di sottolineare:
 - la dipendenza della nuova assegnazione di b dal valore della guardia, ovvero dalla condizione 'if'

Commenti

- Sintassi:
 - I commenti in C, come abbiamo visto, possono essere:
 - Su una sola riga: //
 - Su più righe: /*commento su più righe di codice*/

```
[...]  
if(a>b){//valuto la guardia  
    a=3; /*entro dentro la parentesi graffa ed eseguo le espressioni  
        al suo interno solo se a>b */  
    b=5;  
}
```

- I monitor spesso hanno una larghezza/risoluzione limitata
- Quando si scrive un programma è bene non superare le 80/90 colonne per riga

Condizione – istruzione if else (1)

- L'istruzione **if**, può essere estesa tramite l'uso di **if else** per avere due corpi alternativi che vengono valutati o meno in base al valore della guardia
- Esempio:

```
if (a>b)           //guardia
  a = b + 125;    //corpo1
else {            //corpo 2 alternativo, dentro le graffe
  b = 10;
  a = b + 125;
}
```

Condizione – istruzione if else (2)

Linguaggio C	Flow chart
<pre>... if (a>b) a = b + 125; //corpo 1 else { //corpo 2 dentro graffe b = 10; a = b + 125; }</pre>	<pre>graph TD Start(()) --> Cond{a > b} Cond -- TRUE --> Corp1[a = b + 125;] Corp1 --> Join(()) Cond -- FALSE --> Corp2Box subgraph Corp2Box [corpo 2] Corp2a[b = 10;] Corp2b[a = b + 125;] Corp2a --> Corp2b end Corp2b --> Join Join --> Exit(())</pre>

Istruzione if else – esempi (1)

```
...
int a;
int b;
a = 6;
b = a++; //operatore incremento postfisso, b=6, a=7
printf("Stampo \n il valore di a %d e quello di b %d PRIMA
dell'if\n", a, b);
if (a>b) { // (a>b) → guardia
    b = 10; //graffe → corpo 1
    a = b + 125;
    printf("\nPasso dal corpo 1!\n Stampo i NUOVI valori di a
%d e b %d\n", a, b);
}
else //la printf(...) è il corpo 2
    printf("\nPasso dal corpo 2 e stampo i VECCHI valori di a
%d e di b %d \n", a, b);
```

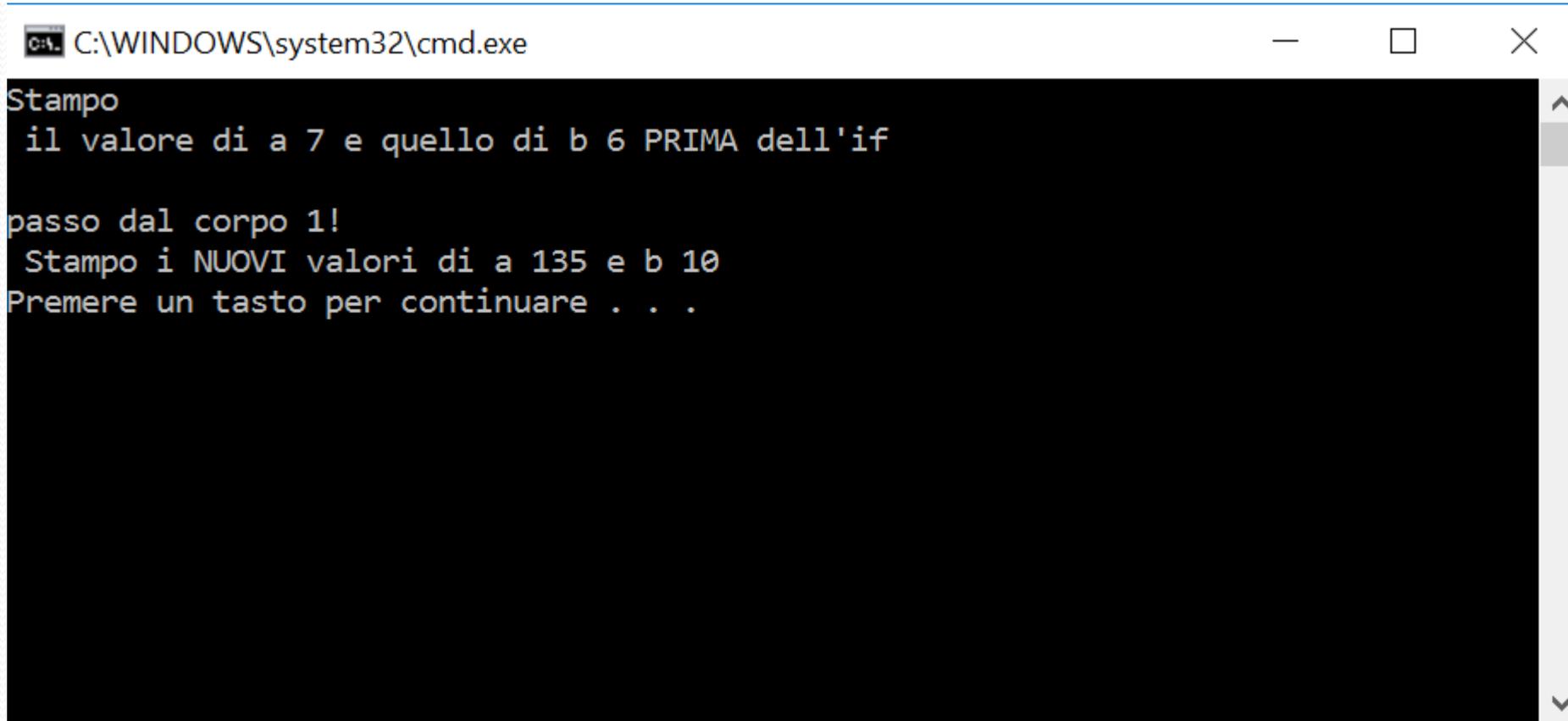
RISULTATO **????**

...



Istruzione if else – esempi (2)

RISULTATO: nella finestra di esecuzione del programma...



The screenshot shows a Windows command prompt window titled "C:\WINDOWS\system32\cmd.exe". The output text is as follows:

```
Stampo  
il valore di a 7 e quello di b 6 PRIMA dell'if  
  
passo dal corpo 1!  
Stampo i NUOVI valori di a 135 e b 10  
Premere un tasto per continuare . . .
```

Istruzione if else – esempi (3)

```
...
int a;
int b;
a = 6;
= ++a; //operatore incremento prefisso, b=7, a ha valore 7
printf("Stampo \n il valore di a %d e quello di b %d PRIMA
dell'if\n", a, b);
if (a>b) { // (a>b) → guardia
    b = 10; //graffe → corpo 1
    a = b + 125;
    printf("\nPasso dal corpo 1!\n Stampo i NUOVI valori di a
%d e b %d\n", a, b);
}
else //la printf(...) è il corpo 2
    printf("\nPasso dal corpo 2 e stampo i VECCHI valori di a
%d e di b %d \n", a, b);
```

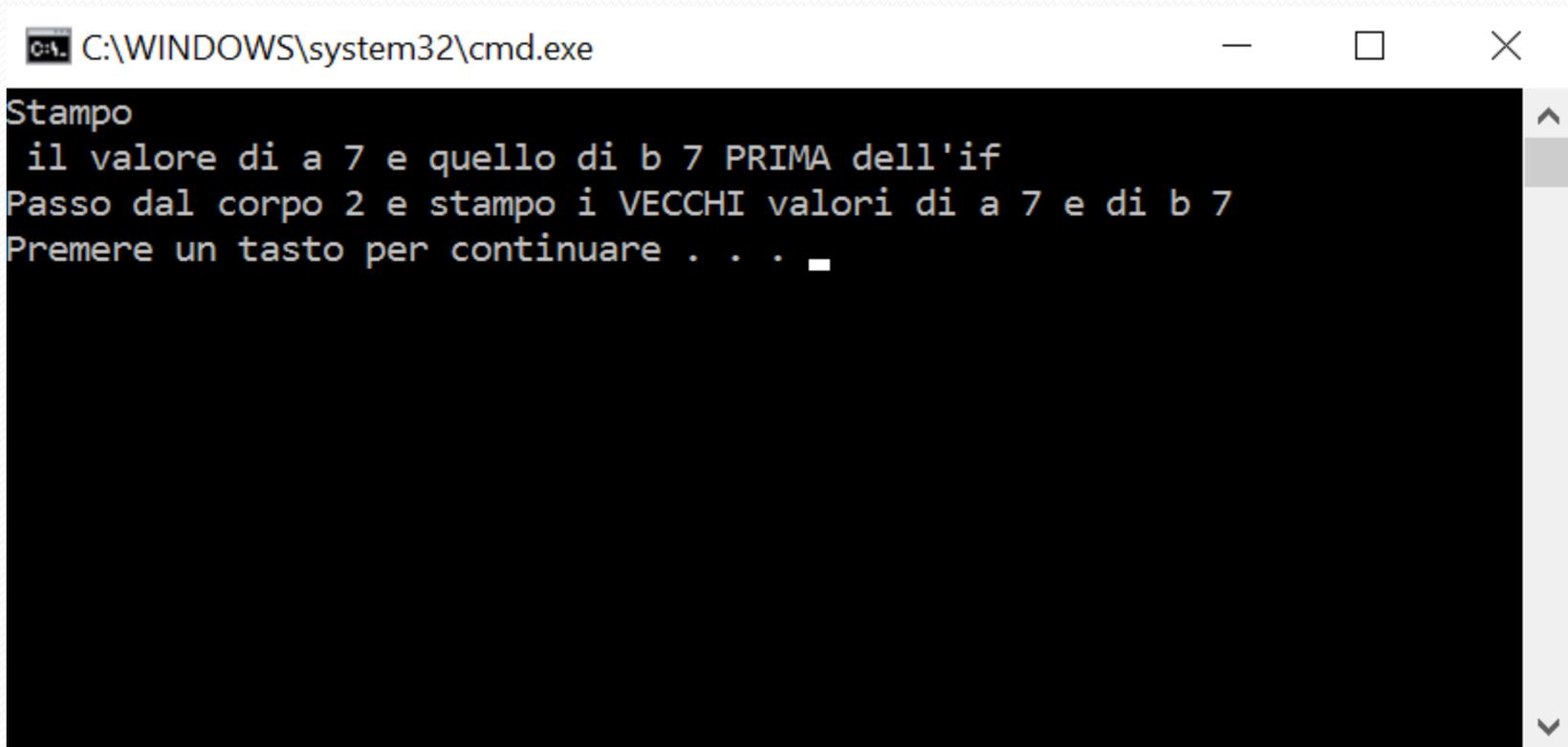
RISULTATO **?????**

...



Istruzione if else – esempi (4)

RISULTATO: nella finestra di esecuzione del programma...



A screenshot of a Windows command prompt window. The title bar shows the path 'C:\WINDOWS\system32\cmd.exe'. The window contains the following text:

```
Stampo
 il valore di a 7 e quello di b 7 PRIMA dell'if
 Passo dal corpo 2 e stampo i VECCHI valori di a 7 e di b 7
 Premere un tasto per continuare . . . █
```

Condizione – iterazione

- Le istruzioni di **iterazione** permettono di eseguire **ripetitivamente** un corpo di istruzioni finchè non si verifica una certa condizione sui valori delle variabili del programma
- Le istruzioni di iterazione sono le seguenti:
 - for
 - while
 - do while

• Sintassi: **Condizione – iterazione: for**

```
for(condizione_inizio_ciclo; GUARDIA; incremento){
    CORPO
}
```

• Esempio:

```
for (count = 0; count <10; count = count + 1) {
    printf("entro nel corpo");
    sum = sum + count;
}
```

	Quando viene valutata	Descrizione
condizione_inizio ciclo (es: count = 0;)	Prima di eseguire il ciclo (1 sola volta)	Usato per l'inizializzazione degli indici del ciclo
Guardia (es: count <10;)	Prima della esecuzione di OGNI iterazione (es: 10 volte...)	usato per verificare l'uscita dal ciclo for
Incremento (es: count = count +1;)	Incremento effettuato DOPO OGNI iterazione	Usato per incrementare gli indici del ciclo

Condizione – iterazione: for

Linguaggio C	Flow chart
<pre>int sum; int count; sum =0; for (count=0; count <10; count=count+1){ sum = sum + count; }</pre> <p>//Con: // count =0; inizializzazione – si valuta a //INIZIO ciclo // count <10; guardia // count = count + 1; incremento</p>	<pre>graph TD Start([Start]) --> Sum0[sum = 0;] Sum0 --> Count0[count = 0;] Count0 --> Cond{count < 10} Cond -- TRUE --> SumAdd[sum = sum + count;] SumAdd --> CountInc[count = count + 1;] CountInc --> Cond Cond -- FALSE --> End([End])</pre>

Ciclo for – esempi (1)

```
int sum;
int count;
sum = 0;

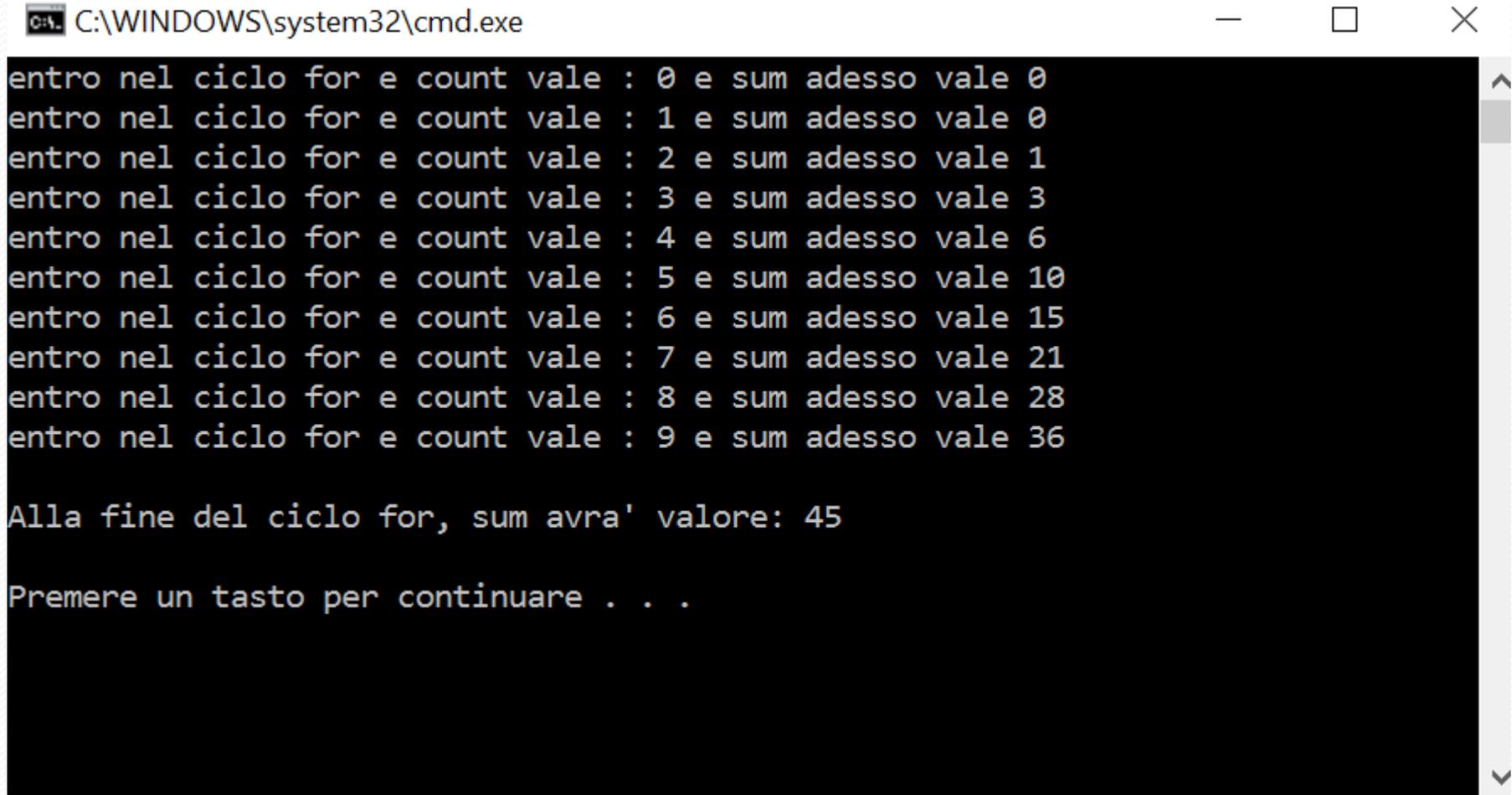
for (count = 0; count < 10; count = count + 1) {
    printf("entro nel ciclo for e count vale: %d e sum
           adesso vale %d\n", count, sum);
    sum = sum + count;
}

printf("\nAlla fine del ciclo for, sum avra' valore: %d\n\n",
sum);
```

RISULTATO ????

Ciclo for – esempi (2)

RISULTATO: nella finestra di esecuzione del programma...



```
C:\WINDOWS\system32\cmd.exe

entro nel ciclo for e count vale : 0 e sum adesso vale 0
entro nel ciclo for e count vale : 1 e sum adesso vale 0
entro nel ciclo for e count vale : 2 e sum adesso vale 1
entro nel ciclo for e count vale : 3 e sum adesso vale 3
entro nel ciclo for e count vale : 4 e sum adesso vale 6
entro nel ciclo for e count vale : 5 e sum adesso vale 10
entro nel ciclo for e count vale : 6 e sum adesso vale 15
entro nel ciclo for e count vale : 7 e sum adesso vale 21
entro nel ciclo for e count vale : 8 e sum adesso vale 28
entro nel ciclo for e count vale : 9 e sum adesso vale 36

Alla fine del ciclo for, sum avra' valore: 45

Premere un tasto per continuare . . .
```

Ciclo for – esempi (3)

```
int sum;
int count;
sum = 0;
//COSA SUCCEDE SE USO OPERATORE INCREMENTNO
POST FISSO?
for (count = 0; count <10; count++) {
    printf("entro nel ciclo for e count vale: %d e sum
        adesso vale %d\n", count, sum);
    sum = sum + count;
}

printf("\nAlla fine del ciclo for, sum avra' valore: %d\n\n",
sum);
```

RISULTATO **?????**

Ciclo for – esempi (3)

RISULTATO: nella finestra di esecuzione del programma...

```
C:\WINDOWS\system32\cmd.exe

entro nel ciclo for e count vale : 0 e sum adesso vale 0
entro nel ciclo for e count vale : 1 e sum adesso vale 0
entro nel ciclo for e count vale : 2 e sum adesso vale 1
entro nel ciclo for e count vale : 3 e sum adesso vale 3
entro nel ciclo for e count vale : 4 e sum adesso vale 6
entro nel ciclo for e count vale : 5 e sum adesso vale 10
entro nel ciclo for e count vale : 6 e sum adesso vale 15
entro nel ciclo for e count vale : 7 e sum adesso vale 21
entro nel ciclo for e count vale : 8 e sum adesso vale 28
entro nel ciclo for e count vale : 9 e sum adesso vale 36

Alla fine del ciclo for, sum avra' valore: 45

Premere un tasto per continuare . . .
```

*Nota: usando l'operatore post
fisso, il risultato NON
Cambia*

Ciclo for – esempi (4)

```
int sum;
int count;
sum = 0;
//COSA SUCCEDE SE USO OPERATORE INCREMENTO
PRE FISSO?
for (count = 0; count <10; ++count) {
    printf("entro nel ciclo for e count vale: %d e sum
        adesso vale %d\n", count, sum);
    sum = sum + count;
}

printf("\nAlla fine del ciclo for, sum avra' valore: %d\n\n",
sum);
```

RISULTATO **?????**

Ciclo for – esempi (2)

RISULTATO: nella finestra di esecuzione del programma...

```
C:\WINDOWS\system32\cmd.exe

entro nel ciclo for e count vale : 0 e sum adesso vale 0
entro nel ciclo for e count vale : 1 e sum adesso vale 0
entro nel ciclo for e count vale : 2 e sum adesso vale 1
entro nel ciclo for e count vale : 3 e sum adesso vale 3
entro nel ciclo for e count vale : 4 e sum adesso vale 6
entro nel ciclo for e count vale : 5 e sum adesso vale 10
entro nel ciclo for e count vale : 6 e sum adesso vale 15
entro nel ciclo for e count vale : 7 e sum adesso vale 21
entro nel ciclo for e count vale : 8 e sum adesso vale 28
entro nel ciclo for e count vale : 9 e sum adesso vale 36

Alla fine del ciclo for, sum avra' valore: 45

Premere un tasto per continuare . . .
```

*Nota: ANCHE usando
l'operatore incremento prefisso,
il risultato NON Cambia*

Condizione – iterazione: while

- Sintassi:

```
while (GUARDIA){  
CORPO  
}
```

- Esempio

```
while (count < 10){  
    sum = sum + count;  
    count = count +1;  
}
```

	Quando viene valutata	Descrizione
Guardia (es: count <10;)	Prima della esecuzione di OGNI iterazione (es: 10 volte...)	usato per verificare l'uscita dal ciclo for

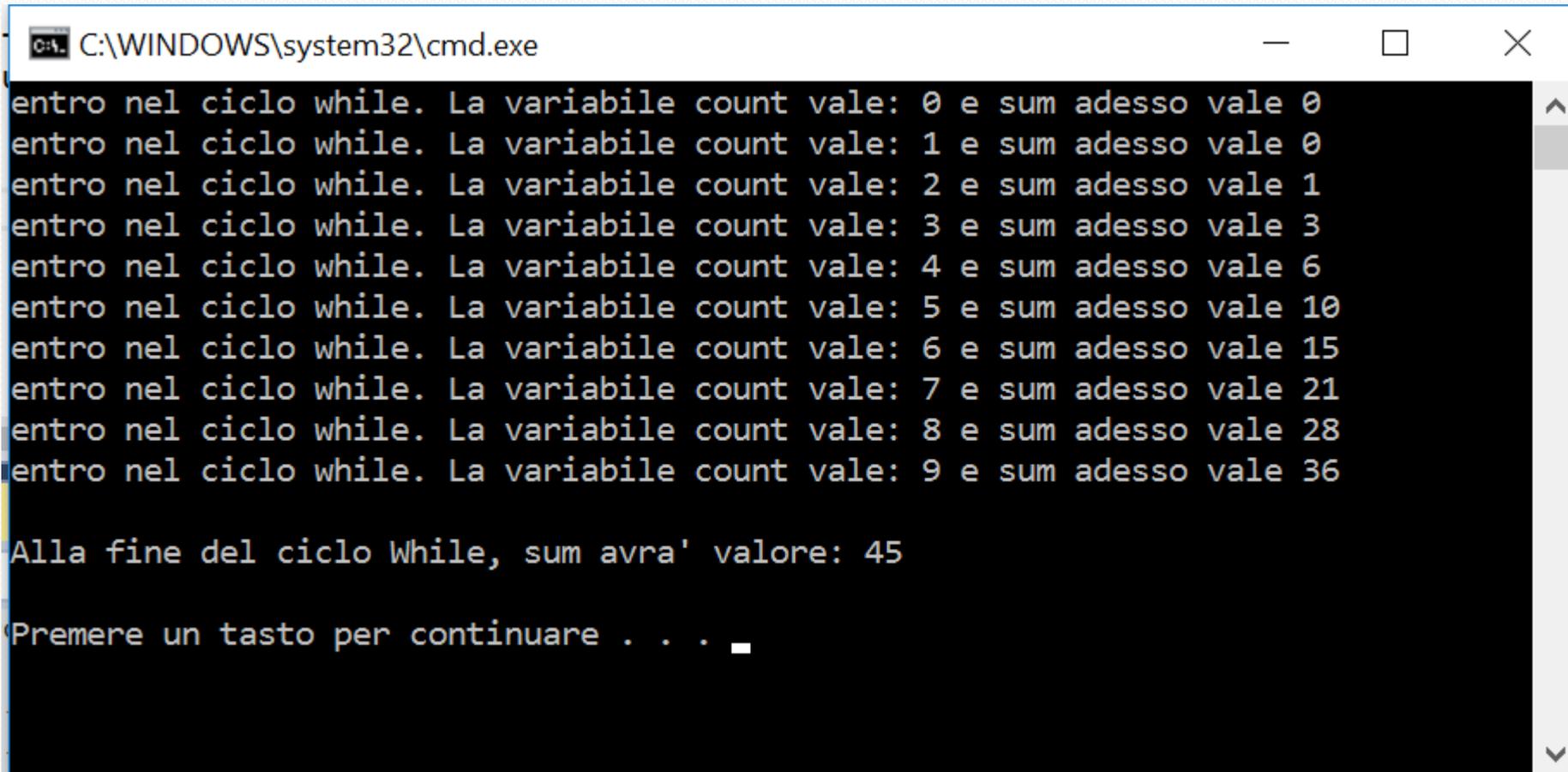
Condizione – iterazione: while

Linguaggio C	Flow chart
<pre>... count =0; sum =0; while (count < 10){ sum = sum + count; count = count +1; } // Con: // count =0; inizializzazione // count <10; guardia // count = count + 1; incremento</pre>	<pre>graph TD Start(()) --> C0[count = 0;] C0 --> S0[sum = 0;] S0 --> D{count < 10} D -- TRUE --> L1[sum = sum + count;] L1 --> L2[count = count + 1;] L2 --> D D -- FALSE --> Exit(())</pre>

Ciclo while – esempi (1)

```
int sum;
int count;
sum = 0;
count = 0;
sum = 0;
while (count < 10) {
    printf("entro nel ciclo while. La variabile count vale: %d e
           sum adesso vale %d\n", count, sum);
    sum = sum + count;
    count = count + 1;
}
printf("\nAlla fine del ciclo While, sum avra' valore: %d\n\n",
sum);
```

RISULTATO: nella finestra di esecuzione del programma...



```
C:\WINDOWS\system32\cmd.exe
entro nel ciclo while. La variabile count vale: 0 e sum adesso vale 0
entro nel ciclo while. La variabile count vale: 1 e sum adesso vale 0
entro nel ciclo while. La variabile count vale: 2 e sum adesso vale 1
entro nel ciclo while. La variabile count vale: 3 e sum adesso vale 3
entro nel ciclo while. La variabile count vale: 4 e sum adesso vale 6
entro nel ciclo while. La variabile count vale: 5 e sum adesso vale 10
entro nel ciclo while. La variabile count vale: 6 e sum adesso vale 15
entro nel ciclo while. La variabile count vale: 7 e sum adesso vale 21
entro nel ciclo while. La variabile count vale: 8 e sum adesso vale 28
entro nel ciclo while. La variabile count vale: 9 e sum adesso vale 36

Alla fine del ciclo While, sum avra' valore: 45

Premere un tasto per continuare . . .
```

Condizione – iterazione: do while

- Sintassi:

```
do { //entro nel ciclo sempre almeno una volta
    CORPO
}
```

while (GUARDIA)

- Esempio

```
do {
    sum = sum + count;
    count = count +1;
}
```

while (count < 10)

	Quando viene valutata	Descrizione
Guardia (es: count <10;)	Dopo l'esecuzione di OGNI iterazione (es: 10 volte...)	usato per verificare l'uscita dal ciclo for

Condizione – iterazione: do while

Linguaggio C	Flow chart
<pre>count = 0; sum = 0; do { //entro nel corpo almeno una volta sum = sum + count; count = count + 1; } while (count < 10); //NOTE: /* count = 0; inizializzazione count < 10; guardia count = count + 1; incremento */</pre>	<pre>graph TD Start([Start]) --> Init1[<i>count = 0;</i>] Init1 --> Init2[<i>sum = 0;</i>] subgraph LoopBody [] direction TB Body1[<i>sum = sum + count;</i>] Body2[<i>count = count + 1;</i>] end Init2 --> Body1 Body1 --> Body2 Body2 --> Decision{<i>count < 10</i>} Decision -- TRUE --> Body1 Decision -- FALSE --> Exit([Exit])</pre>

Ciclo do while – esempi (1)

```
int sum;
int count;
sum = 0;
count = 0;
sum = 0;

do {
    printf("entro nel ciclo do while. La variabile count vale:
    %d e sum adesso vale %d\n", count, sum);
    sum = sum + count;
    count = count + 1;
}
while (count < 10);
```

Ciclo do while – esempi (2)

```
C:\WINDOWS\system32\cmd.exe
entro nel ciclo do while. La variabile count vale: 0 e sum adesso vale 0
entro nel ciclo do while. La variabile count vale: 1 e sum adesso vale 0
entro nel ciclo do while. La variabile count vale: 2 e sum adesso vale 1
entro nel ciclo do while. La variabile count vale: 3 e sum adesso vale 3
entro nel ciclo do while. La variabile count vale: 4 e sum adesso vale 6
entro nel ciclo do while. La variabile count vale: 5 e sum adesso vale 10
entro nel ciclo do while. La variabile count vale: 6 e sum adesso vale 15
entro nel ciclo do while. La variabile count vale: 7 e sum adesso vale 21
entro nel ciclo do while. La variabile count vale: 8 e sum adesso vale 28
entro nel ciclo do while. La variabile count vale: 9 e sum adesso vale 36

Alla fine del ciclo do while, sum avra' valore: 45

Premere un tasto per continuare . . .
```

Esempio: calcolo del fattoriale (1)

- $N! = N*(N-1)*(N-2)...*3*2*1$
- Implementazione grazie all'uso del ciclo for

```
main(){  
    int n, fact=1;  
    int N = 10;  
    for(n=2; n<=N; n=n+1)  
        fact = fact *n; //alla fine si avrà N!  
}
```

- SINTASSI:
 - `main()` → è seguito dalle graffe {...} entro cui si trova tutto il programma
 - In seguito si vedrà l'uso delle **funzioni** per scomporre il programma in sotto-programmi

Esempio: calcolo del fattoriale (2)

```
main(){//inizio corpo della funzione main()
    int n, fact=1;
    int N = 10;
    for(n=2; n<=N; n=n+1)
        fact = fact *n; //alla fine si avrà N!
    printf("Fattoriale di %d = %d", N, fact);
} /*fine della funzione main()*/
```

- SINTASSI:
 - In questo caso il programma è il corpo della funzione main() e contiene un insieme di variabili e istruzioni
 - INDENTAZIONE: serve per leggere meglio il programma
 - COMMENTI: //una riga /*una o più righe*/

Esempio: calcolo del fattoriale di un qualsiasi

Numero N e restituzione del valore calcolato (1)

```
#define N 10
#include <stdio.h>
//10*(10-1)*(10-2)*(10-3)*(10-4)*...*(10-8)*(10-9)
// 10 * 9 * 8 * 7 * 6 * ... * 2 * 1
void main(void){
    int n, fact;
    fact = 1;//inizializzazione
    for (n = 2; n <= N; n=n+1) {
        fact = fact *n;
        printf("entro nel ciclo for e n vale : %d e fact
            adesso vale %d\n", n, fact);
    }
    printf("Risultato finale:  %d! = %d \n", N, fact);
}
```

Esempio: calcolo del fattoriale di un qualsiasi Numero N e restituzione del valore calcolato (2)

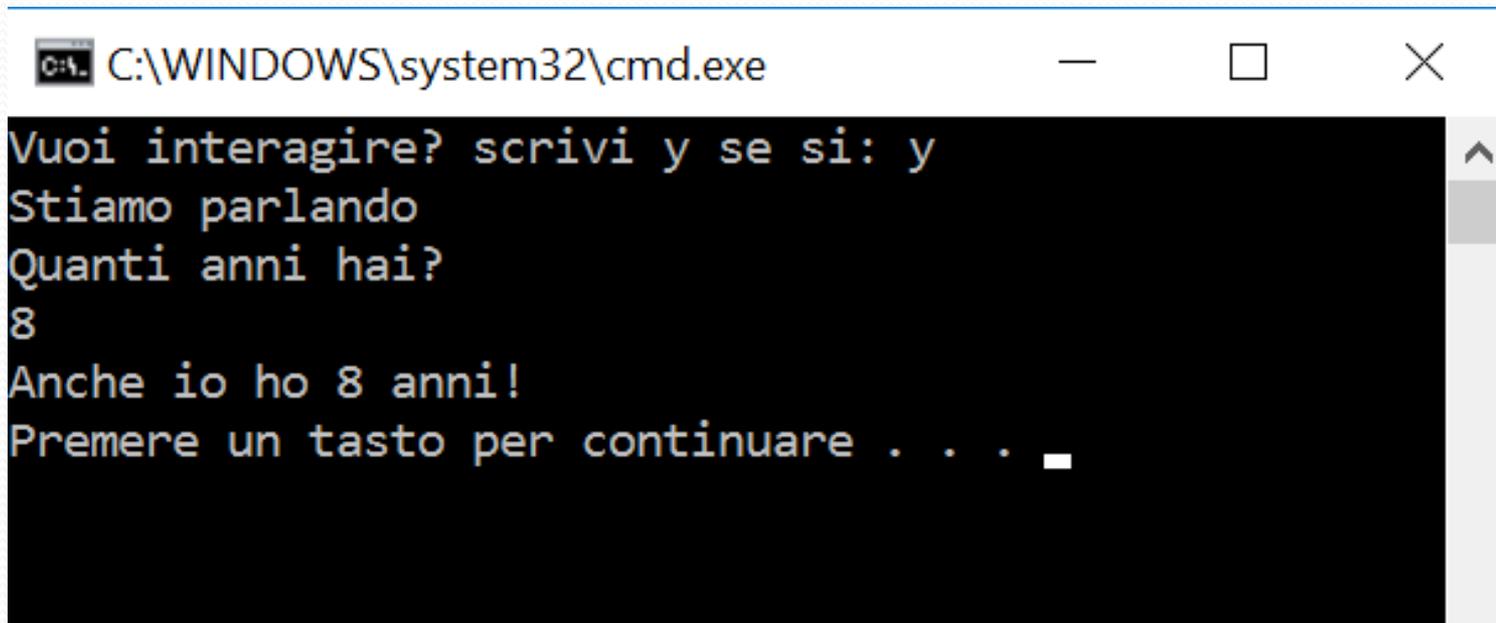
```
C:\WINDOWS\system32\cmd.exe
entro nel ciclo for e n vale : 2 e fact adesso vale 2
entro nel ciclo for e n vale : 3 e fact adesso vale 6
entro nel ciclo for e n vale : 4 e fact adesso vale 24
entro nel ciclo for e n vale : 5 e fact adesso vale 120
entro nel ciclo for e n vale : 6 e fact adesso vale 720
entro nel ciclo for e n vale : 7 e fact adesso vale 5040
entro nel ciclo for e n vale : 8 e fact adesso vale 40320
entro nel ciclo for e n vale : 9 e fact adesso vale 362880
entro nel ciclo for e n vale : 10 e fact adesso vale 3628800
Risultato finale: 10! = 3628800
Premere un tasto per continuare . . .
```

Esempio: Interazione da console (1)

```
#include <stdio.h>
main() {
    char c;
    int anni= 0;
    printf("Vuoi interagire? scrivi y se si: ");
    scanf("%c", &c); // &c da comprendere meglio in futuro
    if (c == 'y') { //operatore di uguaglianza
        printf("Stiamo parlando\nQuanti anni hai?\n");
        scanf("%d", &anni);
        if(anni
            printf("Anche io ho %d anni!\n", anni); //corpo1
        else //corpo2:
            printf("La tua eta' non mi e' chiara!\n
                La variabile anni, vale ancora %d\n", anni);
    }
    else
        printf("ADDIO!\n");
}
```

Esempio: Interazione da console (2)

- Alla domanda 'Quanti anni hai', l'utente immette un intero (ciò che ci si aspetta)
- Il programma tramite la scanf associa correttamente l'intero 8 alla variabile anni (scanf("%d", &anni);)
- Il programma infatti stampa

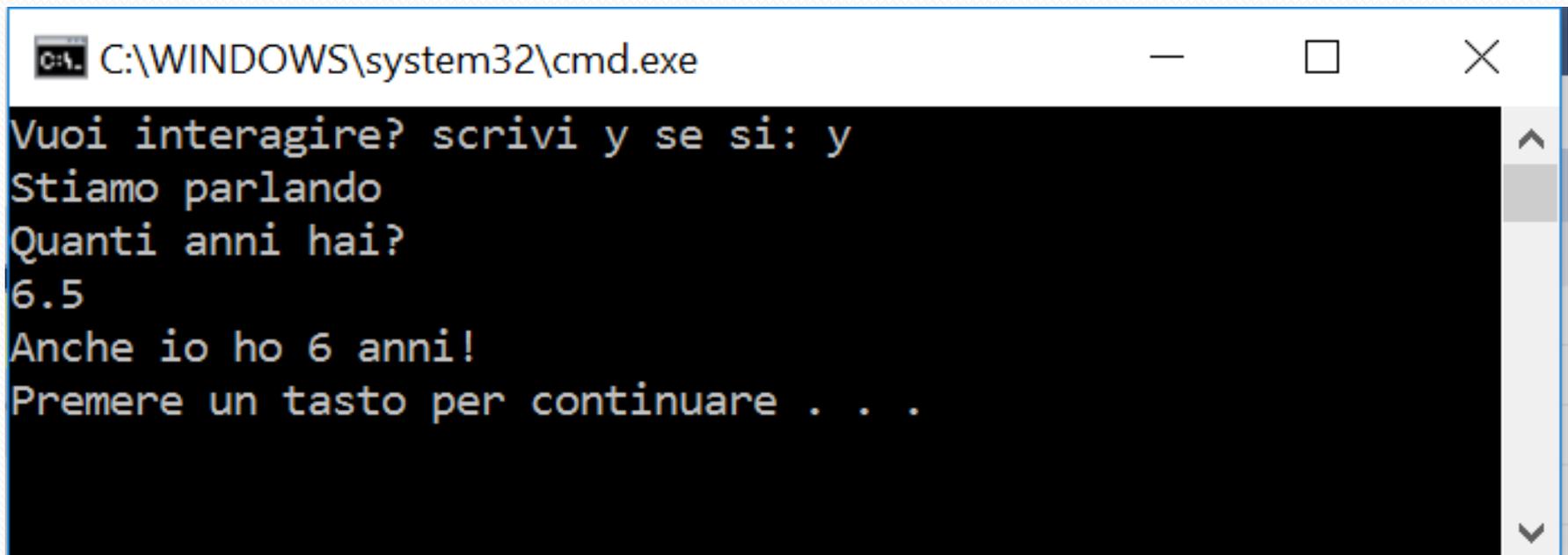


```
C:\WINDOWS\system32\cmd.exe

Vuoi interagire? scrivi y se si: y
Stiamo parlando
Quanti anni hai?
8
Anche io ho 8 anni!
Premere un tasto per continuare . . .
```

Esempio: Interazione da console (3)

- Alla domanda 'Quanti anni hai', l'utente immette un float (risposta che NON ci aspettiamo...)
- Il compilatore 'cerca' un intero, trova un float (immesso dall'utente) e fa una 'conversione' associa quindi alla variabile anni il valore intero 6 (anziché il float 6.5 immesso dall'utente)
- Si approfondirà in futuro...

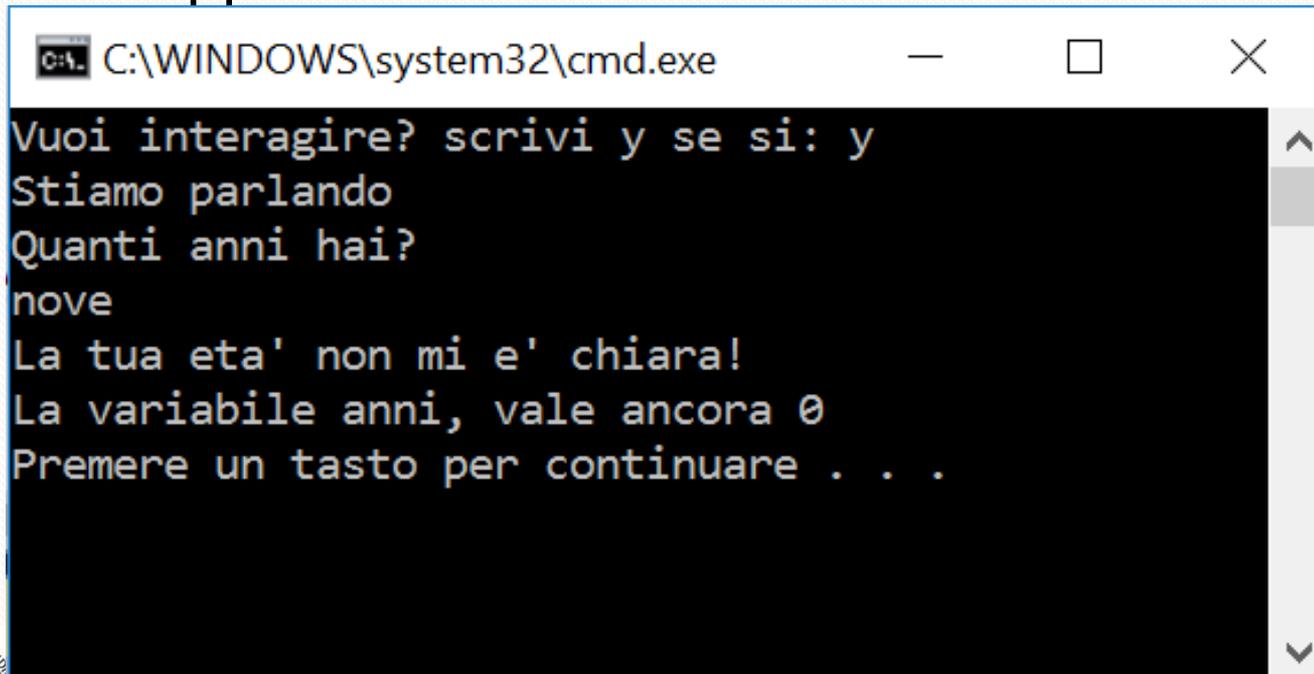


```
C:\WINDOWS\system32\cmd.exe

Vuoi interagire? scrivi y se si: y
Stiamo parlando
Quanti anni hai?
6.5
Anche io ho 6 anni!
Premere un tasto per continuare . . .
```

Esempio: Interazione da console (4)

- Alla domanda 'Quanti anni hai', l'utente immette una stringa (risposta che NON ci aspettiamo...)
- Il compilatore 'cerca' un intero, trova un char, non riesce a fare una conversione, quindi il valore della variabile anni resta quello iniziale (anni = 0;) -> la guardia è false (...if (anni)...) e viene eseguito il corpo 2
- Si approfondirà in futuro...



```
C:\WINDOWS\system32\cmd.exe

Vuoi interagire? scrivi y se si: y
Stiamo parlando
Quanti anni hai?
nove
La tua eta' non mi e' chiara!
La variabile anni, vale ancora 0
Premere un tasto per continuare . . .
```

Esercizi sugli Array usando i cicli



Somma dei primi N numeri Pari (1)

```
#include <stdio.h>
#define N 10
main() {
    int A[N];
    int i, sum=0;
    for (i = 0; i < N; i++) { //algoritmo di inserimento nell'array
        A[i] = (i+1)*2;
        printf("Ho inserito il valore A[%d]: %d\n", i, A[i]);
    }
    i=0; //assegno di nuovo il valore 0 a i perché?
    while (i<N) { //algoritmo calcolo somma elementi
        sum = sum + A[i];
        printf("somma parziale: %d\n", sum);
        i = i + 1;
    }
    printf("Somma: %d", sum); //NOTA: si poteva usare 1 solo ciclo...
    //vediamo....
}
```

Somma dei primi N numeri Pari (2)

```
#include <stdio.h>
#define N 10
//svolgimento esercizio precedente usando un solo ciclo....

main() {
    int A[N];
    int i, sum=0;
    for (i = 0; i < N; i++) {
        A[i] = (i+1)*2;
        printf("Ho inserito il valore A[%d]: %d\n", i, A[i]);
        sum = sum + A[i];
    }
    printf("Somma: %d", sum);
}
```

Esempio: calcolare la somma degli elementi memorizzati in un array (1)

- Uso del ciclo **while**

```
main(){
    float A[100];
    int count;
    float sum;
    ... //inserimento dei valori nell'array,
/*fare come esercizio, ad esempio inserire i primi 100
numeri pari */
    sum=0;
    count=0;
    while(count<100){
        sum=sum+A[count];
        count = count+1;
    }
```

Esempio: calcolare la somma degli elementi memorizzati in un array (2)

- Uso del ciclo **do while**

```
main(){
    float A[100];
    int count, sum;
    ...//inserimento dei valori nell'array
    sum=0;
    count=0;
    do{
        sum=sum+A[count];
        count = count+1;
    } while(count<100);
    printf ("somma: %d", sum);
}
```

Le stringhe (1)

- Una stringa consiste in una sequenza di caratteri.
- In C le stringhe NON sono definite come tipi (come avviene in altri linguaggi di programmazione)
- In C le stringhe vengono implementate grazie all'uso degli array
- In particolare, una stringa in C è costituita da un array di caratteri che termina con il carattere nullo
- Quando si dichiara una stringa occorre specificarne la dimensione, che sarà pari al numero di caratteri che deve contenere aumentato di una unità per il carattere nullo in fondo alla stringa.

Le stringhe (2)

- Quando si dichiara una stringa occorre specificarne la dimensione, che sarà pari al numero di caratteri che deve contenere aumentato di una unità per il carattere nullo in fondo alla stringa.

- Esempio:

```
char nome[12];
```

```
nome[12] = "Mario Rossi"; //11 caratteri
```

```
//alternativa alla assegnazione:
```

```
strcpy(nome, "Mario Rossi"); //libreria string.h
```

/*NOTA il carattere nullo in fondo alla stringa viene aggiunto automaticamente dal compilatore*/

Alcune funzioni per manipolare le stringhe

- La libreria `string.h` offre una serie di **funzioni** per manipolare le stringhe
- Eccone alcune:

Nome	Funzione
<code>strcpy(s1,s2)</code>	Copia <code>s2</code> in <code>s1</code>
<code>strcat(s1,s2)</code>	Concatena <code>s2</code> alla fine di <code>s1</code>
<code>strlen(s1)</code>	Restituisce la lunghezza di <code>s1</code>
<code>strcmp(s1,s2)</code>	Restituisce 0 se <code>s1</code> e <code>s2</code> sono uguali; un valore minore di 0 se <code>s1<s2</code> ; maggiore di 0 se <code>s1>s2</code>

Stampa e lettura di stringhe da console (1)

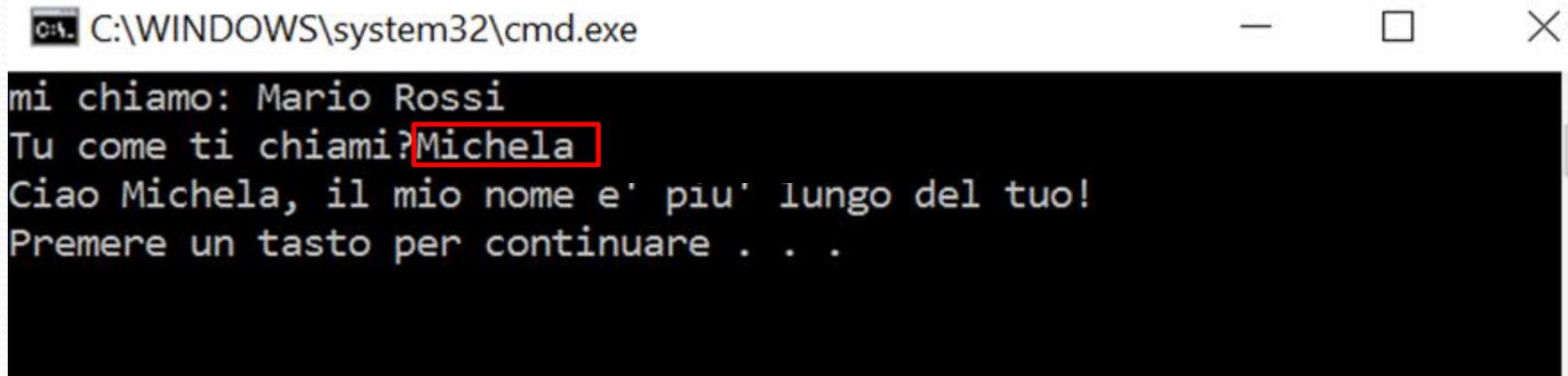
```
#include <stdio.h>
#include <string.h>
void main(void) {
    char nome_prog[12];
    char nome_utente[100]; //si suppone un max di 100 caratteri
    strcpy(nome_prog, "Mario Rossi");
    //nome_prog[12] = "Mario Rossi"; //equivalente alla riga precedente
    printf("mi chiamo: %s\nTu come ti chiami?", nome_prog);
    scanf("%s", nome_utente); //nota: per gli array NON ci vuole l'&....
    int u = strlen(nome_utente);
    int p = strlen(nome_prog);
    if (u > p)
        printf("Ciao %s, il tuo nome e' piu' lungo del mio!\n", nome_utente);
    else
        printf("Ciao %s, il mio nome e' piu' lungo del tuo!\n",
            nome_utente);
}
```

NOTA: //scanf legge la stringa fino a che non trova un delimitatore (lo spazio)

Stampa e lettura di stringhe da console (2)

```
#include <stdio.h>
#include <string.h>
void main(void) {
    char nome_prog[12];
    char nome_utente[100]; //si suppone un max di 100 caratteri
    strcpy(nome_prog, "Mario Rossi");
    //nome_prog[12] = "Mario Rossi"; //equivalente alla riga precedente
    printf("mi chiamo: %s\nTu come ti chiami?", nome_prog);
    scanf("%s", nome_utente); //nota: per gli array NON ci vuole l'&....
    int u = strlen(nome_utente);
    int p = strlen(nome_prog);
    if (u > p)
        printf("Ciao %s, il tuo nome e' piu' lungo del mio!\n", nome_utente);
    else
        printf("Ciao %s, il mio nome e' piu' lungo del tuo!\n",
            nome_utente);
}
```

NOTA: //scanf legge la stringa fino a che non trova un delimitatore (lo spazio)



```
C:\WINDOWS\system32\cmd.exe
mi chiamo: Mario Rossi
Tu come ti chiami?Michela
Ciao Michela, il mio nome e' piu' lungo del tuo!
Premere un tasto per continuare . . .
```

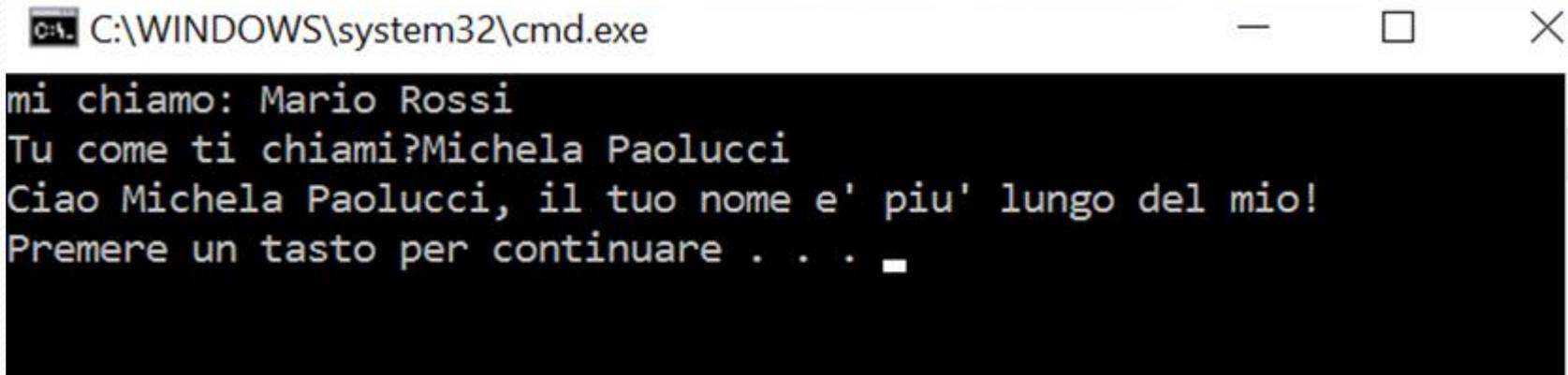
Stampa e lettura di stringhe da console (3)

```
#include <stdio.h>
#include <string.h>
void main(void) {
    char nome_prog[12];
    char nome_utente[100]; //si suppone un max di 100 caratteri
    //nome[12] = "Mario Rossi";
    strcpy(nome_prog, "Mario Rossi");
    printf("mi chiamo: %s\nTu come ti chiami?", nome_prog);
    gets(nome_utente); //legge ciò che l'utente immette spazi compresi
    int u = strlen(nome_utente);
    int p = strlen(nome_prog);
    if (u > p)
        printf("Ciao %s, il tuo nome e' piu' lungo del mio!\n",
            nome_utente);
    else
        printf("Ciao %s, il mio nome e' piu' lungo del tuo!\n",
            nome_utente);
}
```

Stampa e lettura di stringhe da console (4)

```
#include <stdio.h>
#include <string.h>
void main(void) {
    char nome_prog[12];
    char nome_utente[100]; //si suppone un max di 100 caratteri
    strcpy(nome_prog, "Mario Rossi");
    //nome_prog[12] = "Mario Rossi"; //equivalente alla riga precedente
    printf("mi chiamo: %s\nTu come ti chiami?", nome_prog);
    scanf("%s", nome_utente); //nota: per gli array NON ci vuole l'&....
    int u = strlen(nome_utente);
    int p = strlen(nome_prog);
    if (u > p)
        printf("Ciao %s, il tuo nome e' piu' lungo del mio!\n", nome_utente);
    else
        printf("Ciao %s, il mio nome e' piu' lungo del tuo!\n",
            nome_utente);
}
```

NOTA: //scanf legge la stringa fino a che non trova un delimitatore (lo spazio)



```
C:\WINDOWS\system32\cmd.exe
mi chiamo: Mario Rossi
Tu come ti chiami?Michela Paolucci
Ciao Michela Paolucci, il tuo nome e' piu' lungo del mio!
Premere un tasto per continuare . . .
```

Compilatori...



Compilatore online: Coliru

← → ↻ | coliru.stacked-crooked.com | 📖 ☆ | ☰ 📄 🗑️ ...

Donate \$ Coliru Restore defaults Help Feedback
Editor Command Q&A Read Write

```
1 #include <iostream>
2 #include <string>
3 #include <vector>
4
5 template<typename T>
6 std::ostream& operator<<(std::ostream& os, const std::vector<T>& vec)
7 {
8     for (auto& el : vec)
9     {
10         os << el << ' ';
11     }
12     return os;
13 }
14
15 int main()
16 {
17     std::vector<std::string> words = {
18         "Hello", "from", "GCC", __VERSION__, "!"
19     };
20     std::cout << words << std::endl;
21 }
22
```

[Home page](#)

<http://coliru.stacked-crooked.com>

```
g++ -std=c++14 -O2 -Wall -pedantic -pthread
main.cpp && ./a.out
```

Compile, link and run...

Share!

Compilatore online (2): Coliru

Donate

Coliru

Restore defaults

Help

Feedback



Editor

Command

Q&A

Read Write

```
1 #include <stdio.h>
2
3 int main()
4 {
5     int sum =0, count=0;
6     do {
7         printf("La variabile count vale: %d \n", count);
8         sum += count;
9         printf("Incremento la somma: %d \n", sum);
10        count = count + 1;
11        printf("Incremento count: %d \n\n", count);
12    }while (count < 4);
13    printf("Alla fine del ciclo do while, sum avra' valore: %d\n\n", sum);
14 }
15
```

Inclusione libreria

**Ciclo do{...} while()
dalla riga 6
alla riga 12**

**main(){...}
dalla riga 3
alla riga 14**

```
g++ -std=c++14 -O2 -Wall -pedantic -pthread main.cpp && ./a.out
```

Compilazione ed esecuzione del programma

Compile, link and run...

Share!

Donate

Coliru

Restore defaults Help Feedback Editor Command Q&A Read Write

Esempi di compilazione (1)

```

1 #include <stdio.h>
2
3 int main()
4 {
5     int sum =0, count=0;
6     do {
7         printf("La variabile count vale: %d \n", count);
8         sum += count;
9         printf("Incremento la somma: %d \n", sum);
10        count = count + 1;
11        printf("Incremento count: %d \n\n", count);
12    }
13    while (count < 4);
14    printf("Alla fine del ciclo do while, sum avra' valore: %d\n\n", sum);
15 }
16

```

```

La variabile count vale: 0
Incremento la somma: 0
Incremento count: 1

La variabile count vale: 1
Incremento la somma: 1
Incremento count: 2

La variabile count vale: 2
Incremento la somma: 3
Incremento count: 3

La variabile count vale: 3
Incremento la somma: 6
Incremento count: 4

Alla fine del ciclo do while, sum avra' valore: 6

```

Output / scrittura su console

```
g++ -std=c++14 -O2 -Wall -pedantic -pthread main.cpp && ./a.out
```

Compile, link and run...

Share!

Esempi di compilazione (2)

The screenshot shows a web browser window with the URL `coliru.stacked-crooked.com`. The page title is "Coliru" and it includes navigation links like "Restore defaults", "Help", "Feedback", "Editor", "Command", "Q&A", "Read", and "Write".

```
1 #include <stdio.h>
2
3 int main()
4 {
5     int sum;
6     int count;
7     sum = 0;
8     count = 0;
9     sum = 0;
10 do {
11     printf("entro nel ciclo do while. La variabile count vale: %d e sum adesso vale %d\n",
12           count, sum);
13     sum = sum + count;
14     count = count + 1;
15 }
16 while (count < 10);
17
18 printf("\nAlla fine del ciclo do while, sum avra' valore: %d\n\n", sum);
19 }
20
```

At the bottom, the terminal shows the compilation command:

```
g++ -std=c++14 -O2 -Wall -pedantic -pthread main.cpp && ./a.out
```

A red arrow points to the "Compile, link and run..." button, which is highlighted with a red border. A "Share!" button is also visible to the right.

Esempi di compilazione (3)

coliru.stacked-crooked.com

Donato \$ Coliru Restore defaults Help Feedback Editor Command Q&A Read Write

```
1 #include <stdio.h>
2
3 int main()
4 {
5     int sum;
6     int count;
7     sum = 0;
8     count = 0;
9     sum = 0;
10    do {
11        printf("entro nel ciclo do while. La variabile count vale: %d e sum adesso vale %d\n",
12              count, sum);
13        sum = sum + count;
14        count = count + 1;
15    }
16    while (count < 10);
17
18    printf("\nAlla fine del ciclo do while, sum avra' valore: %d\n\n", sum);
19 }
```

```
entro nel ciclo do while. La variabile count vale: 0 e sum adesso vale 0
entro nel ciclo do while. La variabile count vale: 1 e sum adesso vale 0
entro nel ciclo do while. La variabile count vale: 2 e sum adesso vale 1
entro nel ciclo do while. La variabile count vale: 3 e sum adesso vale 3
entro nel ciclo do while. La variabile count vale: 4 e sum adesso vale 6
entro nel ciclo do while. La variabile count vale: 5 e sum adesso vale 10
entro nel ciclo do while. La variabile count vale: 6 e sum adesso vale 15
entro nel ciclo do while. La variabile count vale: 7 e sum adesso vale 21
entro nel ciclo do while. La variabile count vale: 8 e sum adesso vale 28
entro nel ciclo do while. La variabile count vale: 9 e sum adesso vale 36

Alla fine del ciclo do while, sum avra' valore: 45
```

g++ -std=c++14 -O2 -Wall -pedantic -pthread main.cpp && ./a.out

Compile, link and run... Share!



Esempi di compilazione (4)

Donate

C o l i r u

```
$  
1 include <stdio.h>  
2  
3 int main()  
4 {  
5     int sum =0, count=0;  
6     for(int i=0; i<4;i++) {  
7         printf("La variabile count vale: %d \n", count);  
8         sum += count;  
9         printf("Incremento la somma: %d \n", sum);  
10        count = count + 1;  
11        printf("Incremento count: %d \n\n", count);  
12    }  
13    printf("Alla fine del ciclo for, sum avra' valore: %d\n\n", sum);  
14 }  
15
```

Esempi di compilazione (5)

```

1  #include <stdio.h>
2
3  int main()
4  {
5      int sum =0, count=0;
6      for(int i=0; i<4;i++) {
7          printf("La variabile count vale: %d \n", count);
8          sum += count;
9          printf("Incremento la somma: %d \n", sum);
10         count = count + 1;
11         printf("Incremento count: %d \n\n", count);
12     }
13     printf("Alla fine del ciclo for, sum avra' valore: %d\n\n", sum);
14 }
15

```

```

La variabile count vale: 0
Incremento la somma: 0
Incremento count: 1

```

```

La variabile count vale: 1
Incremento la somma: 1
Incremento count: 2

```

```

La variabile count vale: 2
Incremento la somma: 3
Incremento count: 3

```

```

La variabile count vale: 3
Incremento la somma: 6
Incremento count: 4

```

```

Alla fine del ciclo for, sum avra' valore: 6

```

```

g++ -std=c++14 -O2 -Wall -pedantic -pthread main.cpp && ./a.out

```

Esempi di compilazione (6)

Donate

C o l i r u

\$

```
1 #include <stdio.h>
2
3 int main()
4 {
5     int sum =0, count=0;
6     for(i=0; i<4;i++) {
7         printf("La variabile count vale: %d \n", count);
8         sum += count;
9         printf("Incremento la somma: %d \n", sum);
10        count = count + 1;
11        printf("Incremento count: %d \n\n", count)
12    }
13    printf("Alla fine del ciclo for, sum avra' valore: %d\n\n", sum);
14 }
15
```

Cosa succede se sbaglio?

Che errori ci sono?

Esempi di compilazione (7)

Donate

C o l i r u

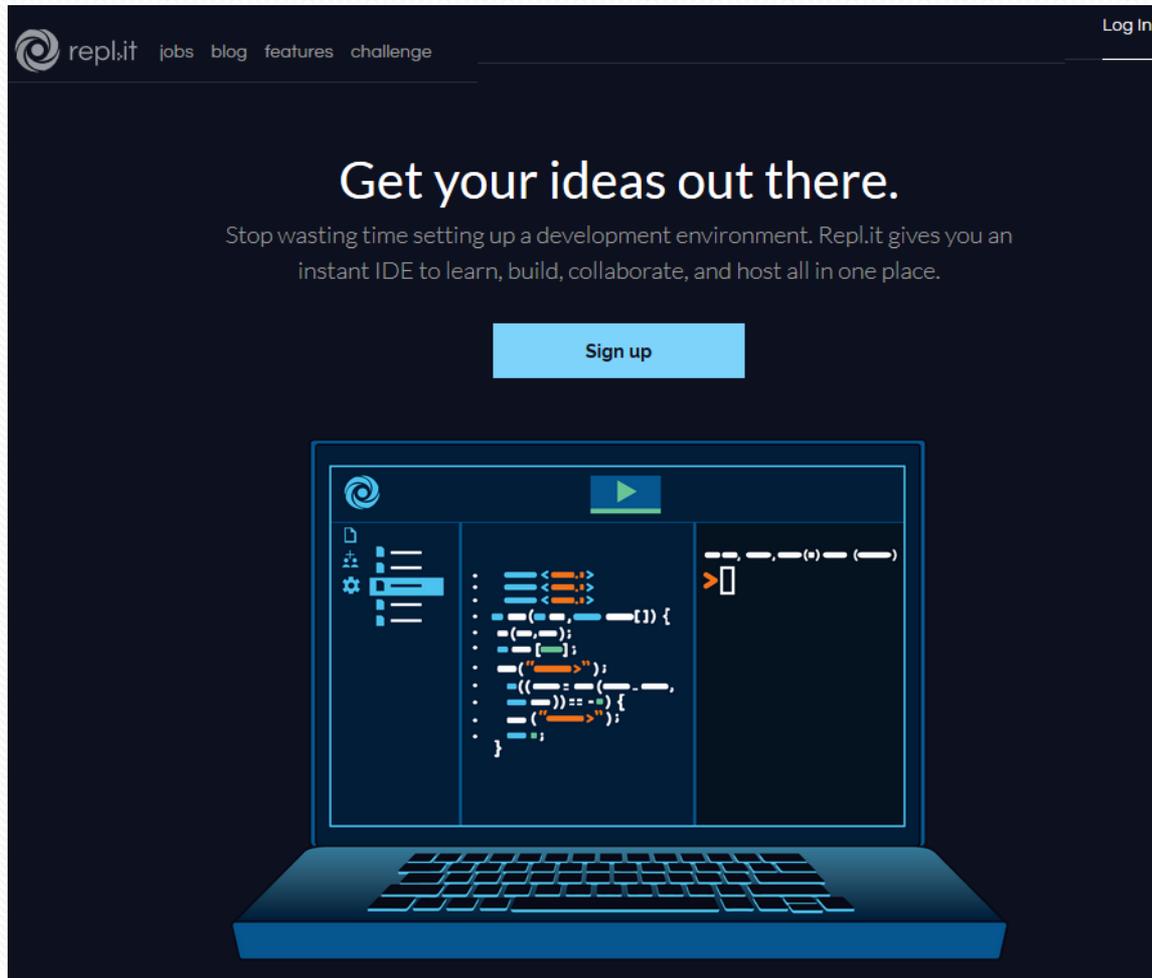
```
$
1  #include <stdio.h>
2
3  int main()
4  {
5      int sum =0, count=0;
6      for(i=0; i<4;i++) {
7          printf("La variabile count vale: %d \n", count);
8          sum += count;
9          printf("Incremento la somma: %d \n", sum);
10         count = count + 1;
11         printf("Incremento count: %d \n\n", count)
12     }
13     printf("Alla fine del ciclo for, sum avra' valore: %d\n\n", sum);
14 }
15
```

```
main.cpp: In function 'int main()':
main.cpp:6:6: error: 'i' was not declared in this scope
  for(i=0; i<4;i++) {
    ^
main.cpp:12:2: error: expected ';' before '}' token
  }
  ^
```

Ecco gli errori...

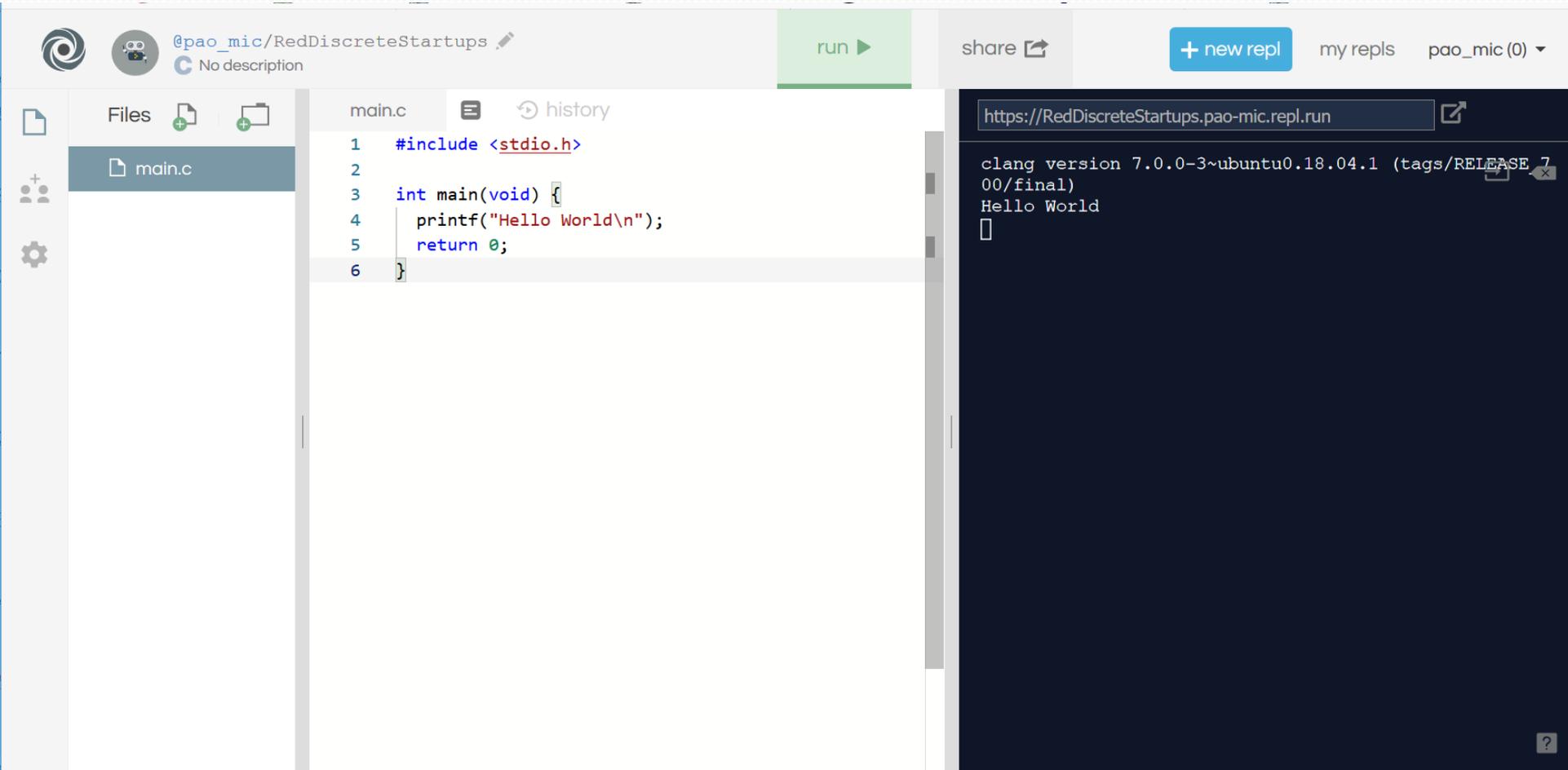
Compilatore online: Repl.it

- <https://repl.it/>



Compilatore online: Repl.it

- <https://repl.it/>



The screenshot displays the Repl.it online compiler interface. At the top, the user profile is '@pao_mic/RedDiscreteStartups' with a 'No description' note. A green 'run' button is visible. The main area is divided into three sections: a file explorer on the left showing 'main.c', a code editor in the center containing the following C code:

```
main.c history
1 #include <stdio.h>
2
3 int main(void) {
4     printf("Hello World\n");
5     return 0;
6 }
```

On the right, the terminal output shows the compilation and execution results:

```
https://RedDiscreteStartups.pao-mic.repl.run
clang version 7.0.0-3~ubuntu0.18.04.1 (tags/RELEASE_700/final)
Hello World
[]
```

Compilatore online: Repl.it

- <https://repl.it/>

The screenshot displays the Repl.it online compiler interface. At the top, there are navigation links and a search bar. The main area is divided into three sections: a file explorer on the left, a code editor in the center, and a terminal on the right.

File Explorer: Shows a folder named 'main.c'.

Code Editor: Contains the following C code:

```
1 #include <stdio.h>
2
3 #define N 10
4 //somma primi N numeri pari
5
6 main() {
7     int A[N];
8     int i, sum=0;
9     for (i = 0; i < N; i++) {
10         A[i] = (i+1)*2;
11         printf("Ho inserito il valore A[%d]: %d\n", i, A[i]);
12         sum = sum + A[i];
13     }
14     printf("Somma: %d", sum);
15 }
16
```

Terminal: Shows the compilation and execution output:

```
clang version 7.0.0-3~ubuntu0.18.04.1 (tags/RELEASE_700/final)
main.c:6:1: warning: type specifier missing, defaults to 'int' [-Wimplicit-int]
        main() {
        ^
1 warning generated.

Ho inserito il valore A[0]: 2
Ho inserito il valore A[1]: 4
Ho inserito il valore A[2]: 6
Ho inserito il valore A[3]: 8
Ho inserito il valore A[4]: 10
Ho inserito il valore A[5]: 12
Ho inserito il valore A[6]: 14
Ho inserito il valore A[7]: 16
Ho inserito il valore A[8]: 18
Ho inserito il valore A[9]: 20
Somma: 110
```

Visual Studio

- Home page: <https://www.visualstudio.com>
- Download: <https://www.visualstudio.com/it/downloads>

Microsoft Tecnologie Documentazione Risorse

Visual Studio IDE di Visual Studio Funzionalità Offerte Download Support

La tua sottoscrizione Visual Studio gratuito

Download di Visual Studio

Visual Studio 2017 Community
IDE gratuito con funzionalità complete per studenti e sviluppatori singoli e open-source.
[Download gratuito](#)

Visual Studio 2017 Professional
Strumenti di sviluppo professionali, servizi e vantaggi della sottoscrizione per i piccoli team.
[Versione di valutazione gratuita](#)

Visual Studio 2017 Enterprise
Soluzione end-to-end per soddisfare le complesse esigenze di qualità e scalabilità dei team di tutte le dimensioni.
[Versione di valutazione gratuita](#)

[Note sulla versione e documentazione](#) [Confronta le edizioni di Visual Studio](#) [Come eseguire l'installazione offline](#)

Cerca tutti i download:

Espandi tutto Comprimi tutto

- Visual Studio Code
- Visual Studio 2017

Informazioni licenze studenti

- Licenza da studente
 - https://sol.unifi.it/elms/stud_jsp/login.jsp

domenica 12 marzo 2017

ACCESSO A MSDN ACADEMIC ALLIANCE

La sottoscrizione dell'accordo Campus ha reso possibile l'adesione al programma **MSDN Academic Alliance**, un servizio che consente ai docenti e agli studenti delle scuole di Ingegneria e SMFN di utilizzare gratuitamente una singola installazione di vari software Microsoft. I software installati devono essere utilizzati per finalità didattiche; qualsiasi uso commerciale del software è vietato. Ogni abuso è responsabilità personale dell'utente.

NOTA MSDNAA è un servizio realizzato da Microsoft: SIAF eroga esclusivamente il sistema di autenticazione alla piattaforma per studenti e dipendenti delle Scuole di Ingegneria e SMFN. **Le richieste di assistenza o di abilitazione a ulteriori download del software non saranno prese in considerazione:** raccomandiamo pertanto di effettuare il download attraverso una connessione tramite cavo (non usare connessioni wireless) perché, nel caso in cui la linea si interrompa, non sarà possibile abilitare l'utente a effettuare nuovamente il download.

Per accedere alla piattaforma di distribuzione del software Microsoft, gli studenti devono autenticarsi nella maschera sottostante con le proprie **credenziali per l'autenticazione unica**.

Si ricorda che tutte le volte che si vorrà accedere al sito MSDN, il login dovrà essere effettuato da questa pagina.

LOGIN

Matricola:

Password:

Per segnalare problemi tecnici [scrivi allo staff](#)

- progetto e idea grafica SIAF - Servizi realizzati da SIAF-Ufficio Sistemi Informativi e Processi -

Download visual studio 2017

- <https://visualstudio.microsoft.com/it/downloads/>



Microsoft | Visual Studio IDE di Visual Studio Altro Visual Studio gratuito Tutti i siti Microsoft

Introduzione a Visual Studio

Personalizza il tuo IDE, impara i concetti fondamentali e inizia a creare la tua prima app in pochi minuti.

Scarica Visual Studio

- Community 2017
- Professional 2017
- Enterprise 2017

download



Microsoft | Visual Studio IDE di Visual Studio Altro Visual Studio gratuito Tutti i siti Microsoft

Introduzione a Visual Studio

Personalizza il tuo IDE, impara i concetti fondamentali e inizia a creare la tua prima app in pochi minuti.

Scarica Visual Studio

Consulta la documentazione >

documentazione



Setup & Installation
Learn how to install and set up Visual Studio.

Get Started with Visual Studio
Develop any app for any platform.

What's New in Visual Studio

Visual Studio Documentation

Get Started | Tasks | Languages | Workloads

Learn how to use Visual Studio

- Start a guided tour
- Write and edit code
- Build your code
- Debug your code
- Test your code
- Access data locally or in the cloud

Follow a tutorial

- C#
- F#
- Visual Basic
- C++**
- Python
- JavaScript

Create an app

- Universal Windows app
- Windows desktop app
- Mobile app
- Unity game
- Web app with ASP.NET Core

Microsoft | Docs | Windows | Microsoft Azure | Visual Studio | Office | More

Docs / Visual C++ / Documentation / C++ in Visual Studio

Visual Studio | C++

Get Started

- Install Visual Studio with C++
- Start a Guided Tour

C++ tutorials

- Create your first C++ app
- Create a Universal Windows App
- Create a Windows Desktop app
- Create a UWP game with DirectX

Learn Visual Studio

Learn how to use Visual Studio for your development tasks.

- Write and edit code
- Compile and build
- Debug your code

Download PDF

Alcune linee guida per impostare Visual Studio



Visual Studio: creare un progetto per il C (1)

- Visual Studio supporta il C++
- Visual Studio supporta in modo indiretto anche il C
- Per creare un nuovo progetto C in Visual Studio, è necessario seguire i seguenti passi:
 - Creare un progetto C++
 - Configurarlo per il linguaggio C

Visual Studio: creare un progetto per il C

Fase I
Creare un progetto C++



Visual Studio: creare un progetto per il C (2)

- Vista iniziale

Pagina iniziale - Microsoft Visual Studio

File Modifica Visualizza Debug Team Strumenti Test Analizza Finestra ?

Pagina iniziale

Visual Studio

Inizia

Nuovo progetto...

Apri progetto...

Apri dal controllo del codice sorgente...

Recenti

Esempi_base1

ConsoleApplication1

ConsoleApplication1

Informazioni su Visual Studio Community 2015

Esercitazioni sulla scrittura di codice e progetti di esempio appositamente pensati per i nuovi utenti di Visual Studio

Corsi sui nuovi framework, linguaggi e tecnologie

Creazione di un repository di codice privato e di un backlog per il progetto

Procedure semplificate per iniziare a usare i servizi cloud

Accesso ad altre tecniche per estendere e personalizzare l'IDE

È ora possibile abilitare l'esperienza per l'ambiente cloud.

Connessione ad Azure

Nuovo nelle piattaforme Microsoft

- Windows
- Microsoft Azure
- ASP.NET e Web

Notizie

Visual Studio 2017 – Now Ready for Your

Output

Mostra output di:

Elenco errori Output

Pronto

Visual Studio: creare un progetto per il C (3)

1) Nuovo progetto > Visual C++ > Progetto Console Win 32

The screenshot shows the Visual Studio interface with the 'Nuovo progetto' (New Project) dialog box open. The dialog is titled 'Nuovo progetto' and has a search bar 'Cerca in Modelli installati (Ctrl+E)'. The left pane shows a tree view of installed models, with 'Visual C++' selected. The main pane shows a list of project templates, with 'Progetto console Win32' highlighted. The right pane shows the selected template's details, including the type 'Visual C++' and a description. At the bottom, there are input fields for 'Nome' (ConsoleApplication2), 'Percorso' (C:\Users\disit\Documents\Visual Studio 2015\Projects\Test\), and 'Nome soluzione' (ConsoleApplication2). There are also checkboxes for 'Crea directory per soluzione' (checked) and 'Aggiungi al controllo del codice sorgente' (unchecked), and 'OK' and 'Annulla' buttons.

Visual Studio: creare un progetto per il C (3)

The screenshot shows the 'Nuovo progetto' (New Project) dialog in Visual Studio. The 'Modelli' (Templates) section is expanded to 'Visual C++', and the 'Progetto console Win32' (Win32 Console Application) template is selected. The 'Nome' (Name) field is set to 'PrimoProgramma', the 'Percorso' (Path) is 'C:\Users\disit\Documents\Visual Studio 2015\Projects\PrimoProgramma', and the 'Nome soluzione' (Solution Name) is 'PrimoProgramma'. The 'OK' button is highlighted.

1) Nuovo progetto...

2) Impostazione Nome Progetto

3) OK

Visual Studio: creare un progetto per il C (4)

Pagina iniziale - Microsoft Visual Studio

File Modifica Visualizza Debug Te

Avvio veloce (CTRL+Q)

Accedi

Creazione guidata applicazione Win32 - ConsoleApplication2

Creazione guidata applicazione Win32

Panoramica

Impostazioni applicazione

Impostazioni correnti del progetto:

- Applicazione console

Scegliere **Fine** in una finestra qualsiasi per accettare le impostazioni correnti.

Una volta creato il progetto, vedere il file readme.txt per informazioni sulle funzionalità del progetto e sui file generati.

4) Click su 'avanti'

< Indietro **Avanti >** Fine Annulla

Creazione del progetto 'ConsoleApplication2' in corso...

Visual Studio: creare un progetto per il C (5)

Pagina iniziale - Microsoft Visual Studio

File Modifica Visualizza Progetto Debug Team Strumenti Test Analisi Finestra ?

Creazione guidata applicazione Win32 - PrimoProgramma

Impostazioni applicazione

Panoramica

Impostazioni applicazione

Tipo di applicazione:

- Applicazione Windows
- Applicazione console
- DLL
- Libreria statica

Opzioni aggiuntive:

- Progetto vuoto
- Esporta simboli
- Intestazione precompilata
- Controlli Security Development Lifecycle (SDL)

Aggiungi file di intestazione comune per:

- ATL
- MFC

5a) Click su 'progetto vuoto'

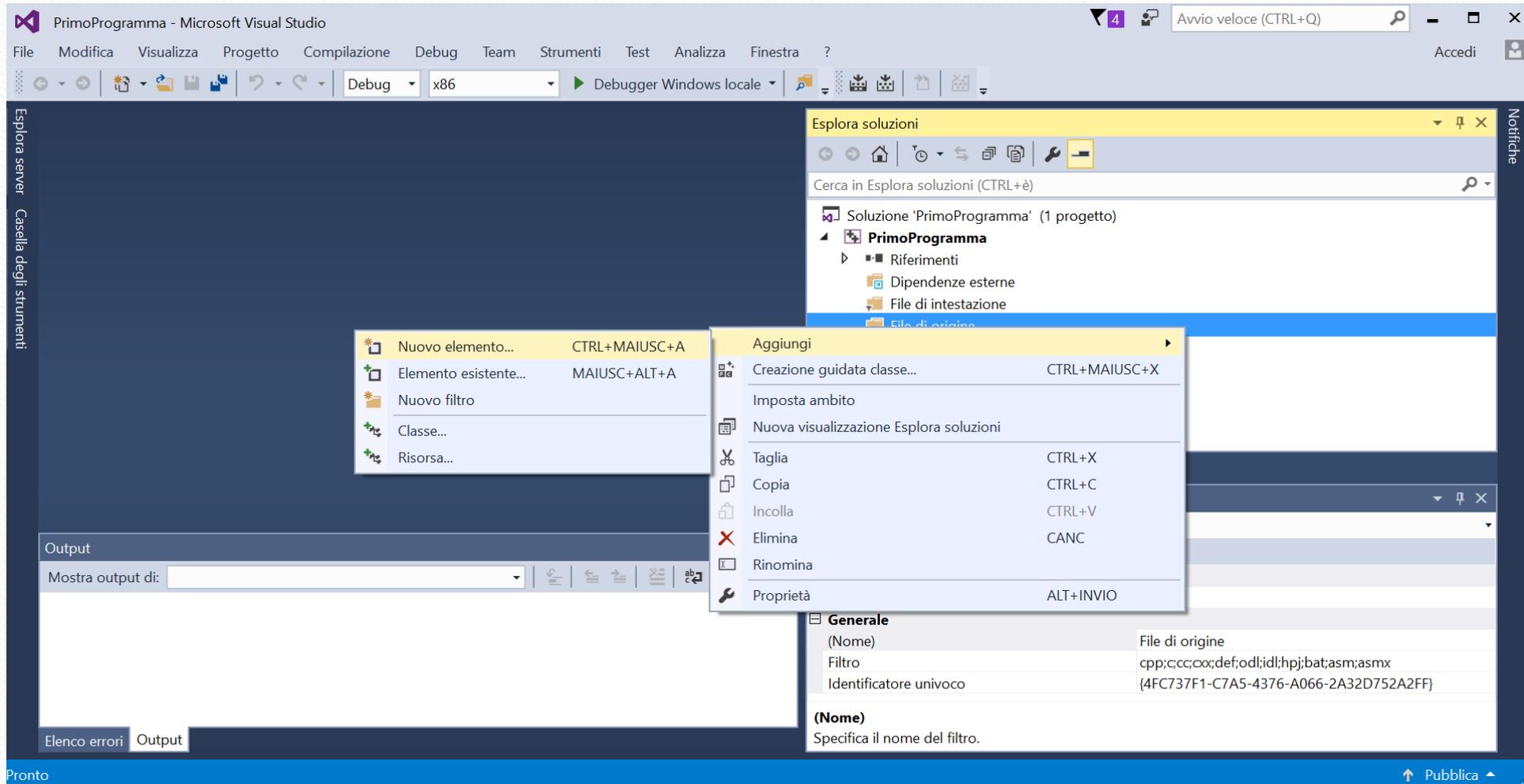
5b) Click su 'fine'

< Indietro Fine Annulla

Creazione del progetto 'PrimoProgramma' in corso...

Visual Studio: creare un progetto per il C (6)

6) Soluzione > nome_soluzione (PrimoProgramma) > File di origine >Aggiungi >Nuovo elemento



Visual Studio: creare un progetto per il C (7)

PrimoProgramma - Microsoft Visual Studio

File Modifica Visualizza Progetto Compilazione Debug Team Strumenti Test Analizza Finestra ?

Avvio veloce (CTRL+Q)

Aggiungi nuovo elemento - PrimoProgramma

Ordina per: Predefinita

Cerca in Modelli installati (Ctrl+E)

Icona	Nome	Visual C++	Descrizione
	File di C++ (.cpp)	Visual C++	File di C++
	File di intestazione (.h)	Visual C++	File di intestazione

7) Creare file C++

8) Salvare il file come C (estensione .c). Esempio nome_file: C:\...\Visual Studio 2015\Projects\PrimoProgramma\PrimoProgramma\Main.c

Fare clic qui per passare alla modalità online e ricercare modelli.

Nome:

Percorso:

Sfogli... Aggiungi Annulla

(Nome) Specifica il nome del filtro.

Visual Studio: creare un progetto per il C (3)

Fase II
Configurare il progetto C++
per il linguaggio C



Visual Studio: configurare progetto per il C (1)

1) Tasto destro sul nome del progetto > Pagine delle proprietà del programma > Proprietà di configurazione > C/C++ > Avanzate > Compila come : 'Compila come codice C (T/C)'

The screenshot shows the Visual Studio interface with the 'PrimoProgramma' project selected in the Solution Explorer. The 'Proprietà' (Properties) window is open, showing the 'C/C++' configuration page. The 'Avanzate' (Advanced) tab is selected, and the 'Compila come' (Compile as) property is set to 'Compila come codice C (/TC)'. The 'Convenzione di chiamata' (Calling convention) is set to 'cdecl (/Gd)'. The 'Compila come' property is highlighted with a red box, and the 'Avanzate' tab is also highlighted with a red box. The 'Convenzione di chiamata' property is also highlighted with a red box.

PrimoProgramma - Microsoft Visual Studio

File Modifica Visualizza Progetto Compilazione Debug Team Strumenti Test Analizza Finestra ?

Debug x86

Esplora soluzioni

Cerca in Esplora soluzioni (CTRL+E)

Soluzione 'PrimoProgramma' (1 progetto)

PrimoProgramma

- Riferimenti
- Dipendenze esterne
- File di intestazione
- File di origine
 - Main.c
- File di risorse

Esplora soluzioni Visualizzazione classi

Proprietà

PrimoProgramma Proprietà del progetto

Varie

- (Nome)
- Dipendenze progetto
- File di progetto
- Spazio dei nomi radice

(Nome)

Specifica il nome del progetto.

Proprietà di configurazione

- Generale
- Debug
- Directory di VC++
- C/C++
 - Generale
 - Ottimizzazione
 - Preprocessore
 - Generazione codice
 - Linguaggio
 - Intestazioni precompilate
 - File di output
 - Informazioni di visualizzazione
 - Avanzate
 - Tutte le opzioni
 - Riga di comando
 - Linker
 - Strumento Manifesto
 - Generatore di documenti
 - Informazioni di visualizzazione
 - Eventi di compilazione
 - Istruzione di compilazione
 - Analisi codice

Convenzione di chiamata: cdecl (/Gd)

Compila come: **Compila come codice C (/TC)**

Disabilita avvisi specifici

File di inclusione imposto

File #using imposto

Mostra inclusioni: No

Usa percorsi completi: No

Ometti nomi librerie predefinite: No

Segnalazione errori interni del compilatore: Richiedi immediatamente (/errorReport:prompt)

Considera avvisi specifici come errori

Compila come

Specifica il linguaggio di compilazione per i file c e cpp. (/TC, /TP)

OK Annulla

Questo elemento non supporta l'anteprima

↑ Pubblica

Visual Studio: configurare progetto per il C (2)

1) Tasto destro sul nome del progetto > Pagine delle proprietà del programma > Proprietà di configurazione > C/C++ > Avanzate > Riga di comando > Opzioni aggiuntive > mettere '/wd4996'

PrimoProgramma - Microsoft Visual Studio

File Modifica Visualizza Progetto Compilazione Debug Team Strumenti Test Analizza Finestra ? Accedi

Debug x86

Main.c

```
1 #include <stdio.h>
2
3 void main(void) {
4     int a;
5     float b;
6     a = 15;
7     b = a*3.3;
8     printf("il valore di a = %d, il valore
9         a, b);
10 }
```

PrimoProgramma (Ambito globale)

Pagine delle proprietà di PrimoProgramma

Configurazione: Tutte le configurazioni Piattaforma: Active(Win32) Gestione configurazione...

Proprietà di configurazione

- Generale
- Debug
- Directory di VC++
- C/C++
 - Generale
 - Ottimizzazione
 - Preprocessore
 - Generazione codice
 - Linguaggio
 - Intestazioni precompilate
 - File di output
 - Informazioni di visualizzazione
 - Avanzate
 - Tutte le opzioni
 - Riga di comando
- Linker
- Strumento Manifesto
- Generatore di documenti
- Informazioni di visualizzazione
- Eventi di compilazione
- Istruzione di compilazione
- Analisi codice

Tutte le opzioni

/GS /TC <opzioni diverse>

Opzioni aggiuntive Eredita da padre o da impostazioni predefinite progetto

/wd4996

OK Annulla Applica

Specifica il nome del progetto.

Primo Programma

```
#include <stdio.h>
```

```
void main(void){  
    int a;  
    float b;  
    a = 15;  
    b = a*3.3;  
    printf("il valore di a = %d, il valore di b= %f", a, b);  
}
```

- Il programma è il corpo della funzione:
 - void main(void){corpo}

Primo Programma su Visual Studio (1)

PrimoProgramma - Microsoft Visual Studio

File Modifica Visualizza Progetto Compilazione Debug Team Strumenti Test Analizza Finestra ?

Debug x86 Debugger Windows locale

```
1 #include <stdio.h>
2
3 void main(void) {
4     int a;
5     float b;
6     a = 15;
7     b = a*3.3;
8     printf("il valore di a = %d, il valore di b = %f",
9         a, b);
10 }
```

1) Scrivere il programma nel file Main.c

Esplora soluzioni

Cerca in Esplora soluzioni (CTRL+E)

Soluzione 'PrimoProgramma' (1 progetto)

- PrimoProgramma
 - Riferimenti
 - Dipendenze esterne
 - File di intestazione
 - File di origine
 - Main.c
 - File di risorse

Esplora soluzioni Visualizzazione classi

Proprietà

Main.c Proprietà file

Varie	
(Nome)	Main.c
Contenuto	False
Incluso nel progetto	True
Percorso completo	C:\Users\disit\Docume
Percorso relativo	Main.c

(Nome)
Attribuisce un nome all'oggetto del file.

Pronto

Chiedimi qualcosa

18:49
12/03/2017

Primo Programma su Visual Studio (2)

PrimoProgramma - Microsoft Visual Studio

File Modifica Visualizzazione Progetto **Compilazione** Debug Team Strumenti Test Analizza Finestra ?

Compila soluzione CTRL+MAIUSC+B

- Ricompila soluzione
- Pulisci soluzione
- Esegui analisi del codice sulla soluzione ALT+F11
- Compila PrimoProgramma
- Ricompila PrimoProgramma
- Pulisci PrimoProgramma
- Solo progetto
- Compilazione batch...
- Gestione configurazione...
- Compila CTRL+F7

```
1 #include <stdio.h>
2
3 void main(void)
4 {
5     int a;
6     float b;
7     a = 15;
8     b = a*3.3;
9     printf("il valore di a e b sono: %d e %f\n", a, b);
10 }
```

2) Compilare il programma:
Compilazione > Compila Soluzione

3) Nella finestra di output sono riportati i messaggi relativi alla compilazione

Output

Mostra output di: Compilazione

```
1>----- Inizio compilazione: Progetto: PrimoProgramma, Configurazione: Debug Win32 -----
1> Main.c
1>c:\Users\disit\documents\visual studio 2015\projects\primoprogramma\primoprogramma\main.c(7): warning C4244: '=': conversione da 'double' a 'float'
1> PrimoProgramma.vcxproj -> C:\Users\disit\Documents\Visual Studio 2015\Projects\PrimoProgramma\Debug\PrimoProgramma.exe
1> PrimoProgramma.vcxproj -> C:\Users\disit\Documents\Visual Studio 2015\Projects\PrimoProgramma\Debug\PrimoProgramma.pdb (Full PDB)
===== Compilazione: 1 completate, 0 non riuscite, 0 aggiornate, 0 ignorate =====
```

Proprietà

main VCCodeFunction

Proprietà	Valore
Name	main
File	c:\Users\disit\Documents\PrimoProgramma\main.c
FullName	main
IsDefault	False
IsDelete	False

Compilazione completata

Chiedimi qualcosa

18:48
12/03/2017

Primo Programma su Visual Studio (3)

The screenshot shows the Visual Studio IDE with the 'PrimoProgramma' project open. The 'Debug' menu is open, and the option 'Avvia senza eseguire debug' (Ctrl+F5) is highlighted with a red box. A red arrow points from the 'Debug' menu item in the top toolbar to the menu. The code editor shows the following C code:

```
1 #include <stdio.h>
2
3 void main(void){
4     int a;
5     float b;
6     a = 15;
7     b = a*3.3;
8     printf("il valore di a =
9         a, b);
10 }
```

The Output window shows the compilation output:

```
1>----- Inizio compilazione: Progetto: PrimoProgramma, Configurazione: Debug Win32 -----
===== Compilazione: 1 completate, 0 non riuscite, 0 aggiornate, 0 ignorate =====
```

**4) Lanciare il programma
Debug > Avvia senza eseguire debug**

Primo Programma su Visual Studio (4)

The screenshot displays the Visual Studio IDE with a C program named 'PrimoProgramma'. The code in 'Main.c' is as follows:

```
1 #include <stdio.h>
2
3 void main(void) {
4     int a;
5     float b;
6     a = 15;
7     b = a*3.3;
8     printf("il valore di a = %d, il valore di b= %f\n",
9         a, b);
10 }
```

The console window, titled 'C:\WINDOWS\system32\cmd.exe', shows the output of the program:

```
il valore di a = 15, il valore di b= 49.500000
Premere un tasto per continuare . . .
```

The text '5) Visualizzazione Console' is overlaid in red on the code editor. The 'Output' window at the bottom shows 'Elenco errori' and 'Output'. The status bar at the bottom indicates 'Compilazione completata'.

Visual Studio

Utilizzo del Debugger



Concetto di Debug (1)

- Il Debugger serve come supporto per eliminare gli eventuali errori di programmazione
- Gli errori in un programma (che implementa un algoritmo) possono essere di tipo:
 - Sintattico: segnati dal Debugger
 - Semantico: NON sono segnalati dal Debugger. Il programma non fa quello che dovrebbe ...
- Il debugger serve anche per verificare che tutti i passaggi dell'algoritmo (istruzioni) siano corretti
 - Il programmatore deve infatti sapere quali sono gli effetti di ogni istruzione che viene eseguita dal compilatore

Concetto di Debug (2)

- L'ambiente di sviluppo (in questo caso Visual Studio) fornisce degli strumenti per monitorare lo stato del programma
 - Ad esempio permette di vedere il valore delle variabili in un determinato 'momento di esecuzione'
- Una volta 'Avviato' il Debugger  è possibile controllare l'esecuzione del programma istruzione per istruzione, grazie all'uso degli step 
 - Step Into -> continua il debug entrando nel codice di una funzione (si vedrà meglio con la definizione delle funzioni)
 - Step Over -> continua il debug ripartendo dal punto successivo alla riga di esecuzione attuale
 - Step Out -> esce dalla funzione (opposto a step into), da approfondire in futuro...

Concetto di Breakpoint

- I breakpoint (punti di interruzione) vengono inseriti in corrispondenza delle istruzioni del programma che hanno rilevanza nel monitoraggio
- Una volta bloccata l'esecuzione del programma è possibile:
 - Continuare fino al breakpoint successivo
 - Continuare passo passo, ovvero istruzione per istruzione (usando uno dei vari step)
- Inoltre è possibile:
 - Monitorare le variabili 'locali' presenti nel programma via via che viene eseguito
 - Inserire degli watch, ovvero delle variabili aggiuntive o espressioni da monitorare

Visual Studio: Debugger (1)

The screenshot shows the Visual Studio IDE with a C program named 'Main.c' open. The code is as follows:

```
1 #include <stdio.h>
2 int count, sum;
3 void main(void) {
4     for (count = 0; count <10; ++count) {
5         printf("entro nel ciclo for e count vale: %d e sum adesso vale %d\n", count, sum);
6         sum + count;
7     }
8 }
9
10 }
```

A red arrow points to a red circle on line 6, indicating a break point. The status bar shows the path: 'Percorso: Main.c, riga 6 ('main(void)')'. The Output window at the bottom shows the following text:

```
Mostra output di: Debug
'PrimoProgramma.exe' (Win32): caricamento di 'C:\Windows\SysWOW64\kernel32.dll' completato. Impossibile trovare o aprire il file PDB.
'PrimoProgramma.exe' (Win32): caricamento di 'C:\Windows\SysWOW64\KernelBase.dll' completato. Impossibile trovare o aprire il file PDB.
'PrimoProgramma.exe' (Win32): caricamento di 'C:\Windows\SysWOW64\vcruntime140d.dll' completato. Impossibile trovare o aprire il file PDB
'PrimoProgramma.exe' (Win32): caricamento di 'C:\Windows\SysWOW64\ucrtbased.dll' completato. Impossibile trovare o aprire il file PDB.
Il programma '[11224] PrimoProgramma.exe' è terminato con il codice 0 (0x0).
```

1) Posizionare il punto di interruzione (break point)

Visual Studio: Debugger (3)

2) Il compilatore ferma l'esecuzione del programma quando trova il (primo) break point

The screenshot displays the Visual Studio IDE in debug mode. The main window shows the source code of a C program named 'PrimoProgramma'. The code is as follows:

```
1 #include <stdio.h>
2 int count, sum;
3 void main(void) {
4
5     for (count = 0; count <10; ++count) {
6         printf("entro nel ciclo for e count vale: %d e sum adesso vale %d\n", count, sum);
7         sum = sum + count;
8     }
9
10    printf("Esco dal ciclo\n");
11
12 }
```

A red arrow points to a yellow break point icon on line 5. The 'Auto' window at the bottom left shows the following variables:

Nome	Valore	Tipo
count	0	int
sum	0	int

The 'Strumenti di diagnostica' window on the right shows a diagnostic session of 0 seconds (84 ms selected). The 'Elenco errori' window at the bottom right shows 0 errors, 0 warnings, and 0 messages.

Visual Studio: Debugger (4)

3) Se si clicca F10 'Esegui istruzione', il compilatore esegue l'istruzione successiva a quella in cui si trova il break point e si ferma

PrimoProgramma (In debug) - Microsoft Visual Studio

File Modifica Visualizza Progetto Compilazione Debug Team Strumenti Test Analizza Finestra

Processo: [9040] PrimoProgramma.exe Thread: [11032] Thread principale Stack frame

Strumenti di diagnostica

Sessione di diagnostica: 0 secondi (84 ms selezionati)

83,2ms 83,4ms

Eventi

Memoria processi Snapshot Byte privati

100 100

Eventi Utilizzo memoria Utilizzo CPU

Cerca negli eventi

Evento	Ora	Durata	Thread
Passaggio registrato	0,08 s	1 ms	[11032]
Passaggio registrato	0,08 s	1 ms	[11032]

```
1 #include <stdio.h>
2 int count, sum;
3 void main(void) {
4
5     for (count = 0; count <10; ++count) {
6         printf("entro nel ciclo for e count vale: %d e sum adesso vale %d\n", count, sum);
7         sum = sum + count;
8     }
9
10    printf("Esco dal ciclo\n");
11
12 }
```

Auto

Nome	Valore	Tipo
count	0	int
sum	0	int

Elenco errori

Intera soluzione 0 Errori 0 Avvisi 0 Messaggi

Elenco errori di ricerca

Codice	Descrizione	Progetto	File
--------	-------------	----------	------

Auto Variabili locali Espressione di controllo 1

Stack di chiam... Punti di interru... Impostazioni e... Finestra di com... Finestra di cont... Output Elenco errori

NOTA: la freccia gialla indica la riga della prossima istruzione (che ancora NON è stata eseguita)

Visual Studio: Debugger (5)

3) Se si clicca su F5 (Continua), il compilatore esegue tutte le istruzioni finché non incontra il break point successivo

The screenshot shows the Visual Studio IDE with the debugger active. The main window displays the source code of 'PrimoProgramma.c'. A red arrow points to the 'Continua' button in the toolbar. The code is as follows:

```
1 #include <stdio.h>
2 int count, sum;
3 void main(void) {
4
5     for (count = 0; count < 10; ++count) {
6         printf("entro nel ciclo for e count vale: %d e sum adesso vale %d\n", count, sum);
7         sum = sum + count;
8     }
9
10    printf("Esco dal ciclo\n");
11
12 }
```

The 'Auto' window shows the following variables:

Nome	Valore	Tipo
count	8	int
sum	28	int

The 'Elenco errori' window shows 0 errors, 0 warnings, and 0 messages.

NOTA: SE count ha valore 8 -> questa è la nona volta che si entra dentro al ciclo (perché all'inizio count vale zero)

Visual Studio: Debugger (6)

4) E' Possibile vedere passo passo cosa viene scritto nella Console.

Prima volta che si entra nel ciclo

```
1 #include <stdio.h>
2 int count, sum;
3 void main(void) {
4
5     for (count = 0; count < 10; ++count) {
6         printf("entro nel ciclo for e count vale: %d e sum adesso vale %d\n", count, sum);
7         sum = sum + count;
8     }
9
10    printf("Esco dal ciclo\n");
11
12 }
```

Console Output:

```
entro nel ciclo for e count vale: 0 e sum adesso vale 0
```

Nome	Valore
printf restituito	56
count	0
sum	0

Stack di chiam... Puntii di interr... Impostazioni e... Finestra di co... Finestra di con... Output Elenco errori

Visual Studio: Debugger (6)

4) E' Possibile vedere passo passo cosa viene scritto nella Console.

Seconda volta che si entra nel ciclo

```
1 #include <stdio.h>
2 int count, sum;
3 void main(void) {
4     for (count = 0; count < 10; ++count) {
5         printf("entro nel ciclo for e count vale: %d e sum adesso vale %d\n", count, sum);
6         sum = sum + count;
7     }
8     printf("Esco dal ciclo\n");
9 }
10
11
12 }
```

Sessione di diagnostica: 0 secondi (1...)

56m

Eventi

Memoria processi Byte privati

CPU (% di tutti i processori)

C:\Users\disit\Documents\Visual Studio 2015\Projects\PrimoProgramma\Debug\PrimoProgramma.exe

entro nel ciclo for e count vale: 0 e sum adesso vale 0
entro nel ciclo for e count vale: 1 e sum adesso vale 0

Nome	Valore
printf restituito	56
count	1
sum	0

Stack di chiam... Punti di interru... Impostazioni e... Finestra di co... Finestra di con... Output Elenco errori

Visual Studio: Debugger (6)

4) E' Possibile vedere passo passo cosa viene scritto nella Console.

Nona volta che si entra nel ciclo

```
1  #include <stdio.h>
2  int count, sum;
3  void main(void) {
4
5  for (count = 0; count < 10; ++count) {
6  printf("entro nel ciclo for e count vale: %d e sum adesso vale %d\n", count, sum);
7  sum = sum + count;
8  }
9
10 printf("Esco dal ciclo\n");
11
12 }
```

Console Output:

```
entro nel ciclo for e count vale: 0 e sum adesso vale 0
entro nel ciclo for e count vale: 1 e sum adesso vale 0
entro nel ciclo for e count vale: 2 e sum adesso vale 1
entro nel ciclo for e count vale: 3 e sum adesso vale 3
entro nel ciclo for e count vale: 4 e sum adesso vale 6
entro nel ciclo for e count vale: 5 e sum adesso vale 10
entro nel ciclo for e count vale: 6 e sum adesso vale 15
entro nel ciclo for e count vale: 7 e sum adesso vale 21
entro nel ciclo for e count vale: 8 e sum adesso vale 28
```

Nome	Valore
printf restituito	57
count	8
sum	28

Visual Studio: Debugger (6)

4) E' Possibile vedere passo passo cosa viene scritto nella Console.

The screenshot shows the Visual Studio IDE in debug mode. The main window displays the source code of a C program named 'PrimoProgramma'. The code is as follows:

```
1 #include <stdio.h>
2 int count, sum;
3 void main(void) {
4
5     for (count = 0; count < 10; ++count) {
6         printf("entro nel ciclo for e count vale: %d e sum adesso vale %d\n", count, sum);
7         sum = sum + count;
8     }
9
10    printf("Esco dal ciclo\n");
11
12 }
```

The debugger is currently stopped at line 10. The 'Auto' window shows the following data:

Nome	Valore
printf restituito	15

The console window shows the output of the program:

```
entro nel ciclo for e count vale: 0 e sum adesso vale 0
entro nel ciclo for e count vale: 1 e sum adesso vale 0
entro nel ciclo for e count vale: 2 e sum adesso vale 1
entro nel ciclo for e count vale: 3 e sum adesso vale 3
entro nel ciclo for e count vale: 4 e sum adesso vale 6
entro nel ciclo for e count vale: 5 e sum adesso vale 10
entro nel ciclo for e count vale: 6 e sum adesso vale 15
entro nel ciclo for e count vale: 7 e sum adesso vale 21
entro nel ciclo for e count vale: 8 e sum adesso vale 28
entro nel ciclo for e count vale: 9 e sum adesso vale 36
Esco dal ciclo
```

The text 'Uscita dal ciclo' is overlaid in red on the code editor. The 'Stack di chiam...' window at the bottom shows the call stack with the current frame being 'main'.

Rappresentazione dei dati



Outline

- **Rappresentazione**
 - Un frammento del linguaggio C
 - Tipi, variabili e costanti
 - Operatori ed espressioni
 - Istruzioni
 - Rappresentazione dei dati: 
 - Numeri
 - Interi senza segno
 - Caratteri
 - Interi con segno

Rappresentazione dei dati

- Esistono due tipi di dati che possono essere rappresentati nella codifica di un programma:
 - Valori numerici:
 - tipo int
 - tipo float
 - tipo double
 - Caratteri:
 - tipo char
- Il char è a sua volta codificato nella forma di un numero



Numeri

- Un NUMERO è un ente dotato di un suo significato intrinseco, del quale è possibile dare rappresentazioni diverse
- ESEMPIO:
 - Il numero 'quattro' definisce un concetto che ha un proprio significato. E' possibile associare a tale concetto rappresentazioni diverse:
 - IV, 4, 100 (in binario)
- Le diverse rappresentazioni sono caratterizzate dalla combinazione di più convenzioni:
 - Posizionalità della codifica
 - Numero di cifre
 - Codifica del segno
 - Rappresentazione di parti frazionarie e valori razionali

Codifica posizionale

- I Numeri Naturali sono codificati attraverso una base finita di cifre elementari (0-9)
- Il peso delle cifre dipende dalla posizione in cui appaiono:
- $[a_N a_{N-1} \dots a_1 a_0]_B ::= \sum_{n=0}^N a_n B^n$
- Con:
 - B = base di numerazione della codifica posizionale
 - a_N cifra più significativa (Most Significant Digit, MSD)
 - a_0 cifra meno significativa (Least Significant Digit, LSD)
- Una convenzione aggiuntiva definisce l'ordine con cui sono presentate le cifre:
 - **Big Endian**: in ultima posizione la cifra più significativa (leggendo da sx a dx, il MSD è a sinistra)
 - **Little Endian**: in ultima posizione la cifra meno significativa (leggendo da sx a dx, il MSD è a destra)

Base di numerazione

- La base di numerazione che si usa nella vita quotidiana è la base 10
- I calcolatori usano la base 2
- La base 2 è la base minima che include cifre diverse, ovvero la base minima su cui è possibile contare in maniera posizionale
- E' utile studiare la conversione della base di rappresentazione



Operazioni con i binari

SOMMA

$$0 + 0 = 0$$

$$0 + 1 = 1$$

$$1 + 0 = 1$$

$$1 + 1 = 0, \text{ con riporto di } 1$$

PRODOTTO

$$0 * 0 = 0$$

$$0 * 1 = 0$$

$$1 * 0 = 0$$

$$1 * 1 = 1$$

SOTTRAZIONE

$$0 - 0 = 0$$

$$1 - 1 = 0$$

$$1 - 0 = 1$$

$$0 - 1 = 1, \text{ con prestito di } 1$$

DIVISIONE

Dividendo/divisore = 1, SE ci sta

Dividendo/divisore = 0, SE non ci sta

Esempi di somma

Base 10	Base 2
	1 1 1
19 +	1 0 0 1 1 +
17 =	1 0 0 0 1 =
36	1 0 0 1 0 0

<u>SOMMA</u>
$0 + 0 = 0$
$0 + 1 = 1$
$1 + 0 = 1$
$1 + 1 = 0, \text{ con riporto di } 1$

Base 10	Base 2
	1 1 1 1 1 1
55 +	1 1 0 1 1 1 +
29 =	0 1 1 1 0 1 =
84	1 0 1 0 1 0 0

Base 10	Base 2
	1 1 1 1 1 1
103 +	1 1 0 0 1 1 1 +
41 =	0 1 0 1 0 0 1 =
144	1 0 0 1 0 0 0 0

Esempi di sottrazione

Base 10	Base 2
	0 1 0 1 0 1
84 -	1 0 1 0 1 0 0 -
29 =	0 0 1 1 1 0 1 =
55	0 1 1 0 1 1 1

SOTTRAZIONE

$$0 - 0 = 0$$

$$1 - 1 = 0$$

$$1 - 0 = 1$$

$$0 - 1 = 1, \text{ con prestito di } 1$$

Base 10	Base 2
	0 0 1
155 -	1 1 1 0 0 1 1 -
23 =	0 0 1 0 1 1 1 =
92	1 0 1 1 1 0 0

Base 10	Base 2
	0 0 0 1 1 0
178 -	1 0 1 1 0 0 1 0 -
95 =	0 1 0 1 1 1 1 1 =
83	0 1 0 1 0 0 1 1

Conversione della base di rappresentazione(1)

- Esistono due algoritmi di conversione della base di rappresentazione
 - Basati sull'uso dell'aritmetica di base
 - Basati sull'uso dell'aritmetica di arrivo
- **CASO 1a): Conversione in base di arrivo DA base 10 A base 2, Algoritmo:**
 - 1) Il numero iniziale viene sviluppato in forma polinomiale nella base di partenza
 - 2) I singoli coefficienti e potenze che ne derivano sono convertiti nella base di arrivo (occorre quindi conoscerne la conversione)
 - 3) La conversione è completata effettuando somme e prodotti nella base di arrivo

• Esempio:

$$[125]_{10} = [1 \cdot 10^2 + 2 \cdot 10 + 5]_{10} = [1 \cdot 1010^2 + 10 \cdot 1010 + 0101]_2$$

$$[a_N a_{N-1} a_1 a_0]_B ::= \sum_{n=0}^N a_n B^n$$



Conversione della base di rappresentazione (2)

- Esempio:
 - $[125]_{10} = [1 \cdot 10^2 + 2 \cdot 10 + 5]_{10} = //$ 1) sviluppo in forma polinomiale
 - $[1 \cdot 10^2 + 0 \cdot 10 + 10 + 10 + 0 + 1]_2 //$ 2) coefficienti e potenze sono convertiti nella base di arrivo
- Con:
 - $[1]_{10} = [1]_2$
 - $[2]_{10} = [10]_2$
 - $[5]_{10} = [101]_2$
- 3) a questo punto è necessario fare la somma dei singoli addendi (dal punto 2)), si procede con il calcolo dei prodotti, ricordando le regole di base

Conversione della base di rappresentazione (3)

Calcolo dei prodotti, ricordando le regole di base:

$$1010 * 1010$$

$$\begin{array}{r} \text{-----} \\ 0000 \\ 1010 \\ 0000 \\ 1010 \\ \text{-----} \\ 1100100 \end{array}$$

- Ripetendo per i vari prodotti e facendo la somma finale si ottiene:

$$[125]_{10} = [1100100 + 10100 + 0101]_2 = [1111101]_2$$

Esempi conversione in base di arrivo da base 10 a base 2 (1)

$$[134]_{10} = [10 \cdot 10 + 2 \cdot 10 + 4]_{10} = [1010 \cdot 1010 + 0011 \cdot 1010 + 0100]_2 =$$

$$\begin{array}{r} 1010 \cdot 1010 \\ \hline 0000 \\ 11010 \\ 0000 \\ 1010 \\ \hline 1100100 \end{array}$$

$$\begin{array}{r} 0011 \cdot 1010 \\ \hline 0000 \\ 0011 \\ 0000 \\ 0011 \\ \hline 0011110 \end{array}$$

Base 10	Base 2
	1 1 1 1
100 +	1 1 0 0 1 0 0 +
30 +	0 0 1 1 1 1 0 +
4 =	0 0 0 0 1 0 0 =
134	1 0 0 0 0 1 1 0

$$[134]_{10} = [10000110]_2$$

Esempi conversione in base di arrivo da base 10 a base 2 (2)

$$[671]_{10} = [6 \cdot 10^2 + 7 \cdot 10 + 1]_{10} = [0110 \cdot 1010 \cdot 1010 + 0111 \cdot 1010 + 0001]_2$$

Sapendo che: $[1010 \cdot 1010]_2 = [1100100]_2$

$$\begin{array}{r}
 [600]_{10} \quad \underline{1100100 \cdot 0110} \\
 \quad \quad \quad 0000000 \\
 \text{1} \quad 1100100 \\
 \quad \quad 1100100 \\
 \quad \quad \quad 0000000 \\
 \hline
 1001011000
 \end{array}$$

$$\begin{array}{r}
 \quad \quad \quad \underline{0111 \cdot 1010} \\
 \quad \quad \quad \text{10000} \\
 \quad \quad \text{10111} \\
 \quad \text{10000} \\
 \quad \quad 0111 \\
 \hline
 1000110
 \end{array}$$

Base 10	Base 2
	1
600 +	1001011000 +
70 +	0001000110 +
1 =	0000000001 =
671	1010011111

$$[671]_{10} = [1010011111]_2$$



Conversione della base di rappresentazione (4)

- CASO 1b): **Conversione in base di arrivo DA base 2 A base 10:**

$$[1100100]_2 = [1*2^6 + 1*2^5 + 1*2^2]_{10} = [64 + 32 + 4]_{10} = [100]_{10}$$

NOTA: quando la conversione è da base 2 a base 10, conviene fare i conti in base di arrivo

Esercizi conversione in base di arrivo

da base 2 a base 10

- $[1100111]_2 = [1 \cdot 2^6 + 1 \cdot 2^5 + 0 \cdot 2^4 + 0 \cdot 2^3 + 1 \cdot 2^2 + 1 \cdot 2^1 + 1 \cdot 2^0]_{10} = [64 + 32 + 4 + 2 + 1]_{10} = [103]_{10}$
- $[10010110]_2 = [1 \cdot 2^7 + 0 \cdot 2^6 + 0 \cdot 2^5 + 1 \cdot 2^4 + 0 \cdot 2^3 + 1 \cdot 2^2 + 1 \cdot 2^1 + 0 \cdot 2^0]_{10} = [128 + 16 + 4 + 2]_{10} = [150]_{10}$
- $[10010]_2 = [1 \cdot 2^4 + 0 \cdot 2^3 + 0 \cdot 2^2 + 1 \cdot 2^1 + 0 \cdot 2^0]_{10} = [16 + 2]_{10} = [18]_{10}$
- $[11111110]_2 = [1 \cdot 2^8 + 1 \cdot 2^7 + 1 \cdot 2^6 + 1 \cdot 2^5 + 1 \cdot 2^4 + 1 \cdot 2^3 + 1 \cdot 2^2 + 1 \cdot 2^1 + 0 \cdot 2^0]_{10} = [256 + 128 + 64 + 32 + 16 + 8 + 4 + 2]_{10} = [510]_{10}$

Conversione della base di rappresentazione (5)

- CASO 2): **Conversione in base di partenza DA base 10 A base 2:**
- Algoritmo dei resti successivi
- In questo caso conviene usarlo per passare dalla base 10 alla base 2
- La rappresentazione di un numero A in base 2 è determinata dalla equazione:

$$\begin{aligned} A &= \sum_{k=0}^{\infty} a_k 2^k = \\ &= a_0 * 2^0 + \sum_{k=1}^{\infty} a_k 2^k = \quad // \text{sia } k=z+1 \\ &= a_0 * 1 + \sum_{z=0}^{\infty} a_{z+1} 2^{z+1} = \quad // \text{sostituisco } k=z \\ &= a_0 + 2 \sum_{k=0}^{\infty} a_{k+1} 2^k \end{aligned}$$

Conversione della base di rappresentazione (6)

- Dalla equazione precedente si deduce che a_0 vale 1 Se e solo se A è dispari ($A = a_0 + \text{numero pari}$)
- Si può calcolare a_0 come il **resto** della divisione intera $A/2$:

$$A/2 = \sum_{k=0}^{\infty} a_{k+1} 2^k$$

- Per determinare a_1 e tutti gli altri coefficienti si procede analogamente. Il procedimento termina quando il resto della divisione per 2 è 0

Base10 | Resto divisione per due

125	1
62	0
31	1
15	1
7	1
3	1
1	1
0	0

$$[125]_{10} = [1111101]_2$$

Conversione in base di partenza DA base 10 A base 2

356	0
178	0
89	1
44	0
22	0
11	1
5	1
2	0
1	1
0	0

$$[356]_{10} = [101100100]_2$$

1279	1
639	1
319	1
159	1
79	1
39	1
19	1
9	1
4	0
2	0
1	1
0	0

$$[1279]_{10} = [1001111111]_2$$

596	0
298	0
149	1
74	0
37	1
18	0
9	1
4	0
2	0
1	1
0	0

$$[596]_{10} = [1001010100]_2$$

Base esadecimale (1)

- I numeri in base due sono lunghi
- Memorizzare la rappresentazione in base 10 è semplice, mentre quella in base due è più complessa:
 - $[125]_{10} = [1111101]_2$
- Per ottenere una codifica più compatta si usa spesso la base 16
- In base 16:
 - Ci sono 16 cifre: [0-9] e A,B,C,D,E,F
 - Le cifre rappresentano i numeri in base 10, rispettivamente: 11,12,13,14,15
- La conversione da base 2 a base esadecimale si ottiene 'impaccando' i bit:
 - $[125]_{10} = [1111101]_2 = [01111101]_2 = [7D]_{16}$

Base esadecimale (2)

- Partendo da:
 - $[125]_{10} = [1111101]_2$
- La conversione da base 2 a base esadecimale si ottiene 'impaccando' i bit:
 - $[125]_{10} = [01111101]_2 = [FD]_{16}$
- Con:
 - $[0111]_2 = [0 \cdot 2^3 + 1 \cdot 2^2 + 1 \cdot 2^1 + 1 \cdot 2^0]_{10} = [4 + 2 + 1]_{10} = [7]_{10} = [7]_{16}$
 - $[1101]_2 = [1 \cdot 2^3 + 1 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0]_{10} = [13]_{10} = [D]_{16}$

Esercizi conversione da base 10 a base esadecimale (1)

1) conversione da base 10 a base 2

$$[497]_{10} = [4 \cdot 10^2 + 9 \cdot 10 + 7]_{10} = [0100 \cdot 1010 \cdot 1010 + 1001 \cdot 1010 + 0111]_2$$

Con: $[1010 \cdot 1010]_2 = 1100100$

$ \begin{array}{r} 1100100 \cdot 0100 \\ \hline 0000000 \\ 0000000 \\ 1100100 \\ 0000000 \\ \hline 0110010000 \end{array} $	$ \begin{array}{r} 1001 \cdot 1010 \\ \hline 0000 \\ 1001 \\ 0000 \\ 1001 \\ \hline 1011010 \end{array} $
Base 10	Base 2
$ \begin{array}{r} 400 + \\ 90 + \\ 7 = \\ \hline 497 \end{array} $	$ \begin{array}{r} 1111 \\ 0110010000 + \\ 0001011010 + \\ 0000000111 = \\ \hline 0111110001 \end{array} $



Esercizi conversione da base 10 a base esadecimale (2)

2) si 'impaccano' i bit (a partire da destra)

$$\begin{aligned}[497]_{10} &= [4 \cdot 10^2 + 9 \cdot 10 + 7]_{10} = [0100 \cdot 1010 \cdot 1010 + 1001 \cdot 1010 + 0111]_2 \\ &= [0111110001]_2 =\end{aligned}$$

$$[0001]_2 = [2^0]_{10} = [1]_{10} = [1]_{16}$$

$$[1111]_2 = [2^3 + 2^2 + 2^1 + 2^0]_{10} = [15]_{10} = [F]_{16}$$

$$[0001]_2 = [2^0]_{10} = [1]_{10} = [1]_{16}$$

$$[497]_{10} = [0111110001]_2 = [1F1]_{16}$$

Esercizi conversione da base 10 a base esadecimale (3)

1) conversione da base 10 a base 2

$$[345]_{10} = [3 \cdot 10^2 + 4 \cdot 10 + 5]_{10} = [0100 \cdot 1010 \cdot 1010 + 1001 \cdot 1010 + 0101]_2$$

Con: $[1010 \cdot 1010]_2 = 1100100$

$ \begin{array}{r} 1100100 \cdot 0011 \\ \hline 11100100 \\ 11100100 \\ 00000000 \\ 00000000 \\ \hline 0100101100 \end{array} $	$ \begin{array}{r} 0100 \cdot 1010 \\ \hline 0000 \\ 0100 \\ 0000 \\ 0100 \\ \hline 0101000 \end{array} $
Base 10	Base 2
$ \begin{array}{r} 300 + \\ 40 + \\ 5 = \\ \hline 345 \end{array} $	$ \begin{array}{r} 111 \\ 0100101100 + \\ 0000101000 + \\ 0000000101 = \\ \hline 0101011001 \end{array} $



Esercizi conversione da base 10 a base esadecimale (4)

2) si 'impaccano' i bit (a partire da destra)

$$\begin{aligned}[345]_{10} &= [3 \cdot 10^2 + 4 \cdot 10 + 5]_{10} = [0100 \cdot 1010 \cdot 1010 + 1001 \cdot 1010 + 0101]_2 \\ &= [101011001]_2 =\end{aligned}$$

$$[0001]_2 = [2^0]_{10} = [1]_{10} = [1]_{16}$$

$$[0101]_2 = [2^2 + 2^0]_{10} = [5]_{10} = [5]_{16}$$

$$[1001]_2 = [2^2 + 2^0]_{10} = [9]_{10} = [9]_{16}$$

$$[497]_{10} = [0111110001]_2 = [159]_{16}$$

Parti Frazionarie (1)

- Fino ad ora abbiamo visto le conversioni di numeri interi
- Una parte frazionaria è un numero razionale inferiore all'unità
- Come si è visto per i numeri interi, anche nel caso della parte frazionaria, la conversione può essere effettuata in due modalità:
 - Base di arrivo
 - Base di partenza

Parti Frazionarie -

Conversione in base di arrivo (1)

- **Conversione in base di arrivo:**
 - (a) Si sviluppa la parte frazionaria in forma polinomiale nella base di partenza
 - (b) Si convertono i singoli termini nella base di arrivo
 - (c) Si eseguono le operazioni in base di arrivo
- **Esempio:**
 - $[0.7]_{10} \stackrel{(a)}{=} [7 \cdot 10^{-1}]_{10} = [7/10]_{10} \stackrel{(b)}{=} [0111/1010]_2$
 - Si esegue poi la divisione tra binari (c)
 - Le regole per eseguire la divisione tra binari sono le stesse regole usate nei decimali

Parti Frazionarie -

Conversione in base di arrivo (2)

	0 1 1 1.0 0 0 0 0 0
0.	<u>0 0 0 0</u>
	1 1 1 0(-)
1	<u>1 0 1 0</u>
—	0 1 0 0 0(-)
0	<u>0 0 0 0</u>
	1 0 0 0 0(-)
1	<u>1 0 1 0</u>
	0 1 1 0 0(-)
1	<u>1 0 1 0</u>
	0 0 1 0 0(-)
0	<u>0 0 0 0 0</u>
	0 1 0 0 0(-)
	<u>0 0 0 0 0</u>
0	1 0 0 0

	1 0 1 0
	<u>0.1 0 1 1 0</u>

Tenere presenti le regole della
SOTTRAZIONE

$$0 - 0 = 0$$

$$1 - 1 = 0$$

$$1 - 0 = 1$$

$$0 - 1 = 1, \text{ con prestito di } 1$$

Parti Frazionarie -

Conversione in base di arrivo (2)

	0 1 1 1.0 0 0 0 0 0 0
	0 0 0 0 0
	1 1 1 0(-)
1	1 0 1 0
	<u>0 1 0 0 0(-)</u>
0	0 0 0 0
	1 0 0 0 0(-)
1	1 0 1 0
	0 1 1 0 0(-)
1	1 0 1 0
	0 0 1 0 0(-)
0	0 0 0 0 0
	<u>0 1 0 0 0(-)</u>
	0 0 0 0 0
0	1 0 0 0

1 0 1 0	
0.1	<u>0 1 1 0</u>

Le ultime 4 cifre si ripetono periodicamente, quindi abbiamo:
 $[0.7]_{10} = [0.\overline{10110}]_2$

Esempio2 conversione parti frazionarie in base di arrivo

- $[0.3]_{10} = [3 \cdot 10^{-1}]_{10} = [3/10]_{10} = [0011/1010]_2$

0.	0 0 1 1.0 0 0 0 0 0 0
	0 0 0 0
0	0 1 1 0(-)
	0 0 0 0
1	1 1 0 0(-)
	1 0 1 0
0	0 0 1 0 0(-)
	0 0 0 0
0	0 1 0 0 0(-)
	0 0 0 0
1	1 0 0 0 0(-)
	0 1 0 1 0
0	0 0 1 1 0 0(-)
.....	

$$\begin{array}{r} 1010 \\ \hline 0.01001 \end{array}$$

Le 5 cifre dopo la virgola si ripetono periodicamente, quindi abbiamo:

$$[0.3]_{10} = [0.01001]_2$$

Tenere presenti le regole della SOTTRAZIONE

$$0 - 0 = 0$$

$$1 - 1 = 0$$

$$1 - 0 = 1$$

$$0 - 1 = 1, \text{ con prestito di } 1$$

Parti Frazionarie - Conversione in base di partenza (1)

- Conversione in base di partenza:
 - (a) Si moltiplica per 2 la parte frazionaria
 - (b) Si sottrae dal risultato ottenuto in (a) la sua parte intera
 - (c) La sequenza delle parti intere sottratte fornisce la rappresentazione voluta
- Esempio:
 - $[0.7]_{10}$

	$0.7 * 2$
	<hr/>
	1.4(-)
1	<hr/>
	1.0
	<hr/>
	$0.4 * 2$
	<hr/>
	0.8(-)
0	<hr/>
	0.0
	<hr/>
	$0.8 * 2$
	<hr/>
	1.6(-)
1	<hr/>
	1.0
	<hr/>
	$0.6 * 2$
	<hr/>
	1.2(-)
1	<hr/>
	1.0
	<hr/>
	$0.2 * 2$
	<hr/>
	0.4(-)
0	<hr/>
	0.0
	<hr/>
	0.4



0.4

Parti Frazionarie -

Conversione in base di partenza (2)

	$0.7 * 2$
	<hr/>
	1.4(-)
1	1.0
	$0.4 * 2$
	<hr/>
	0.8(-)
0	0.0
	$0.8 * 2$
	<hr/>
	1.6(-)
1	1.0
	$0.6 * 2$
	<hr/>
	1.2(-)
1	1.0
	$0.2 * 2$
	<hr/>
	0.4(-)
0	0.0
	0.4

Le ultime 4 cifre si ripetono periodicamente, quindi abbiamo:

$$[0.7]_{10} = [0.10110]_2$$

Parte frazionaria – conversione in base di partenza

	0.3 * 2
	0.6 (-)
0	0.0
	0.6 * 2
	1.2 (-)
1	1.0
	0.2 * 2
	0.4 (-)
0	0.0
	0.4 * 2
	0.8 (-)
0	0.0
	0.8 * 2
	1.6 (-)
1	1.0
	0.6

Le 5 cifre si ripetono periodicamente, quindi abbiamo:

$$[0.7]_{10} = [0.01001]_2$$



Interi senza segno (1)

- Un intero senza segno rappresenta una variabile dichiarata in C nel modo seguente:
 unsigned int **a**;
- Un intero senza segno viene rappresentato in base 2 su N bit di memoria { a_i } con i che va da 0 a N-1
 - $\mathbf{a} = \sum_{i=0}^{N-1} a_i 2^i$
 - La rappresentazione codifica tutti e soli i numeri che vanno da 0 a 2^N-1 inclusi
- Esempio:
 - Se N=8, allora **a** appartiene all'intervallo [0,255]
- Il numero N di bit varia a seconda dell'architettura della macchina (solitamente a 32 o 64 bit)

Caratteri

- Un carattere rappresenta una variabile, che in C è dichiarata con:
 - `char c;`
- Un carattere è rappresentato in memoria su N=8 bit, che codificano un numero intero senza segno tra 0 e 255
- La corrispondenza tra i valori numerici compresi tra 0 e 255 e i caratteri è stabilita dalla **Tabella dei codici ASCII** che include:
 - Caratteri alfanumerici
 - Cifre decimali
 - Lettere minuscole e maiuscole
 - Simboli di interpunzione e parentesi (es: [] { })
 - Simboli aritmetici (es: * + - %)
 - Caratteri di vario genere (es: @ #)
 - Caratteri di controllo (es: a capo, tabulazione, etc.)

Tabella dei codici ANSI-ASCII

!	032	@	064	`	096	€	0128		0160	À	0192	à	0224
"	033	A	065	a	097	€	0129	¡	0161	Á	0193	á	0225
#	034	B	066	b	098	,	0130	¢	0162	Â	0194	â	0226
\$	035	C	067	c	099	f	0131	£	0163	Ã	0195	ã	0227
%	036	D	068	d	0100	„	0132	¤	0164	Ä	0196	ä	0228
&	037	E	069	e	0101	...	0133	¥	0165	Å	0197	å	0229
'	038	F	070	f	0102	†	0134	¦	0166	Æ	0198	æ	0230
(039	G	071	g	0103	‡	0135	§	0167	Ç	0199	ç	0231
)	040	H	072	h	0104	^	0136	¨	0168	È	0200	è	0232
*	041	I	073	i	0105	%	0137	©	0169	É	0201	é	0233
+	042	J	074	j	0106	Š	0138	ª	0170	Ê	0202	ê	0234
,	043	K	075	k	0107	<	0139	«	0171	Ë	0203	ë	0235
-	044	L	076	l	0108	œ	0140	¬	0172	Ì	0204	ì	0236
.	045	M	077	m	0109	Ž	0141	®	0173	Í	0205	í	0237
/	046	N	078	n	0110	ž	0142	™	0174	Î	0206	î	0238
0	047	O	079	o	0111		0143	—	0175	Ï	0207	ï	0239
1	048	P	080	p	0112	‘	0144	°	0176	Ð	0208	ð	0240
2	049	Q	081	q	0113	’	0145	±	0177	Ñ	0209	ñ	0241
3	050	R	082	r	0114	‚	0146	²	0178	Ò	0210	ò	0242
4	051	S	083	s	0115	“	0147	³	0179	Ó	0211	ó	0243
5	052	T	084	t	0116	”	0148	´	0180	Ô	0212	ô	0244
6	053	U	085	u	0117	•	0149	µ	0181	Õ	0213	õ	0245
7	054	V	086	v	0118	—	0150	¶	0182	Ö	0214	ö	0246
8	055	W	087	w	0119	~	0151	·	0183	×	0215	÷	0247
9	056	X	088	x	0120		0152	¸	0184	Ø	0216	ø	0248
:	057	Y	089	y	0121	™	0153	¹	0185	Ù	0217	ù	0249
;	058	Z	090	z	0122	š	0154	º	0186	Ú	0218	ú	0250
<	059	[091	{	0123	>	0155	»	0187	Û	0219	û	0251
=	060	\	092		0124	œ	0156	¼	0188	Ü	0220	ü	0252
>	061]	093	}	0125		0157	½	0189	Ý	0221	ý	0253
?	062	^	094	~	0126	ž	0158	¾	0190	Þ	0222	þ	0254
	063	_	095		0127	ÿ	0159	¿	0191	ß	0223	ÿ	0255

- *Esempi:
il carattere 0 è
codificato
dal numero 48*

*Le parentesi graffe
dai numeri
123 e 124 ...*

*Il numero 0 NON
codifica alcun
carattere,
mentre invece viene
usato come segno
di terminazione
nella codifica delle
stringhe*

Il complemento

- Dato un numero intero N rappresentato in base b con k cifre, Si definisce C_b , il complemento a b di N in base b , tale che: $N + C_b(N) = b^k$
- Ad esempio, Con $b^k = 1\ 0\ 0\ 0\ 0\ 0$, Per ogni b (1 seguito da k cifre uguali a 0)

· ESEMPIO in base 10:
BASE 10: 3552 è il complemento a 10 di 6448;

Verifica (si effettua la somma):

$$\begin{array}{r} 3552 + \\ 6448 \\ \hline \end{array}$$

$10000 = 10^4$, in questo caso si ha: **$b=10$, $k=4$**



Il complemento

Esempio in base 2:

Dato un numero intero N rappresentato in base 2 con k cifre, Si definisce C_2 , il complemento a 2 di N in base 2, tale che: $N + C_2(N) = 2^k$

$[01100]_2$ è il complemento a 2 di $[10100]_2$

Verifica (si effettua la somma)

1 1 1

0 1 1 0 0

1 0 1 0 0

$1\ 0\ 0\ 0\ 0\ 0 = 2^5 + 0 \cdot 2^4 + \dots + 0 \cdot 2^0 = 2^5$, con $b=2$, $k=5$ bit

Algoritmo per calcolare il complemento a 2 di N

- Sia N numero binario
- Algoritmo:
 - Si copiano i bit del numero a partire dal meno significativo fino al primo 1 compreso
 - Si invertono tutti gli altri bit

▣ Esempio: Calcolare il C2(10100011100000)

$$\begin{array}{r} 10100011100000+ \\ \hline 01011100100000 \leftarrow \text{Risultato} \\ 1000000000000000 \leftarrow \text{Verifica sommando} \end{array}$$

Esempi complemento a 2 (1)

- $C2[01101101] = ??, k =$

2) Si inverte | 1) si copia fino al primo 1 compreso

0 1 1 0 1 1 0	1
1 0 0 1 0 0 1	1

- $C2[01101101] = 10010011$
- Verifica:

Base 10	Base 2
	1 1 1 1 1 1 1
109 +	0 1 1 0 1 1 0 1 +
147 =	1 0 0 1 0 0 1 1 =
2⁸ = 256	1 0 0 0 0 0 0 0

Esempi complemento a 2 (2)

- $C2[1011001000] = ??, k=10$

2) Si inverte | 1) si copia fino al primo 1 compreso

1 0 1 1 0 0	1 0 0 0
0 1 0 0 1 1	1 0 0 0

- $C2[1 0 1 1 0 0 1 0 0 0] = \mathbf{0 1 0 0 1 1 1 0 0 0}$
- Verifica:

Base 10	Base 2
	1 1 1 1 1 1
712 +	1 0 1 1 0 0 1 0 0 0 +
312 =	0 1 0 0 1 1 1 0 0 0 =
<u>2¹⁰ = 1024</u>	<u>1 0 0 0 0 0 0 0 0 0</u>

Interi con segno (1)

- Un intero con segno rappresenta una variabile dichiarata in C come:
 - `int a;`
- Gli interi con segno sono rappresentati su N bit usando una rappresentazione in complemento a 2
- Consiste nel rappresentare ogni numero negativo attraverso il complemento a 2 del corrispondente numero positivo
- Questa codifica può essere vista come una variante della codifica posizionale in cui:
 - Il bit più significativo (Most Significant Bit, MSB) ha peso negativo
 - Gli altri bit hanno peso positivo, come visto fino ad ora

Interi con segno (2)

Quindi se $k=8 \rightarrow$ l'ottavo bit, quello più significativo (MSB), ovvero quello più a sinistra, rappresenta il segno del numero:

+ se il MSB = 0

- se il MSB = 1

La sequenza di bit $a_{N-1}, a_{N-2}, \dots, a_1, a_0$ codifica il valore:

$$A = -a_{N-1} * 2^{(N-1)} + \sum_{n=0}^{N-2} a_n * 2^n, \text{ con } a_n = \{0, 1\}$$

I valori rappresentati sono i numeri compresi nell'intervallo $[-2^{(N-1)}; 2^{(N-1)}]$

Esempi Interi con segno

Calcolare $-A$, con $A=[110101]_2$

2) Si inverte | 1) si copia fino al primo 1 compreso
1 1 0 1 0 | 1
0 0 1 0 1 | 1 $-A = [001011]_2$

Calcolare $-A$, con $A=[0111111]_2 = [63]_{10}$

2) Si inverte | 1) si copia fino al primo 1 compreso
0 1 1 1 1 1 | 1
1 0 0 0 0 0 | 1 $-A = [1000001]_2$

Interi con segno (3)

- In generale in una operazione di somma può verificarsi una condizione di **overflow**
- L'**overflow** si verifica quando il risultato di una operazione di somma va oltre i valori rappresentati
- Dal punto di vista della programmazione accade che:
 - L'hardware genera una eccezione che però è ignorata
 - L'effetto è che la somma produce un risultato secondo una aritmetica modulare, ovvero se si somma 1 al max valore positivo → si ottiene il minimo valore positivo

Somma nella forma complemento a 2(2)

- Calcoliamo la somma seguente, con 8 bit

Base 10	Base 2
71 +	0 1 0 0 0 1 1 1 +
60 =	0 0 1 1 1 1 0 0 =
<hr/>	<hr/>
131	1 0 0 0 0 0 1 1

Segno

- Ricordiamo che nella forma complemento a 2, **il primo bit è per il segno**, i possibili valori rappresentati stanno fra: $[-2^7; 2^7]$ = $[-128; 128]$

1 1 1 1 1 0 1	$[125]_{10}$
0 0 0 0 0 1 1	$C_2(125)$

- 131 va oltre i possibili valori rappresentati infatti dalla somma si ottiene come risultato la rappresentazione in forma complemento a 2 di -125 e NON di 131
- $C_2[0000011] = 1111101$, con $[1111101]_2 = [125]_{10}$
- Questo è il caso di Overflow
- INOLTRE L'effetto è che la somma produce un risultato secondo una aritmetica modulare, ovvero se si somma 1 al max valore positivo \rightarrow si ottiene il minimo valore positivo INFATTI togliendo il MSB si ottiene: - 0 0 0 0 0 1 1 \rightarrow (128 + 3 = 131)

Somma nella forma complemento a 2(1)

Situazione di carry

Base 10	Base 2
33 +	0 0 1 0 0 0 0 1 +
21 =	0 0 0 1 0 1 0 1 =
54	0 0 1 1 0 1 1 0

8 bit

Base 10	Base 2
- 33 +	1 1 0 1 1 1 1 1 +
- 21 =	1 1 1 0 1 0 1 1 =
54	1 1 1 0 0 1 0 1 0

8 bit

9 bit: si ha un riporto al di fuori del bit più significativo.

Situazione di carry



Fondamenti di Informatica

Eng. Ph.D. Michela Paolucci

DISIT Lab <http://www.disit.dinfo.unifi.it/>

*Department of Information Engineering, DINFO
University of Florence*

Via S. Marta 3, 50139, Firenze, Italy

tel: +39-055-2758515, fax: +39-055-2758570

michela.paolucci@unifi.it



UNIVERSITÀ
DEGLI STUDI
FIRENZE

DINFO
DIPARTIMENTO DI
INGEGNERIA
DELL'INFORMAZIONE

DISIT
DISTRIBUTED SYSTEMS
AND INTERNET
TECHNOLOGIES LAB