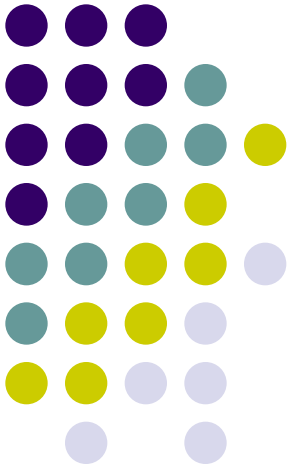
Università di Napoli Federico II - Dip. di Ingegneria Elettrica e delle Tecnologie dell'Informazione
Corso di Laurea in Ingegneria Informatica

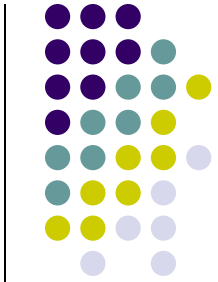
DIE TI

Fondamenti di Informatica

Tipi Semplici



Identificatori



- Identificatori: stringhe di lettere e cifre che iniziano con una lettera, possono contenere anche il carattere ‘_’

data

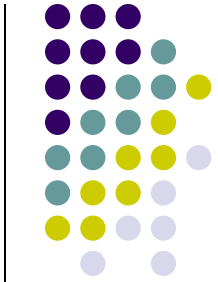
Data

X1, X2

esami_sostenuti

- Maiuscole e minuscole sono diverse
- Identificatori predefiniti

Dichiarazioni di variabili



- In C++, una variabile viene **dichiarata** con la sintassi:

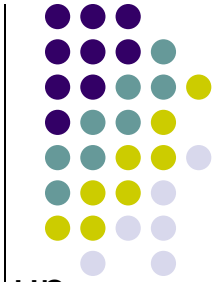
  Tipo <nome_variabile>;

  - <*nome_variabile*> è un nome scelto dal programmatore formato da una qualunque sequenza di lettere (maiuscole o minuscole) e di cifre numeriche, senza altri caratteri o spazi bianchi a parte l'underscore (che è considerato una lettera), che deve cominciare con una lettera;
  - Il C++ è case-sensitive: significa che il compilatore considera le lettere maiuscole e minuscole come caratteri distinti;
  - E' conveniente usare nomi significativi per le variabili;

- Esempi di dichiarazioni

```
int x2;        //x2 è un intero
char c;        //c è un carattere
```

Definizioni di variabili



- In C/C++ una variabile è **definita** quando le è associata un'area di memoria. Il compilatore associa ad ogni variabile con nome (cioè con un identificatore) un indirizzo di memoria RELATIVO (cioè secondo uno schema che non corrisponde alla reale situazione della memoria centrale della macchina su cui il programma verrò eseguito)
- L'indirizzo RELATIVO è una indicazione per il linker ed il caricatore. Sarà il caricatore a trasformare gli indirizzi di memoria relativi in indirizzi ASSOLUTI, cioè negli indirizzi in cui realmente le variabili saranno memorizzate.

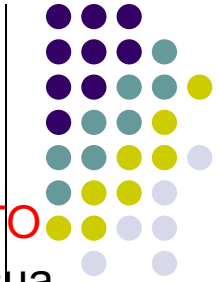
Quindi:

- In C/C++ dichiarazione e definizione sono concetti differenti
  - Dichiarazione: rende noto il nome ed il tipo della variabile
  - Definizione: alloca spazio in memoria

- Dichiarazione e definizione possono essere effettuate contestualmente ma in alcuni casi possono anche essere disgiunte

Una variabile NON può mai essere USATA prima di essere stata DEFINITA

Valore di una variabile



- Una variabile quando viene creata in memoria ha un VALORE INDEFINITO
- Una variabile in C/C++ può essere INIZIALIZZATA contestualmente alla sua definizione (creazione)
- A una variabile poi può essere ASSEGNATO un valore anche dopo la sua definizione (creazione), in questo modo il suo valore viene modificato e quello precedente viene perso (viene fatta una scrittura in memoria e lo stato dei registri di memoria cambia)

- Esempi:

```
extern float f;  //dichiarazione della variabile f
int main() {float f; … } //definizione della variabile f senza inizializzazione
int main() {  float f=3.14; … } //definizione della variabile f con inizializzazione

// ad f viene dato valore 4.2 il precedente valore è perso:

int main() {  float f=3.14;  … f=4.2; }

// f viene definita ma non inizializzata, resta di valore indefinito finchè non le viene
//  asseganto un valore

int main() { float f;  … f=4.2; }
```

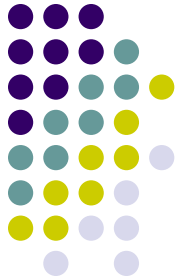
Costanti

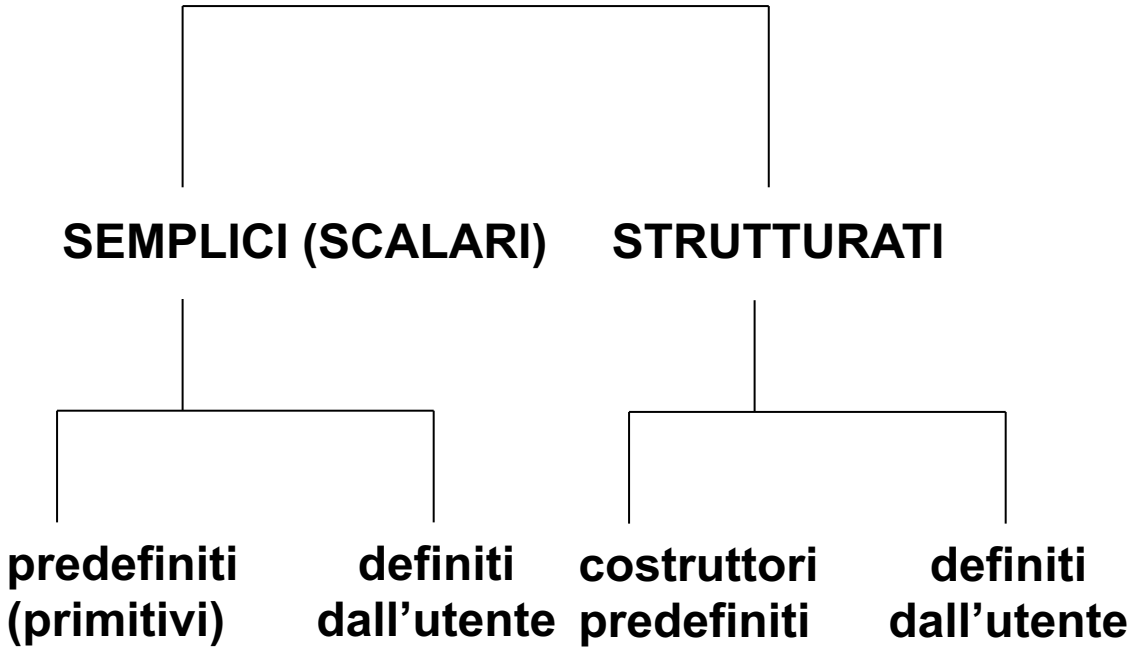
- Il linguaggio consente di assegnare alle costanti un nome da usare al loro posto nel programma per migliorare la leggibilità e la parametricità
- Le costanti esprimono dei valori prefissati non alterabili e possono essere di tipo numerico ed alfanumerico. Esempi:
  - **A+3;**
  - **B+4.5;**
  - **C='a';**
- Le costanti con nome (o tipizzate) consentono di associare un nome ed un tipo esplicito ad una costante, Esempio:

const float x2=1;

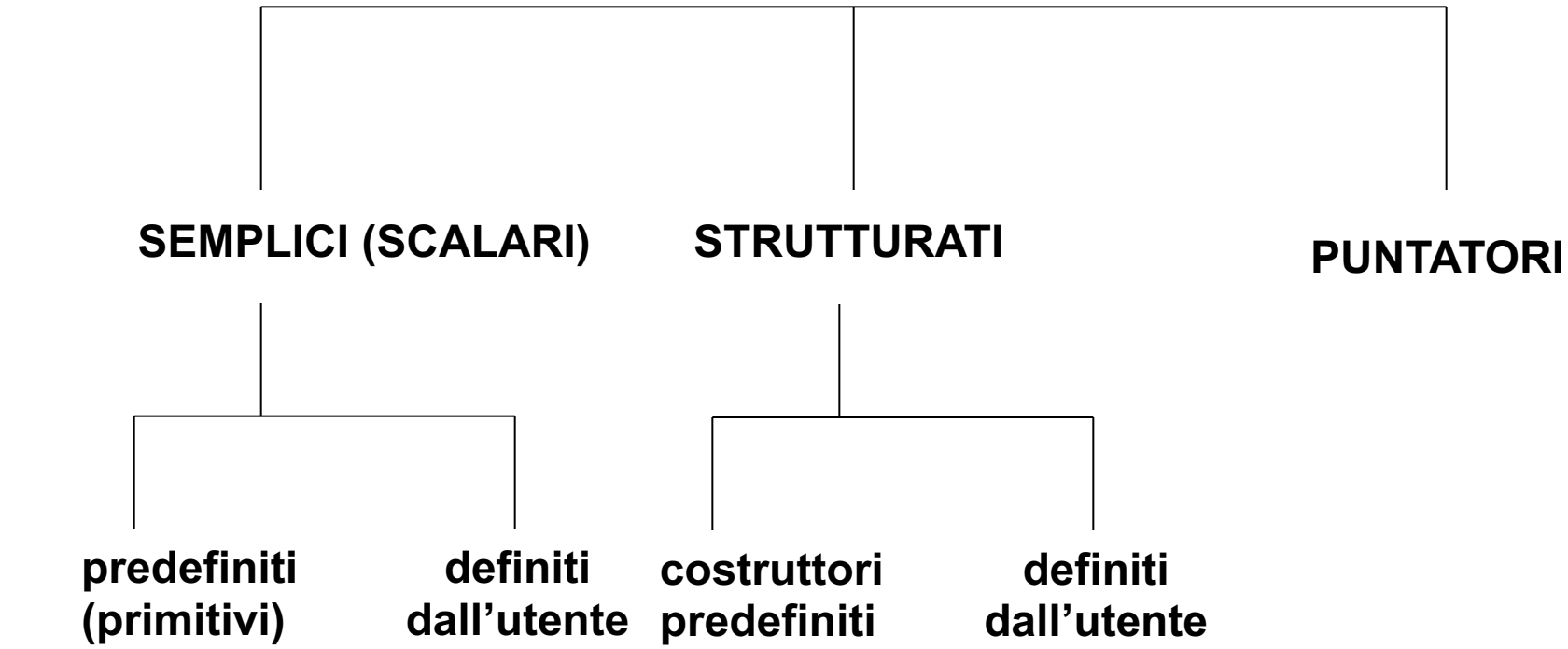
Nota: in questo caso a nome_costante viene attribuito uno spazio di memoria non modificabile

Sistema dei Tipi di un linguaggio di alto livello

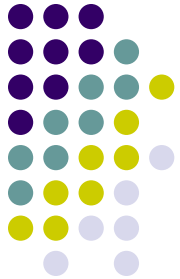




Tipi di Dato in C/C++



Tipi



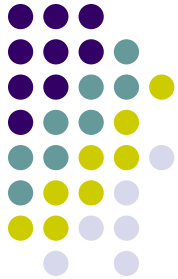
- Semplici
  - I dati sono considerati entità atomiche
    - Es: tipo reale

```
float f;
```

- Strutturati
  - I dati sono composti da dati più semplici
    - Es: Dati Anagrafici

```
struct persona {
    char   name[SIZENAME];
    char   tfnumb[SIZETELE];
    char   addr[SIZEADDR];
    struct persona *ptrnext;
};
```

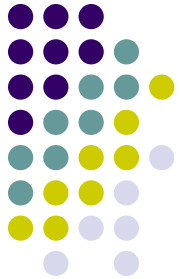
Tipi semplici predefiniti C++



Qualificatoe di accesso	Qualificatore di segno	Qualificatore di lunghezza	Tipi di dati	Uso
const	signed / unsigned	short/ long/ long long	int	Numeri interi o naturali
		long	float	Virgola mobile (IEEE 754 prec. Singola)
			double	Virgola mobile (IEEE 754 prec. doppia)
	signed / unsigned		char	Carattere ASCII
			bool	Valore booleano (logico)

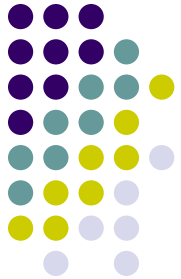
- **Tutti i qualificatori sono opzionali**
- C++ non definisce il numero di byte da usare per ogni tipo, ma dipende dalla rappresentazione (eventualmente standard), dal compilatore e dall'architettura utilizzata: l'istruzione **sizeof(type**) restituisce il numero di byte usati per un tipo.

Tipi semplici definiti dall'utente



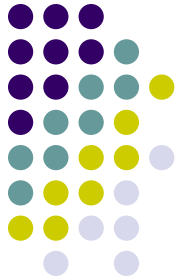
- I linguaggi di alto livello possono mettere a disposizione dei meccanismi che consentono al programmatore di definire dei nuovi tipi
- Tale definizione deve comunque essere supportata dai tipi predefiniti
- I tipi semplici definiti dall'utente sono in generale:
  - Tipi enumerativi
  - Tipi sottocampo

Significato dei qualificatori



- short e long sono *una indicazione* per condizionare lo spazio allocato dal compilatore per la memorizzazione delle variabili del tipo definito
- signed e unsigned condizionano l'uso che si può fare della memoria allocata

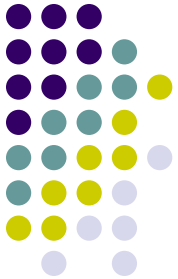
Tipo enumerativo



- È un tipo semplice definito dall'utente
- È intrinsecamente ordinato
- E' possibile modificare il valore (intero) che viene associato implicitamente a ciascun elemento

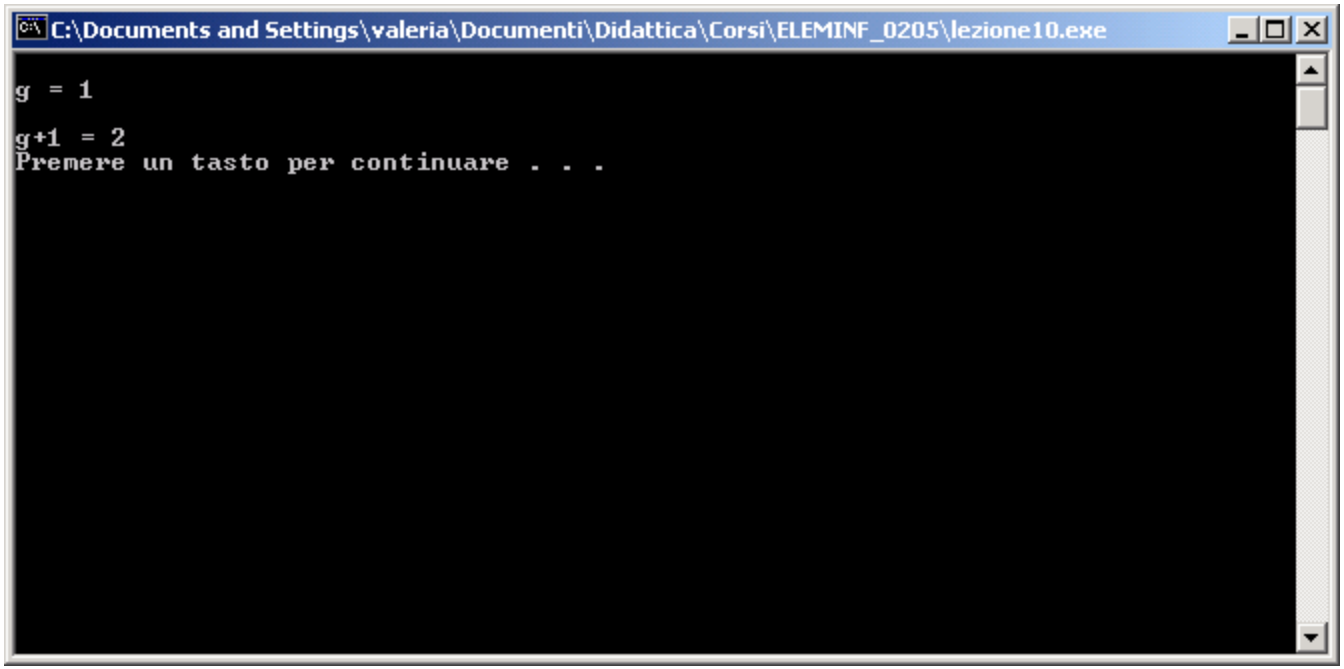
- Esempio:
  - Definizione:
    - typedef enum {
      LUN, MAR, MER, GIO, VEN, SAB, DOM} Giorno;
  - Uso:
    - Giorno g=LUN;

Tipo enumerativo - Esempio

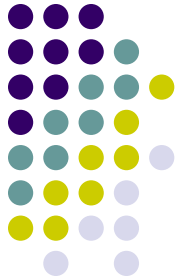


```
int main() {
  typedef enum {LUN, MAR, MER, GIO, VEN, SAB, DOM}Giorno;
  Giorno g=LUN;
  int    i=1;
  g=MAR;
  cout << endl << "g = " << g << endl;
  cout << endl << "g+1 = " << g+1 << endl;
  // cout << endl << "g+9 = " << g+9 << endl; PROVARE!!!
  system("PAUSE");
  return 0;
}
```

Esecuzione


C:\Documents and Settings\valeria\Documenti\Didattica\Corsi\ELEMINF_0205\lezione10.exe
g = 1
g+1 = 2
Premere un tasto per continuare . . .

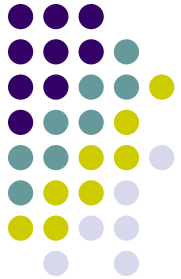
Tipo sottocampo (o intervallo)



- È un tipo (semplice) derivato da un altro tipo, detto originario
- Il tipo originario
  - Deve necessariamente essere un tipo ordinato
  - Può essere tanto un tipo predefinito che un tipo enumerativo

@ Il tipo sottocampo non esiste in C++

Il tipo intero

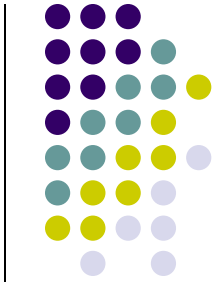


- Il tipo intero fornisce una astrazione in un linguaggio di programmazione dell'insieme dei numeri interi (relativi)
- In pratica non è possibile rappresentare mediante un registro finito un insieme infinito di informazioni:
  - Per rendere finita la cardinalità del tipo in concreto si fissa un intervallo di rappresentazione
  - Indicati con m ed M gli estremi del'intervallo si ha pertanto:

    tipo intero = {i/ i è un numero intero e m<i<M}
  - L'ampiezza dell'intervallo dipende evidentemente dal numero di bit impiegato per rappresentare in macchina il tipo intero

Esiste dunque un numero intero che è il più piccolo numero intero rappresentabile ed un numero intero che à il più grande numero intero rappresentabile

Il tipo intero in C/C++



- Dichiarazione e Definizione di una variabile di tipo intero:

```
int a;
```

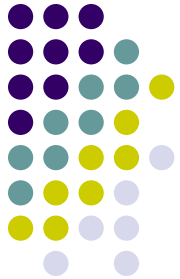
- Inizializzazione di una variabile di tipo intero contestualmente alla sua definizione:

```
int a=10;
```

  - In questo caso all'atto della creazione della variabile, lo spazio di memoria ad esso assegnato verrà inizializzato al valore 10. Pertanto questa istruzione è necessariamente una definizione e non una dichiarazione di variabile
- Costanti numeriche di tipo intero:
  - sequenza di cifre con o senza segno: -1, +5, 238 …
- Definizione di costanti con nome:

```
const int DIM=100;
```

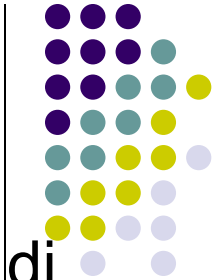
Principali Operazioni applicabili al tipo intero in C/C++



- **Operatori aritmetici** (il risultato di i1 op i2 è un numero intero)
  - addizione i1+i2
  - sottrazione i1-i2
  - moltiplicazione i1*i2
  - divisione intera i1/i2
  - modulo (resto della divisione intera) i1%i2
- **Operatori relazionali** (il risultato di i1 op i2 è un valore booleano)
  - uguale i1==i2
  - diverso i1!=i2
  - maggiore i1>i2
  - minore i1<i2
  - maggiore o uguale i1>=i2
  - minore o uguale i1<=i2
- Vi sono poi funzioni definite mediante librerie, ad esempio:

  abs(i) calcola il valore assoluto dell'intero i

Il tipo reale



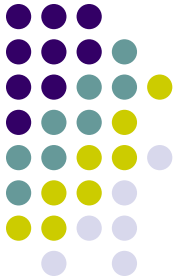
- Il tipo reale fornisce una astrazione in un linguaggio di programmazione dell'insieme dei numeri reali
- Come per i numeri interi si pone il problema che non è possibile rappresentare mediante un registro finito un insieme infinito di informazioni
- Inoltre, poiché il campo dei numeri reali è denso in sé, dati due punti x1 ed x2 sull'asse reale, tra di essi vi sono infiniti numeri reali
- Pertanto non è possibile rappresentare esattamente tutti i numeri nell'intervallo di rappresentazione, e può essere necessario approssimare un numero x per rappresentarlo con il numero "più vicino"

Il tipo reale

- Quindi, fissato un valore ε e dato un l'intervallo di rappresentazione (m,M), l'insieme (finito) dei numeri reali può essere definito come:
- tipo reale = {r/ r è un numero reale che verifica la proprietà β} dove:
  - Proprietà β: per ogni numero reale rs tale che m<=rs<=M, esiste r che lo approssima con assegnata precisione, cioè tale che: |(rs-r)|< ε

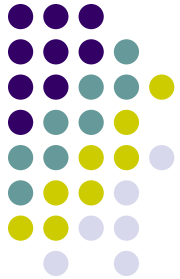
- Per un maggiori informazioni circa la rappresentazione in macchina dei numeri interi e dei numeri reali si rimanda alle relative lezioni

Il tipo reale in C/C++



- Dichiarazione e Definizione di una variabile di tipo reale:
  - float a;
  - double a; // variabile reale in doppia precisione
- Una variabile reale in doppia precisione è rappresentata su un numero maggiore di bit, pertanto il numero reale è rappresentato con maggiore precisione usando il tipo double
- Inizializzazione di una variabile di tipo reale contestualmente alla sua definizione:
  - float a=10.3;
  - Valgono le considerazioni circa l'inizializzazione all'atto della definizione già fatte nella slide relativa ai numeri interi

Costanti di tipo reale in C/C++

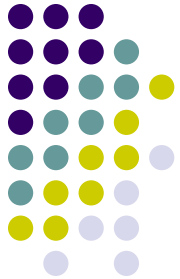


- Costanti numeriche di tipo reale:
  - sequenza di cifre con o senza segno: -1.0, +5.3E+2, 238.25 …
- Definizione di costanti con nome:
  - const float Pgreco=+3,14;
  - const double Epsilon=+1.E-9

Notazione esponenziale $+5,3x10^{+2}$

Notazione esponenziale $+10^{-9}$

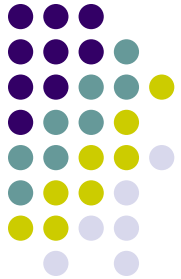
Principali Operazioni applicabili al tipo reale in C/C++



- Operatori aritmetici (il risultato di i1 op i2 è un numero reale)
  - addizione i1+i2
  - sottrazione i1-i2
  - moltiplicazione i1*i2
  - divisione i1/i2
  - Operatori relazionali (il risultato di i1 op i2 è un valore booleano)
    - Sono previsti i principali operatori logici. Bisogna però ricordarsi, nell'usarli, che si sta trattando numeri con precisione fissata.
    - Ad esempio, dette r1 e r2 due variabili di tipo float, ed Espilon una costante di tipo float, sufficientemente vicina a zero, la relazione r1==r2 dovrebbe essere sostituita da:

      abs(r1-r2)<Epsilon
  - Vi sono poi funzioni definite mediante librerie (trigonometriche, logaritmiche, esponenziali, etc…)

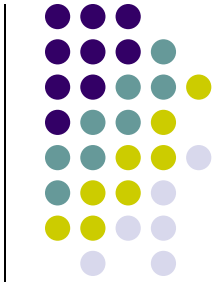
Il tipo carattere



- Il tipo carattere è l'insieme dei simboli che possono essere stampati o visualizzati mediante periferiche, o più in generale riprodotti mediante unità di ingresso/uscita.

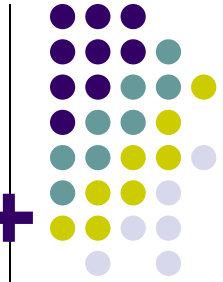
- Comprendono le lettere dell'alfabeto inglese (maiuscole e minuscole), le 10 cifre decimali (da 0 a 9), i caratteri di interpunzione, alcuni simboli matematici e quelli relativi ai principali operatori aritmetiche e relazionali, ed anche dei caratteri speciali e di controllo (ad esempio tabulazione, a capo, etc…)

Il tipo carattere in C/C++



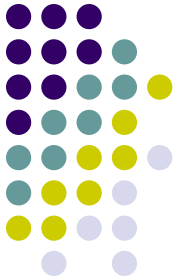
- Dichiarazione e Definizione di una variabile di tipo carattere:
  - char c;
- Inizializzazione di una variabile di tipo carattere contestualmente alla sua definizione:
  - char c= 'a' ;
- Costanti di tipo carattere (notare l'uso del singolo apice prima e dopo il carattere):
  - 'a', 'A' , ' ' (spazio), '1'
- Definizione di costanti con nome:
  - const char CAR= 'A';

Principali Operazioni applicabili al tipo carattere in C/C++



- Dalla sua definizione discende che sul tipo carattere è definito un ordinamento. Pertanto ad esso sono applicabili:
- Operatori relazionali (il risultato di c1 op c2 è un valore booleano)
  - uguale c1==c2
  - diverso c1!=c2
  - maggiore c1>c2
  - minore c1<c2
  - maggiore o uguale c1>=c2
  - minore o uguale c1<=c2
- E' inoltre possibile definire le funzioni "carattere precedente", "carattere successivo", concatenazione etc..,
- Nelle librerie standard sono disponibili numerose funzioni per la manipolazione dei caratteri

Il tipo booleano



- Il tipo booleano contiene solo due elementi:

  tipo booleano = {falso, vero}

- Dichiarazione e Definizione di una variabile di tipo booleano in C++:

  bool trovato;

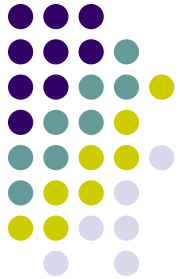
- Inizializzazione di una variabile di tipo booleano contestualmente alla sua definizione:

  bool trovato=**false**;

- Costanti di tipo booleano**: false e true**
- Definizione di costanti on nome:

  const bool OK= **true**;

Principali Operazioni applicabili al tipo booleano in C/C++



- Agli elementi del tipo booleano sono associati dei valori interi. In particolare a false è associato 0 e a true è associato 1
- Sul tipo booleno sono definiti gli operatori della logica boolena, in particolare:
  - la congiunzione AND (in C/C++: &&)
  - la disgiunzione OR (in C/C++: ||)
  - la negazione NOT (in C/C++: !)
- Sono inoltre definiti gli operatori relazionali
- Il risultato della valutazione di un predicato è false se il valore valutato è 0, è vero altrimenti

Riferimenti

- Teoria:
  - cap. 5, §5.1, §5.2, §5.3
- Da «Che C serve»: tutto il Capitolo 2 tranne i paragrafi 2.9 e 2.10