Linguaggio C:

struttura di un programma
variabili e assegnamento
funzioni di input-output

Università degli Studi di Brescia

Docente: Massimiliano Giacomin

Struttura di un programma C

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

main(){
    int x;
    int y, z;

    x=4;
    y=5;
    z=x+y;
    x=x+1;
    x=x+y+z;

    printf("Ciao mondo\n");
    printf("Il valore di x è %d\n", x);
    system("pause");
}
```

Struttura di un programma C

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

main(){
    int x;
    int y, z;

    x=4;
    y=5;
    z=x+y;
    x=x+1;
    x=x+y+z;

    printf("Ciao mondo\n");
    printf("Il valore di x è %d\n", x);
    system("pause");
}
```

Parte dichiarativa globale

Programma principale

Altre funzioni

Struttura di un programma C

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

main(){
    int x;
    int y, z;

    x=4;
    y=5;
    z=x+y;
    x=x+1;
    x=x+y+z;

    printf("Ciao mondo\n");
    printf("Il valore di x è %d\n", x);
    system("pause");
}
```

Parte dichiarativa globale

Parte dichiarativa

Parte esecutiva

Programma principale

Altre funzioni

Parte dichiarativa globale

- Nel nostro caso, conterrà sempre direttive al compilatore e in particolare la direttiva *#include*, per esempio:

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

- Prima che il file sia compilato, viene sostituita dal file *stdio.h*, che include le dichiarazioni delle funzioni della libreria standard di I/O
- In pratica, la direttiva è necessaria per poter usare le “istruzioni” (che in realtà sono funzioni) di ingresso e uscita
- Notare che le direttive non sono seguite dal punto e virgola

Programma principale

- Parte dichiarativa: vengono dichiarate le ***variabili*** (e le costanti) usate dal programma

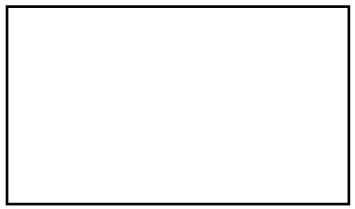
- Parte esecutiva: la sequenza delle ***istruzioni*** del programma
  - le istruzioni sono normalmente eseguite in sequenza
  - ciascuna istruzione termina con un punto e virgola

- Tre tipologie di istruzioni principali:
  - istruzioni ingresso/uscita:
    permettono di acquisire dati e produrre risultati
  - istruzioni aritmetico/logiche
    permettono di svolgere calcoli
  - istruzioni di controllo
    permettono di alterare il flusso sequenziale di esecuzione delle istruzioni

Variabili

- Una variabile rappresenta una “porzione di memoria” per contenere un valore
- Prima di essere usata, una variabile deve essere dichiarata

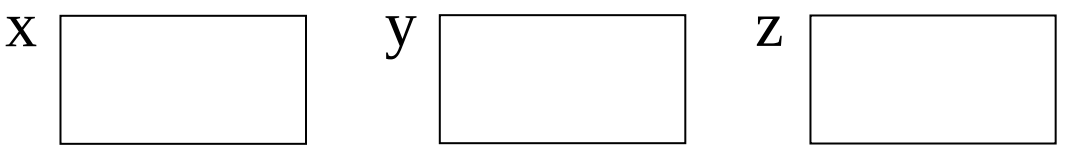
int x x



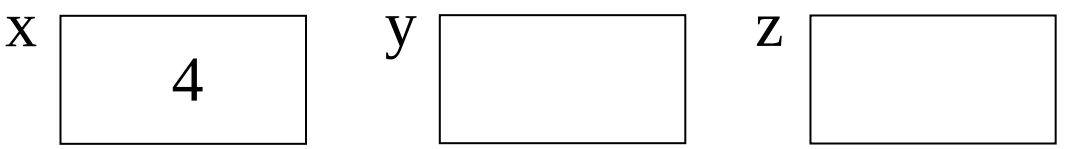
spazio di memoria per contenere un valore intero

- Il valore di una variabile può essere cambiato durante l’esecuzione del programma, mediante un’operazione detta ***assegnamento*** (operatore di assegnamento: =)
- Rivediamo il programma precedente…

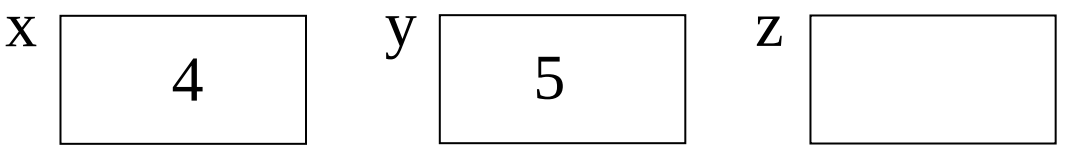
```
#include<stdio.h>
#include <stdlib.h>

main(){
    int x;
    int y, z;

    x=4;
    y=5;
    z=x+y;
    x=x+1;
    x=x+y+z;

    printf("Ciao mondo\n");
    printf("Il valore di x è %d\n", x);
    system("pause");
}
```



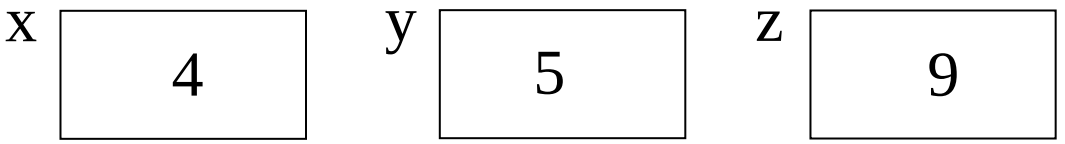
```
#include<stdio.h>
#include <stdlib.h>

main(){
    int x;
    int y, z;

    x=4;
    y=5;
    z=x+y;
    x=x+1;
    x=x+y+z;

    printf("Ciao mondo\n");
    printf("Il valore di x è %d\n", x);
    system("pause");
}
```



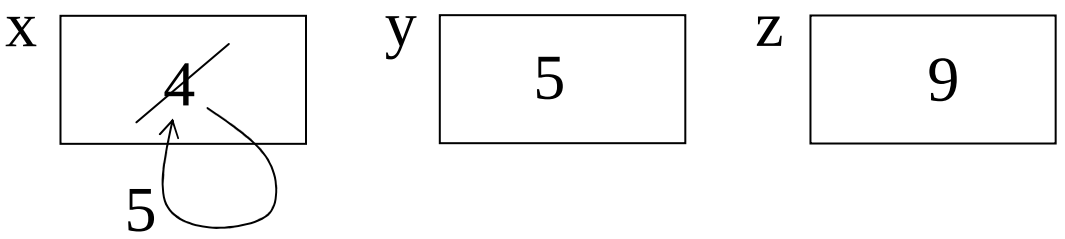
```
#include<stdio.h>
#include <stdlib.h>

main(){
    int x;
    int y, z;

    x=4;
    y=5;
    z=x+y;
    x=x+1;
    x=x+y+z;

    printf("Ciao mondo\n");
    printf("Il valore di x è %d\n", x);
    system("pause");
}
```



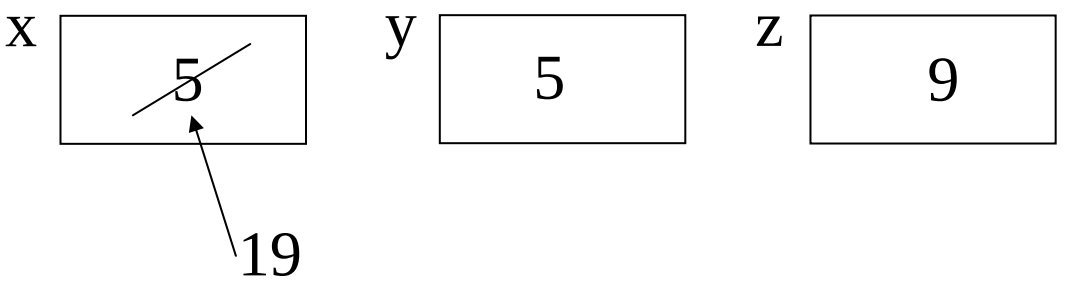
```
#include<stdio.h>
#include <stdlib.h>

main(){
    int x;
    int y, z;

    x=4;
    y=5;
    z=x+y;
    x=x+1;
    x=x+y+z;

    printf("Ciao mondo\n");
    printf("Il valore di x è %d\n", x);
    system("pause");
}
```



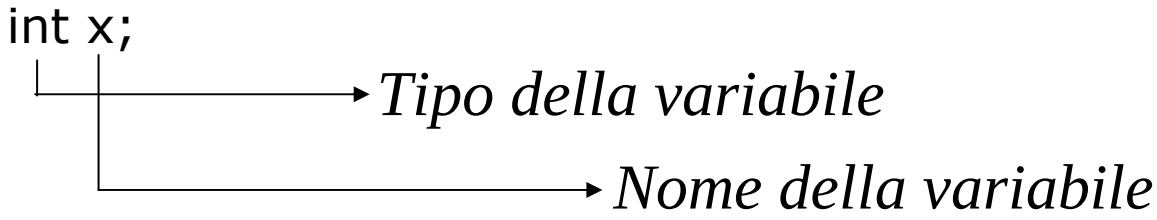
```
#include<stdio.h>
#include <stdlib.h>

main(){
    int x;
    int y, z;

    x=4;
    y=5;
    z=x+y;
    x=x+1;
    x=x+y+z;

    printf("Ciao mondo\n");
    printf("Il valore di x è %d\n", x);
    system("pause");
}
```



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

main(){
    int x;
    int y, z;

    x=4;
    y=5;
    z=x+y;
    x=x+1;
    x=x+y+z;

    printf("Ciao mondo\n");
    printf("Il valore di x è %d\n", x);
    system("pause");
}
```



Dichiarazione di variabili: sintassi



- E' anche possibile dichiarare più variabili dello stesso tipo:

```
int y, z;
```

- E' anche possibile dichiarare variabili e assegnarvi dei valori:

```
int altezza=3;
```

- Negli esempi che faremo la dichiarazione di una variabile coincide con la creazione della variabile stessa

Le caratteristiche di una variabile

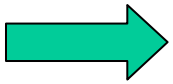
- ***Nome*** e ***tipo*** sono specificati nella dichiarazione
  - > nome: sequenza di caratteri che comincia con un carattere alfabetico, univoco, case sensitive, non può essere una "keyword" (parola riservata del linguaggio C)
  - > tipo: identifica i **valori** che la variabile può assumere e le **operazioni** che su di essa possono essere compiute

    P.es. esistono variabili per memorizzare valori numerici interi, valori numerici reali, dati di una persona, …

Perché si devono dichiarare tutte le variabili prima di usarle?

```
#include<stdio.h>

main(){
    int somma;
    int x, y;

    …
    somme=x+y;
    …
}
```

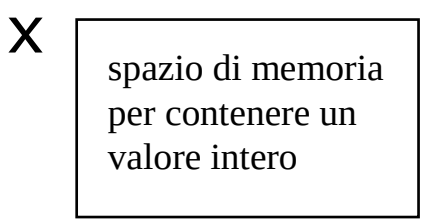
Supponiamo che scriva *somme* al posto di *somma*



In questo caso il compilatore “se ne accorge”, perché *somme* non è stata dichiarata!

Il tipo di una variabile

Es. una variabile di tipo “int”



“operatori” disponibili:
=, +, -, *, ecc. ecc.

Es. una variabile di tipo “studente”

“operatori” disponibili:
=, .

Tipi di dati in C

	Semplici	**Strutturati**
Predefiniti	*char, int, float, double*	-
Definiti dall'utente	*Ridefinizione, enum*	*definiti con: array, struct, pointer*

Tipi di dati semplici: le variabili contengono informazioni "logicamente indivisibili"

Tipi di dati strutturati: informazioni scomponibili in più componenti (es: i giocatori di una squadra di calcio)

Tipi di dati in C

	Semplici	**Strutturati**
Predefiniti	*char, int, float, double*	-
Definiti dall'utente	*Ridefinizione, enum*	*definiti con: array, struct, pointer*

In C i tipi di dati semplici predefiniti (disponibili direttamente) sono tutti numerici

char

8 bit	Valori interi compresi tra -128 e 127

int

di solito 32 bit	Valori interi (compresi tra -2,147,483,648 e +2,147,483,647)

float

di solito 32 bit	Valori razionali (numeri con la virgola)

double

di solito 64 bit	Valori razionali (numeri con la virgola)

Esempio 1

```
…
char a;
a=300;
…
```

in questo caso 300 non rientra nell'intervallo dei valori rappresentabili in un char

Esempio 2

```
…
int a;
a=300;
…
```

in questo caso 300 rientra nell'intervallo dei valori rappresentabili in un int, quindi tutto viene gestito correttamente

Commenti in C

- Si possono inserire commenti per spiegare alcuni dettagli del codice: ovviamente è necessario che vengano "marcati" affinché il compilatore li possa ignorare
- Tutto ciò che è tra /* e */ (anche su più righe) è un commento:

```
/* questo è
un commento */
```

- Quasi tutti i compilatori ammettono anche i commenti nella forma

```
printf("Ciao\n");    //anche questo è un commento
```

tutto ciò che segue // è un commento fino alla fine della riga (in questo caso quindi il commento non è su più righe)

Funzioni della libreria standard di I/O

Torniamo al programma di esempio…

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

main(){
    …
    …
    x=x+y+z;

    printf("Ciao mondo\n");
    printf("Il valore di x è %d\n", x);
    system("pause");
}
```

```
Ciao mondo
Il valore di x è 19
Premere un tasto per
          continuare
```

Funzioni della libreria standard di I/O

Funzione di stampa printf

printf(stringa di controllo, elementi da stampare);

tra apici

lista di variabili, costanti, espressioni separate dalla virgola

Effetto:

- stampa in uscita la stringa di controllo, che contiene caratteri comuni e *caratteri di conversione* preceduti dal simbolo % (ad esempio: %d, %f, %c)
- ad ogni carattere di conversione corrisponde un elemento da stampare nella lista
- un caratteri di conversione provoca la stampa del corrispondente elemento, dopo averlo convertito nel formato di stampa specificato dal carattere di conversione stesso (es: %d per il formato decimale)

Una sequenza speciale

\n rappresenta il carattere di “a capo”

Es.

```
printf(“fuori”);
printf(“gioco\n”);
```

} Stampa “fuorigioco” e va a capo

Caratteri di conversione

%d	notazione decimale
%o	notazione ottale
%x	notazione esadecimale
%c	si considera l’argomento un carattere
%s	si considera l’argomento una stringa (vedremo in seguito)
%f	notazione in virgola fissa (per valori reali)
%e	notazione in virgola mobile
%g	notazione più semplice tra virgola fissa e virgola mobile
.	specifica quante cifre dopo la virgola

Nota sul tipo char

- I caratteri vengono rappresentati come “piccoli numeri interi” (compresi tra 0 e 255)
- In un’operazione di assegnamento, ad esempio

```
char carattere;
carattere = 'a';
```

la variabile assume valore intero corrispondente al codice ASCII, nell’esempio `carattere` risulta pari a 97

- Si possono utilizzare operatori relazionali, ad esempio

`'a' < 'b'` risulta vero

(il codice ASCII mantiene l’ordinamento alfabetico)

- La funzione di stampa che useremo consente poi di specificare il formato di stampa (come carattere o come intero, vedi la differenza tra `%d` e `%c`)

Esempi

```
int g1=3;
int g2=0;
printf("Brescia %d Crotone %d, %s",g1,g2,"Caracciolo");
```

Stampa " Brescia 3 Crotone 0, Caracciolo"

```
float gradi=36.6;
printf("La tua temperatura è di %f gradi", gradi);
```

Stampa "La tua temperatura è di 36.6 gradi"

```
char carattere='a';
printf("Il carattere è %c, ma anche %d\n", carattere,carattere);
```

Stampa "Il carattere è a, ma anche 97"

Funzione di acquisizione dati scanf

scanf(stringa di controllo, elementi da acquisire);

stringa di controllo	elementi da acquisire
tra apici contiene solo caratteri di conversione	lista di variabili separate dalla virgola, ciascuna preceduta dal simbolo &

Es.

```
scanf("%d", &valore);
```

- La funzione scanf legge caratteri dalla tastiera e li interpreta secondo il formato specificato dai caratteri di conversione
- Ciascun valore così ottenuto è immagazzinato in un argomento della lista
- Gli argomenti della lista indicano in realtà indirizzi di memoria in cui memorizzare i valori acquisiti (per questo si usa &)

Caratteri di conversione

%d	si aspetta di acquisire un intero decimale
%o	si aspetta di acquisire un intero ottale
%x	si aspetta di acquisire un intero esadecimale
%h	si aspetta di acquisire un intero short
%c	si aspetta di acquisire un carattere
%s	si aspetta di acquisire una stringa (vedremo in seguito)
%f	si aspetta di acquisire un valore reale

Esempio

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

main(){
    int primo, secondo;
    int somma;

    printf("Inserisci il primo numero\n");
    scanf("%d",&primo);
    printf("Inserisci il secondo numero\n");
    scanf("%d",&secondo);
    somma=primo+secondo;
    printf("Somma uguale a %d\n",somma);
    system("pause");
}
```

Funzione system

- Fa parte della libreria ***stdlib*** (ecco perché abbiamo incluso stdlib.h)
- E' una funzione che esegue comandi del sistema operativo
- Ad esempio, la chiamata system("pause") causa l'esecuzione del programma pause
- Questo programma stampa a video
  "Premere un tasto per continuare"
  e aspetta che l'utente digiti un tasto prima di continuare
- Noi lo utilizziamo per fare in modo che la finestra di esecuzione non scompaia prima che l'utente abbia potuto vedere il risultato dell'esecuzione del programma eseguibile!