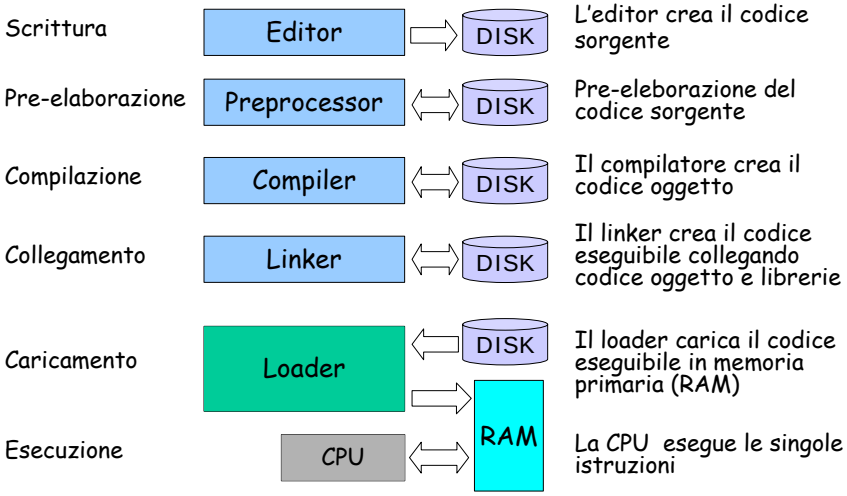
# Laboratorio di informatica

## Ingegneria meccanica

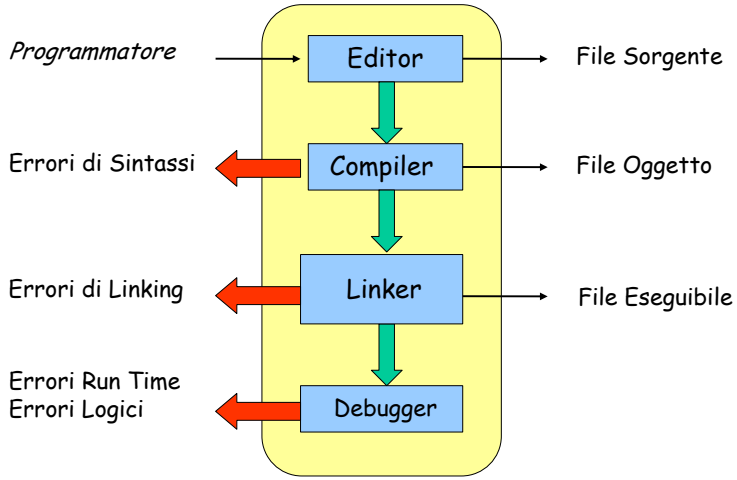
Tutor: Andrea Bernardini
Email: a.berna@fub.it

Esercitazione 1 – 3 Ottobre 2007

## Ciclo di vita di un programma




## Ambiente di Sviluppo




## La funzione main

- Un programma C deve contenere sempre una funzione speciale, detta **main** che indica dove inizia il programma
- Le istruzioni specificate in una funzione sono eseguite quando la funzione viene attivata
- La funzione **main** è attivata dal sistema operativo quando si richiede l'esecuzione del programma.
- Le altre funzioni sono attivate "dall'interno" del programma, mediante un'istruzione di chiamata di funzione

Funzione Main


int CalcolaArea (char *vett, int dim, int * pos)


## Stampa di una linea di testo (1)

```
/* Primo programma in C */
#include <stdio.h>
#include <stdlib.h>
   int main()
   {
     printf( "Primo programma in C!\n" );
      system( "PAUSE" );
      return 0;
   }
```

```
Primo programma in C!
Premere un tasto per continuare...
```

- Il testo compreso tra /* e */ è un *commento* ed è ignorato dal compilatore

## Stampa di una linea di testo (2)

`#include` direttiva per il pre-processore: specifica di caricare il contenuto di un file

`<stdio.h>` contiene la dichiarazione di `printf( )` e delle funzioni della libreria standard del *C* per input/ output

`<stdlib.h>` contiene la dichiarazione di `system( )` e di altre funzioni della libreria standard del *C*

Un programma *C* consiste di moduli o funzioni

- funzioni create dal programmatore stesso
- funzioni della libreria standard del *C*

Le funzioni della libreria sono progettate con grande attenzione all'efficienza

# Stampare di una linea di testo (3)

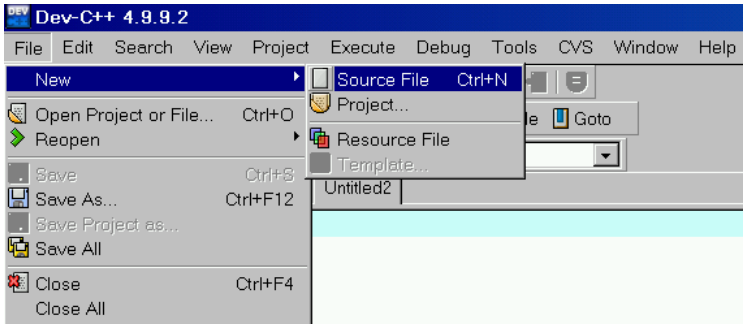
- `int main()`
  - I programmi in C contengono una o più funzioni, una di queste **deve** essere `main()`
  - I nomi di funzioni sono seguiti da parentesi
  - `int` indica che la funzione *restituisce* un valore intero (nel caso del main, questo valore è restituito al sistema operativo, che normalmente interpreta "0" come "esecuzione corretta")
  - Le parentesi graffe `{}` delimitano il *corpo* della funzione, ovvero l'insieme delle istruzioni eseguite alla sua chiamata

# Stampa di una linea di testo (4)

- `printf( "Primo programma in C!\n" );`
  - Istruzione che produce la stampa (visualizzazione su schermo) della frase: **`Primo programma in C!`**
  - Termina, come ogni istruzione, con un punto e virgola
  - **`\n`** specifica che la prossima stampa dovrà essere eseguita su una nuova linea
- `system( "PAUSE" );`
  - Blocca il sistema fino a che non viene premuto un tasto (permettendoci di vedere l'output per il tempo desiderato)
  - L' interpretazione di questa istruzione **dipende** dall'ambiente in cui il programma viene eseguito
- `return 0;`
  - Specifica il valore che la funzione deve restituire

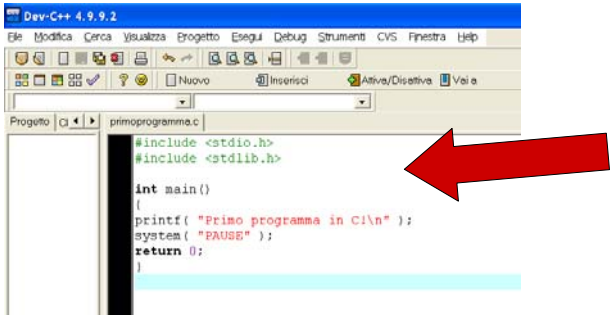
# DevCpp: creare nuovo file

- selezionare `FILE->NEW->SOURCE`



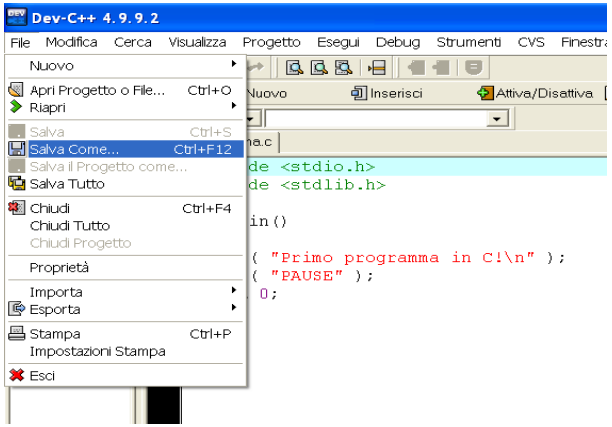

# DevCpp: editor

- scrivere il programma



# DevCpp: salvare il file

- salvare il programma




# Salvare il programma

- Durante la scrittura del file sorgente si deve trasferire il file da memoria volatile (memoria principale) a memoria non volatile (disco)
- Due concetti fondamentali:
  1. Posizione del file (nella memoria secondaria su cui viene salvato)

     `A:\` **(floppy)**

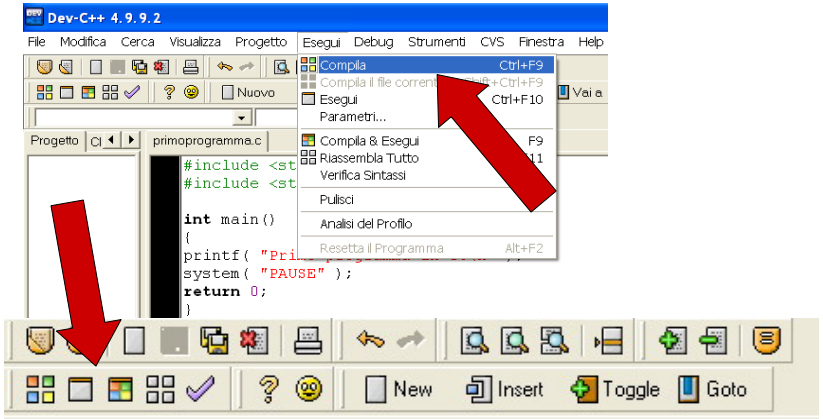
     `C:\cartella\` **(hard-disk)**
  2. Nome del file (composto normalmente da due parti separate da un punto)
     - Il nome
     - L'estensione: caratterizza il tipo di dati contenuti nel file, **`pippo.c`** è un file contenente codice C.

  Esempio:

  **`pippo.c mscalc.exeautoexec.bat`**

# DevCpp: compilare

- compilare ed eseguire il programma




# La nozione di variabile

- Le variabili sono un'astrazione del concetto di cella di memoria
- Una variabile è caratterizzata (principalmente) da:
  - nome logico (deve iniziare con una lettera o "underscore" _ e non deve essere una parola chiave!)
  - valore
  - tipo: Specifica le proprietà della variabile, in termini di valori che può assumere e di operazioni ammissibili sulla variabile. Il tipo consente di verificare in fase di compilazione la correttezza dell'uso della variabile
- Prima di essere usata una variabile deve essere dichiarata
- Una variabile assume un valore a seguito di un'istruzione di assegnamento

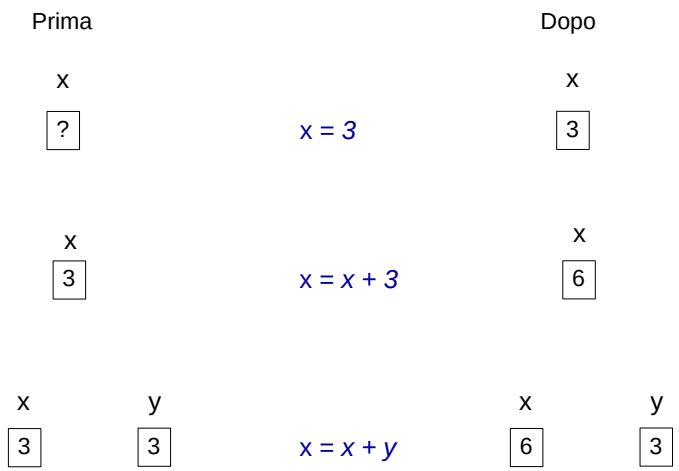
# L'istruzione di assegnamento

- La forma generale (sintassi) di un assegnamento è
  <variabile>=<espressione>
- Il significato è: "valuta il valore di <espressione> e poni il risultato nella cella di memoria individuata dall' indirizzo di <variabilie>"
- Il termine <espressione> può essere:
  - Un' espressione semplice
    - una costante
    - il valore di una variabile
    - il valore restituito da una funzione
  - Un'espressione composta
    - composizione di espressioni mediante operatori

# Operatori aritmetici

| Operazione | Operatore C | Esempio |
|---|---|---|
| Inversione segno | - | -15 |
| Addizione | + | 12+2<br>12.0 + 2 |
| Sottrazione | - | 12 - 4<br>14.5 – 4 |
| Moltiplicazione | * | 4*2 |
| Divisione | / | 7/2 (=3)<br>7.0/2 (=3.5) |
| Modulo (resto) | % | 7%3 |

# Esempi




# Area di un rettangolo

```
1  /* Area di un rettangolo */
2  #include <stdlib.h>
3  #include <stdio.h>
4
5  int main()
6  {
7     int base, altezza, area;        /* dichiarazione */
8
9     printf( "\nInserisci la base\n" );
10    scanf( "%d", &base );           /* leggi un intero */
11    printf( "\nInserisci l'altezza\n" );
12    scanf( "%d", &altezza );          /* leggi un intero */
13    area = base*altezza;          /* calcolo dell'area */
14    printf( "\nL'area e' %d\n", area );       /* stampa area */
15    system("PAUSE");
16    return 0;  /* indica che il programma è terminato con successo */
17 }
```

1. Dichiarazione variabili
2. Input
3. Calcolo Area
4. Print

Program Output

```
Inserisci la base
45
Inserisci l'altezza
72
L'area è 3240
```

## Area di un rettangolo (1)

- **`int base, altezza, area;`**
  - Dichiarazione delle variabili
    - Le variabili devono essere dichiarate prima di essere utilizzate
    - In questo caso si definiscono le tre variabili come intere
  - Diversi tipi di variabili, ad esempio
    - Interi **`int`**
    - Reali **`float`**
    - Caratteri **`char`**

## Area di un rettangolo (2)

- **`scanf("%d", &base);`**
  - Legge in ingresso il valore della base
  - **%d** specifica che il valore in ingresso è un intero (per i reali si usa %f, per i caratteri %c)
  - **&** è **importante**, il motivo sarà chiaro in seguito!
  - L'utente risponde allo **`scanf`** digitando il numero e premendo **ENTER**

## Area di un rettangolo (3)

**`area = base*altezza;`**
- Assegnamento di un valore ad una variabile

```
a = 5;
b = a + 3;
g = f/2;
```

- Alla variabile viene assegnato il valore a destra

**`printf("\nL'area è %d\n", area);`**
- Stampa in output:
  `L'area è` **"valore della variabile area"**
- **%d** specifica che il valore in output è un intero (per i reali si usa %f, per i caratteri %c)

## Esercizi

1 Scrivere un programma che chieda all'utente di immettere due numeri interi, ottenga i numeri dall'utente e visualizzi la loro somma, prodotto, differenza, quoziente e modulo

2 Scrivere un programma che chieda all'utente di immettere un numero reale che rappresenta il raggio di un cerchio e visualizzi il diametro, la circonferenza e l'area dello stesso (usare il valore 3.14159 per π)

3 Scrivere un programma che chieda all'utente di inserire un carattere e stampi 3 occorrenze del carattere letto separate da spazio