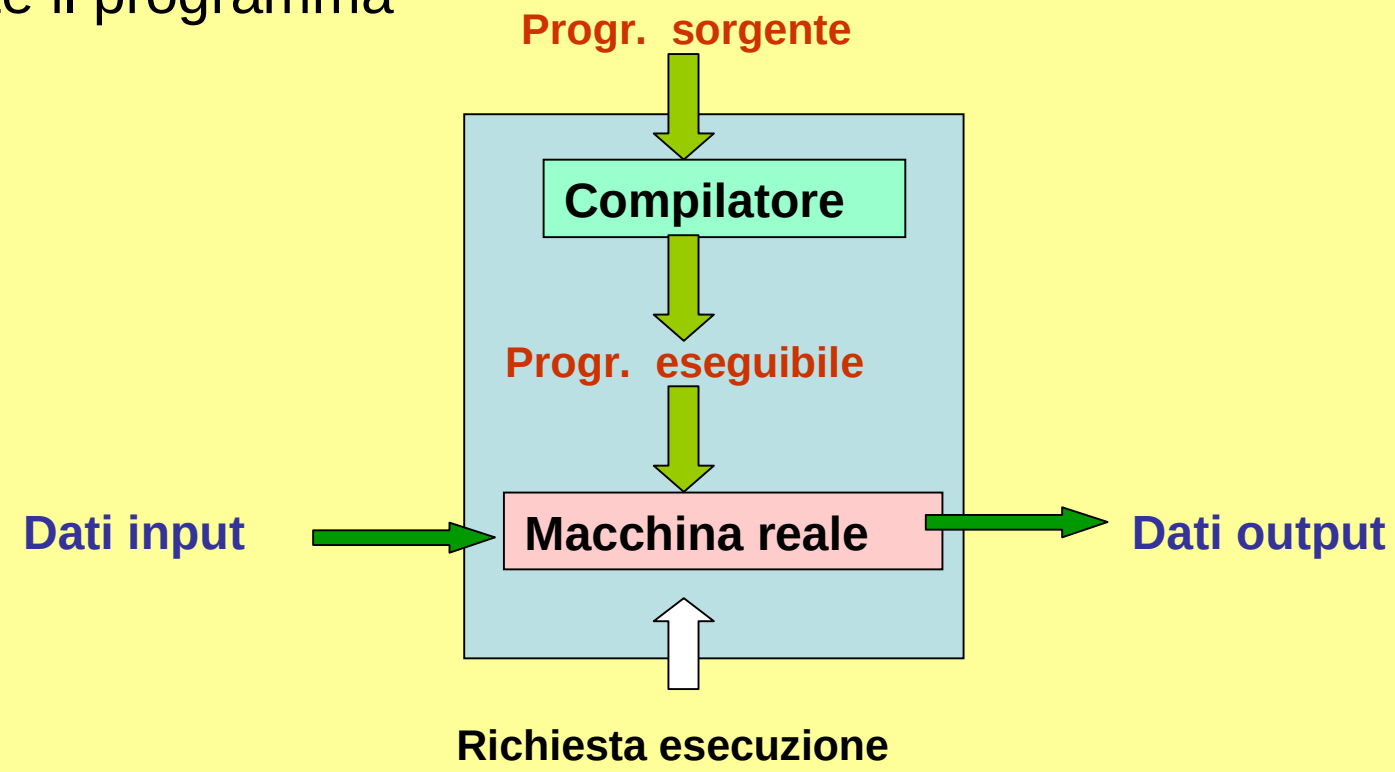
Un esecutore di un linguaggio simbolico e' costituito dalla coppia **Compilatore, processore** (o Interprete, processore)

Macchina astratta: un linguaggio di programmazione trasforma un calcolatore in una *macchina astratta*, ossia le caratteristiche e le funzionalità di tale macchina sono determinate dalle funzionalità del linguaggio che si esprimono tramite il programma



Ogni programma, in un qualsiasi linguaggio di programmazione sia scritto, prima di essere eseguito viene sottoposto ad un processo di ***compilazione*** o ***interpretazione*** (a seconda che si usi un compilatore o un interprete).

Lo scopo e` di tradurre il programma originale (*codice sorgente*) in uno semanticamente equivalente, ma eseguibile su una certa macchina.

Il processo di compilazione e` suddiviso in piu` fasi, ciascuna delle quali volta all'acquisizione di opportune informazioni necessarie alla fase successiva.

La prima di queste fasi e` nota come ***analisi lessicale*** ed ha il compito di riconoscere gli elementi costitutivi del linguaggio sorgente, individuandone anche la categoria lessicale.

Elementi lessicali del linguaggio C

- Commenti;
- Identificatori;
- Parole riservate
- Costanti letterali
- Segni di punteggiatura e operatori

I commenti

Hanno valore soltanto per il programmatore; vengono ignorati dal compilatore.

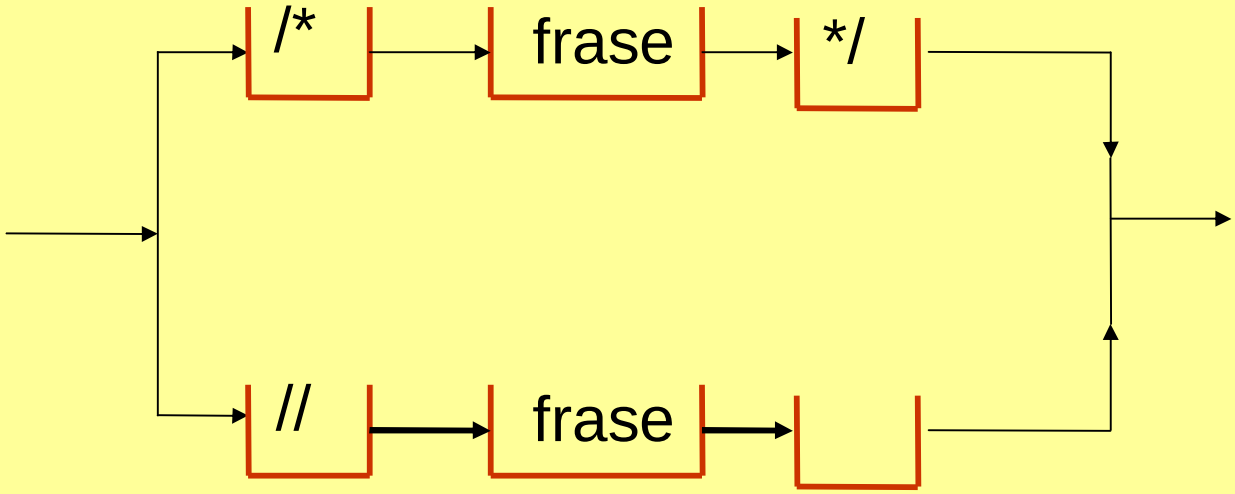
E` possibile inserirli nel proprio codice in due modi diversi:

- racchiudendoli tra i simboli /* e */
- facendoli precedere dal simbolo //

Nel primo caso e` considerato commento tutto quello che e` compreso tra /* e */ .
(il commento quindi si estende su piu` righe)

Le carte sintattiche

Rappresentazione grafica delle regole sintattiche della sintassi di un linguaggio



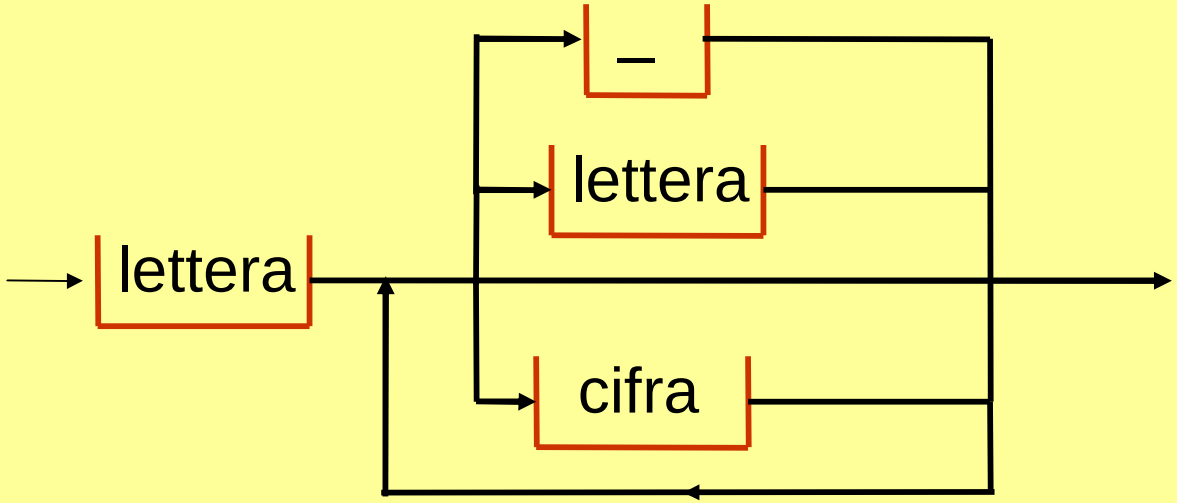
Carta sintattica per frasi di commento

Identificatori

- Devono iniziare con un carattere alfabetico
- Utilizzabili lettere, cifre e carattere “_”
- Possono essere di lunghezza arbitraria (non sempre)
- Non utilizzabili nomi delle paroli chiavi
- Significativi caratteri maiuscoli e minuscoli (*case sensitive)*
- Non utilizzabile il carattere “spazio”

pi_greco alpha1 eq2grado

~~2alpha~~ ~~ax?xyeq~~ ~~2grado~~



Carta sintattica per identificatori

Identificatori di:

Variabili
Costanti
Etichette
Tipi definiti dal programmatore
Funzioni
Parole riservate

Le ***variabili***

Contenitori di valori (celle di memoria)
Ogni variabile puo` contenere un singolo valore che puo` cambiare nel corso del processo elaborativo
Il tipo di valore contenibile nella cella viene stabilito una volta per tutte e non puo` cambiare

Le attività sulle variabili:

Definizione tramite identificatore e tipo del valore

```
int  a , b
```

Scrittura valore (assegnazione)

```
a= 1;
```

Lettura valore (uso)

```
b = a+1
```

Una variabile non può mai essere USATA
prima di essere DEFINITA

Le ***costanti***

Identificano valori che non cambiano nel tempo

Definizione tramite identificatore e tipo

```
const  int  numero
const char carattere
```

Scrittura costante (assegnazione)

```
numero  = -103;
carattere= ‘!’
parola=“dizionario”
```

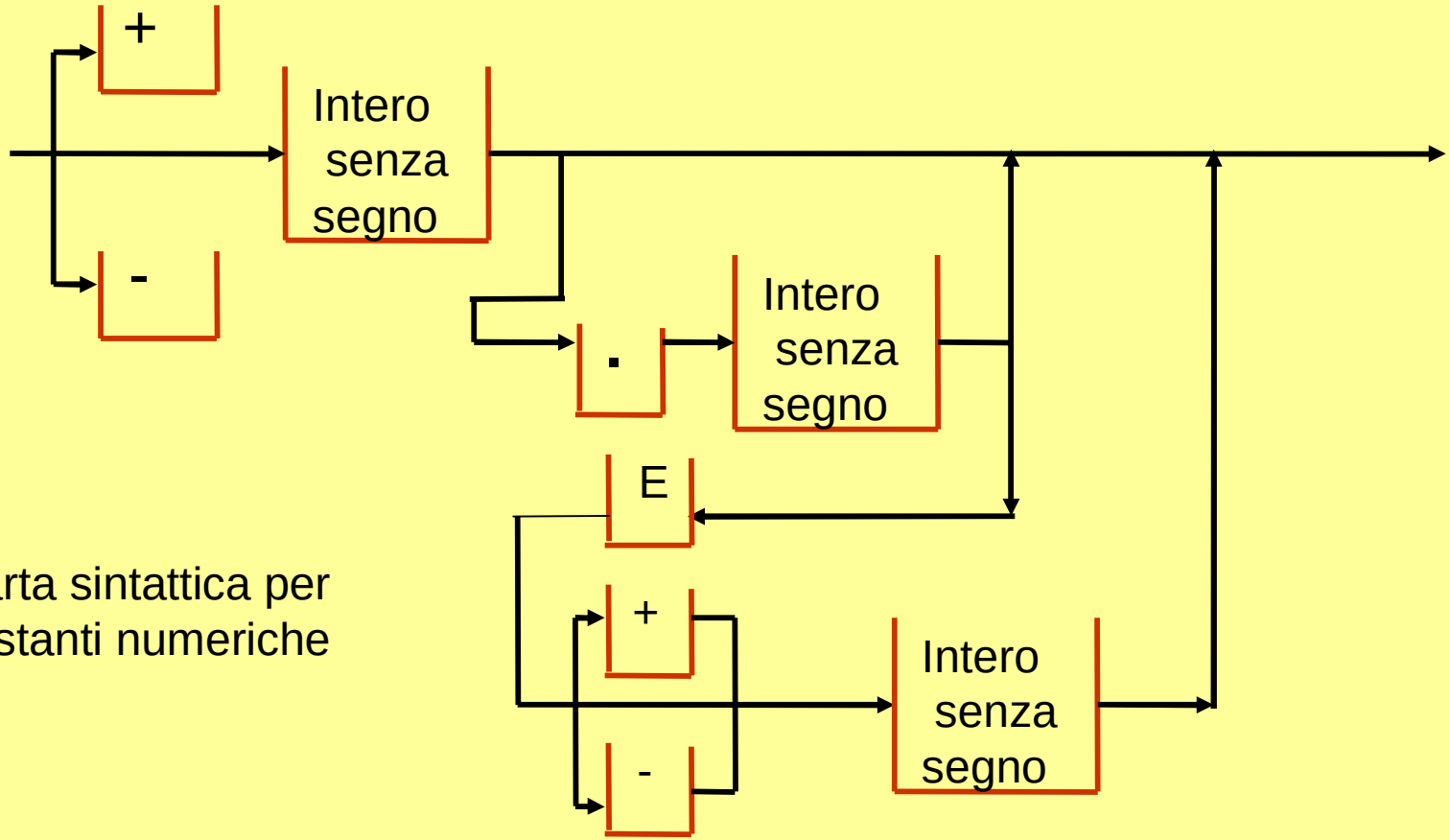
Costante numerica

Costante carattere

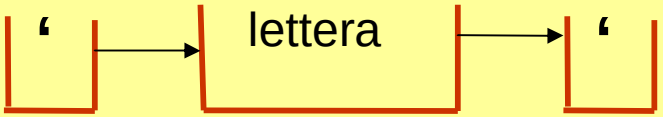
Costante stringa

Possono essere anche espressioni composte

```
a= 1023 + ( - 254)
```



Carta sintattica per
costanti numeriche



Carta sintattica per
costanti carattere

Le ***etichette***

nomi che identificano istruzioni del programma
utilizzati dall'istruzione di salto incondizionato ***goto***

I ***tipi***

➢Un tipo e` una coppia **< V, O >**, dove **V** e` un insieme di valori e **O** e` un insieme di operazioni per la creazione e la manipolazione di elementi di **V**.

ogni linguaggio dispone di tipi **primitivi** (cui e` associato un identificatore predefinito ***int, char,…)***
e dei meccanismi ***costruttori*** per permettere la costruzione di nuovi tipi (**tipi di utente**) definiti tramite identificatore di utente

Tipi primitivi

- **bool**
- **char**
- **int**
- **float**
- **double**

Tipi di utente

- **array**
- **struttura**
- **unione**
- **enumerativo**
- **classe**

Esempio : Costruttore tipo array

tipo identificatore [cardinalità]

Int vettore[10]

Segni di punteggiatura e operatori

Alcuni simboli sono utilizzati dal C++ per separare i vari elementi sintattici o lessicali di un programma o come operatori per costruire e manipolare espressioni

] ( ) { } + - * % ! ^ & =

| \ ; ' : “ < > ? , . ++ --

.* ->* << >> <= >= == != &&

+= -= *= <<= /= %= &= ^= |=

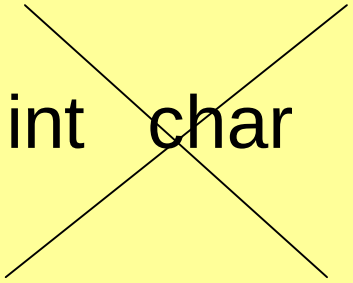
>>=

Le funzioni

Sono i sottoprogrammi

Le parole riservate

Il linguaggio si riserva delle parole chiave (***keywords***) il cui significato e` prestabilito e che non possono essere utilizzate dal programmatore come identificatori



Parole riservate (Keywords)

Utilizzo di sole lettere minuscole

asm	dynamic_cast	new	template
auto	else	operator	this
bool	enum	private	trow
break	explicit	protected	true
case	extern	public	try
catch	false	register	typedef
char	float	reinterpret_cast	typeid
class	for	return	typename
const	goto	short	union
continue	if	signed	unsigned
const_cast	inline	sizeof	using
default	int	static	virtual
delete	long	static_cast	void
do	mutable	struct	while
double	namespace	switch	

Il programma,scritto in linguaggio ***simbolico***,è una sequenza di :

- frasi ***dichiarative*** o dichiarazioni

  int x,y

- frasi ***esecutive*** o istruzioni
  - di ingresso o di uscita
  - di assegnazione e calcolo
  - di chiamata a sottoprogramma
  - di controllo sequenza

Istruzioni di assegnazione e calcolo

Identificatore = ***espressione***

X = 5
Y = 7
Z = X + Y + (X *4)

Una espressione determina un valore di un certo tipo (*tipo dell'espressione*) che deve essere congruente con il tipo della variabile di assegnazione
Gli operatori utilizzati nella espressione devono essere congruenti con i tipi delle variabili utilizzate

int x,y
char a

~~x =a~~

~~x =a * y~~

Istruzioni di ingresso

consentono, al momento della esecuzione, il trasferimento di dati da un supporto periferico alla memoria centrale

Nel linguaggio C++ lo standard input è la tastiera

cin >> *variabile*

cin >> X

cin >> X + Y

La correttezza di una operazione di input dipende dalla corrispondenza tra valore che si acquisisce ed il tipo specificato per la variabile

Istruzioni di uscita

consentono, al momento della esecuzione, il trasferimento di dati dalla memoria centrale ad un supporto periferico

Nel linguaggio C++ lo standard output è il video

cout << *espressione*

cout << X

cout << X + Y

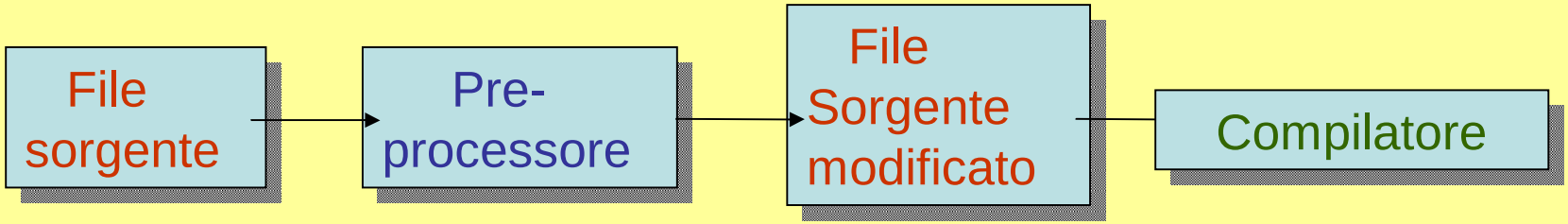
La correttezza di una operazione di input dipende dalla corrispondenza tra valore che si acquisisce ed il tipo specificato per la variabile

La struttura del programma

```
/*    eventuale testata di commento */
#     direttive per il preprocessore
int  main ( )
{
     dichiarazioni;
     frasi;
system (“PAUSE”);
return EXIT_SUCCESS;
}
```

Un **preprocessore** o **precompilatore** è un programma che effettua , preliminarmente, sostituzioni testuali sul *codice sorgente* di un programma, prima che quest'ultimo venga consegnato al *compilatore* vero e proprio

Le istruzioni inserite nel codice sorgente per essere elaborate dal preprocessore sono solitamente dette "direttive“ e si caratterizzano per il simbolo iniziale #



- Le direttive non sono istruzioni del linguaggio C++ e non ne hanno la sintassi

- Le direttive vengono soppresse una volta elaborate dal Preprocesore (il compilatore non le vede)

Una prima direttiva include

#include <nome dell' header file>

Nome della libreria

#include <iostream>

Un header file del linguaggio C è un semplice file di testo che contiene i prototipi delle funzioni definite nel relativo file *.c*.

Il concetto di prototipo sarà chiarito in seguito

```
/*    Presentazione struttura */
#include <iostream>
using namespace std;
int  main ( )
{
     dichiarazioni;
     frasi;
system (“PAUSE”);
return EXIT-SUCCESS;
}
```

Include nel programma, le funzionalità per l'utilizzo degli ***stream*** (flussi) di input e output.(Utilizzo di cin e cout)

Il linguaggio C++ fornisce delle librerie già pronte contenenti funzionalità riguardanti elaborazioni varie (math,…)

Esercizio 0

Date tre variabili intere a, b, c provvedere a determinare la loro somma

Input a = 5 b = 7 c= 9

Output somma=21

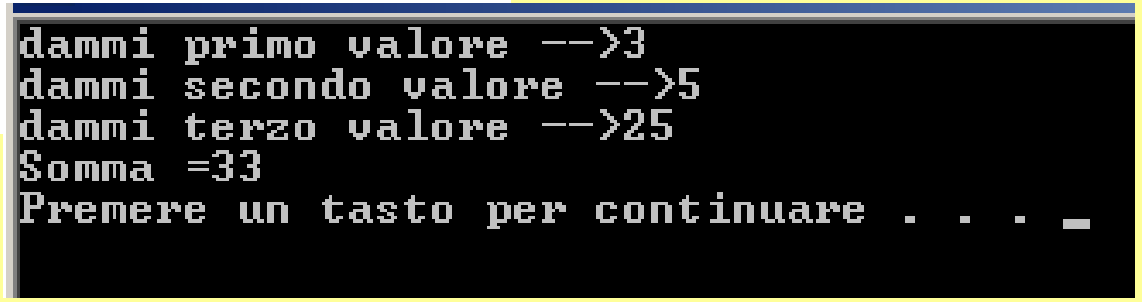
Informazioni di ingresso

a	Var. semplice intera
b	Var. semplice intera

Informazioni di uscita

somma	Var. semplice intera

```
# include <iostream.h>
using namespace std;
//- Somma di tre variabili a, b , c intere
int main ()
{
     int a, b, c, somma;
     cout << "dammi primo valore -->";
     cin >> a;
     cout << "dammi secondo valore -->";
     cin >> b;
     cout << "dammi terzo valore -->";
     cin >> c;
     somma=a+b+c;
     cout << "Somma =";
     cout << somma<<endl;
     system ("PAUSE");
     return 0;
}
```



Volendo produrre anche la media aritmetica dei 3 valori?

Informazioni di uscita

somma	Var. semplice intera
media	Var. semplice ?????

```
# include <iostream.h>
using namespace std;
//- Media di tre variabili a, b , c intere
int main ()
{
     int a, b, c, somma;
     float media;
     cout << "dammi primo valore -->";
     cin >> a;
     cout << "dammi secondo valore -->";
     cin >> b;
     cout << "dammi terzo valore -->";
     cin >> c;
     somma=a+b+c;
     cout << "Somma =";
     cout << somma<<endl;
     cout << "Media ="<<somma/3.;
     system ("PAUSE");
     return 0;
}
```

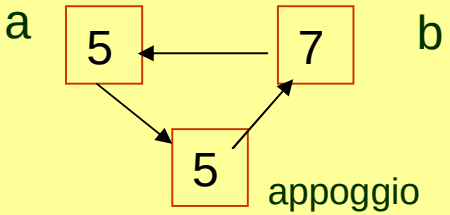
```
dammi primo valore -->1
dammi secondo valore -->3
dammi terzo valore -->7
Somma =11
Media =3.66667Premere un tasto per continuare . . .
```

Esercizio 2

Date due variabili A e B provvedere a scambiare i loro valori

Input a = 5 b = 7

Output a = 7 b = 5



Informazioni di ingresso

a	Var. semplice intera
b	Var. semplice intera

Informazioni di algoritmo

appoggio	Var. semplice intera

Informazioni di uscita

a	Var. semplice intera
b	Var. semplice intera

```
#include <iostream>
using namespace std;
// Assegnati 2 variabili intere a e b, provvedere
// allo scambio dei rispettivi valori
int main()
{
    int a,b,appoggio;
    cout << " valore di a -->";
    cin >> a;
    cout << " valore di b -->";
    cin >> b;
    appoggio=a;
    a=b;
    b=a appoggio;
    cout << "valore di a -->" <<a;
    cout << "valore di b -->" <<b;
    system("PAUSE");
}
```

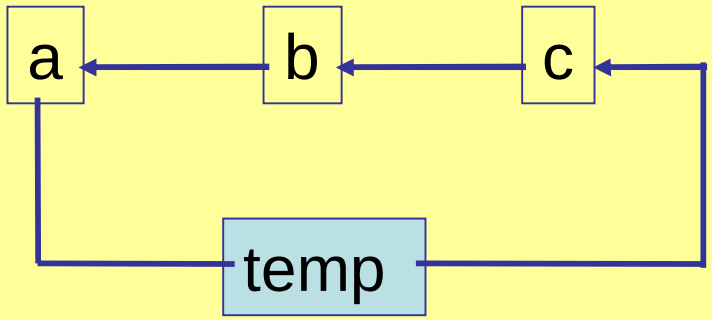
Esercizio 1

Assegnate 3 variabili reali:

A= 3.5 B=7.2 C= 6.1

Si provveda allo scambio circolare dei loro valori

a←b b←c c←a



```
#include <cstdlib>
#include <iostream>
using namespace std;
// Esercizio 1  Scambio circolare fra tre variabili
int main()
{
   float a,b,c,temp;
   cout << "valore a=" ;  cin >>a;
   cout << "valore b=" ;  cin >>b;
   cout << "valore c=" ;  cin >>c;
   temp=a;
   a=b;
   b=c;
   c=temp;
   cout << "Valori dopo lo scambio"  <<endl  <<endl;
   cout << "valore a=" ;  cout <<a <<endl;
   cout << "valore b=" ;  cout <<b <<endl;
   cout << "valore c=" ;  cout <<c <<endl;
   system("PAUSE");
   return 0;
}
```

Una ulteriore direttiva define

define testo1 testo2

Definisce la ricerca e sostituzione di ogni occorrenza di *testo1* sostituendola con *testo2*

Utilizzata nella definizione di costanti simboliche

Prima del pre-processing

```
# define  RADICE2  1.41
main ( )
float lato = 18;
float diagonale = lato * RADICE2;
-----------------------------------
```

Dopo il pre-processing

```
main ( )
float lato = 18;
float diagonale = lato * 1,41;
-----------------------------------
```

Presentazione di un semplice programma in C++

Assegnata una sequenza di numeri positivi ne determina “quanti ne sono” e “la loro somma”

La sequenza è terminata da un valore negativo

3
5
7
8
2
1
-6

Numero valori positivi =6
Somma =26

```
#include <iostream>
using namespace std;
/* Esercizio100  */
int main()
{
//    dichiarazioni variabili
      int numero, conta,somma;
//    assegnazioni iniziali
      conta=somma=0;
//    lettura primo numero
       cout <<" valore ="; cin >>numero; cout <<endl;
//    esame dei valori
      while (numero >0)
        {
            conta=conta+1;
            somma =somma+numero;
            cout <<" valore ="; cin >>numero; cout <<endl;
         }
      cout <<"Numero valori positivi=" <<conta <<endl;
      cout <<"Somma=" <<somma;
      cout <<'\n';
system("PAUSE");
return EXIT_SUCCESS;
}
```

```
valore =3
valore =5
valore =7
valore =8
valore =2
valore =1
valore =-6
Numero valori positivi=6
Somma=26
Premere un tasto per continuare . . .
```