Il C++

Il C++ ha una parte centrale chiamata main, che contiene l' intero programma tra { e }, che nel pascal corrisponde a un “begin” e un “end.” , ossia a un blocco di istruzioni, e come nel pascal anche nel C++ si possono inserire blocchi di istruzioni, un esempio di codice C++ è questo:

```
#include <stdio.h>

main()
{  cout <<  “Ciao!”; }.
```

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

Le librerie da utilizzare

stdio.h = Quella base

iostrea = Quella da utilizzare per i comandi cin e cout

```
main()
```

La parte principale del programma

```
cout
```

consolle output

```
<< invia il valore “Ciao!” a console output per essere stampato a video

;
```

come nel pascal!

Il compilatore da usare sono due: Dev-C++, per l' editor di testi (e compilatore, solo che ha diversi proglemi ed è un miracolo se si compila qualcosa) e quincy2002, come vero compilatore (c e C++).

Le note o commenti

Un commento da estendere su più righe si inizia con “ /* “ e si finisce con “ /* “, invece uno che va dall' inizio del commento per poi finire a fine riga si inizializza con “ // “, le note vengono completamente ignorate dal compilatore e servono solo all' utente per rendere il codice più chiaro o per dare spiegazioni.

Le variabili

Il C++, come il pascal ha diversi tipi di variabili, ma mentre nel pascal si dichiarano con la parola chiave “Var” per poi dichiarare il tipo delle variabili, nel C++ le variabili vengono dichiarate con diverse parole chiavi a seconda del tipo di variabili da dichiarare, ecco la lista:

bool = Variabile di tipo booleano, può restituire solo due valori, true (vero) o false (falso)
int = Variabile contenente numeri non decimali (senza la virgola)
float = Variabile contenente numeri decimali e non, ma sempre espressi con la virgola
char = Variabile contenente caratteri.
Le variabili sono addette a contenere dati, siano essi numeri decimali, interi, lettere, sempre dati, poi ne vedremo altri.

Inizializzare le variabili

Le variabili possono essere dichiarate con inizializzazione, cioè dandogli subito un dato o meno, esempio di inizializzazione di variabile:

```
int a = 12;
```

in questo modo la variabile viene dichiarate per assumere un valore predefinito: in questo caso 12, esempio di dichiarazione di variabile senza inizializzazione:

```
int a;
```

Come nel pascal, si possono dichiarare più variabili allo stesso tempo:

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

si possono concatenare:

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

si può sommare:

```
int a,b,c
a = b + c;
```

sottrarre:

```
int a,b,c;
a = b – c;
```

moltiplicare:

```
int a,b,c;
a = b * c;
```

dividere:

```
int a,b,c;
a = b / c;
```

Ma possiamo anche eseguire operazioni non solo tra variabili ma anche con numeri, esempio:

```
int a;
a = 12 + 3;
```

ecc.....

L' input da tastiera

Come abbiamo visto cout (consolle output) stampa caratteri sullo schermo, ma ora impareremo cin (consolle input), che viene usato per mettere in una variabile i dati inseriti da tastiera, ora farò un esempio di un programma, che dati due numeri li somma:

```
#include <stdio.h>
main()
{ float numero1,numero2,risultato;
  cout << “Inserire il primo numero: “;
  cin >> numero1;
  cout << “Inserire il secondo numero: “;
  cin >> numero2;
```

```
  risultato = numero1 + numero2;
}
```

Vedremo in seguito come stampare a video il valore di una variabile.

Le variabili Booleane

Per utilizzare il C++ è neccessario sapere gli operatori logici:

== uguale

<= minore o uguale

>= maggiore o uguale

!= diverso

< minore

> uguale.

Vedremo ora, come far assumere valore a una variabile booleana:

```
#include <stdio.h>
main()
{ float numero1,numero2;
  bool boo
  cout << “Inserire il primo numero: “;
  cin >> numero1;
  cout << “Inserire il secondo numero: “;
  cin >> numero2;
  boo = numero1 == numero2;
}
```

se la situazione espressa dall' operatore logico è vera anche la variabile booleana assumerà il valore vero (true), se invece la situazione non è quella, la variabile booleana assumerà il valore falso (false).

Infine passiamo ad analizzare gli operatori logici, usati nell’ambito delle espressioni
booleane.

&& and

il risultato è true se entrambi gli operandi sono true

|| or

il risultato è true se almeno uno dei due operandi è true

! not

il risultato è la negazione dell'operando, cioè se un operando ris è true, allora

ris è false e viceversa.

Incremento e decremento

Per incrementare una variabile di uno si fa così:

variabile++

e per decrementarla:

variabile--.

Gli array

Quando è necessario manipolare collezioni di dati omogenei, è opportuno ordinarli in una struttura, detta array. Un array consente di accedere ai dati in esso contenuti specificando un indice che esprime la posizione del dato cercato all'interno della struttura. Per dichiarare un array è necessario specificare il tipo dei dati in esso contenuti, il nome dell'array e le dimensioni. Ad esempio, mediante la dichiarazione:

float v[10];

il compilatore costruisce un array di float, chiamato v, e gli assegna 10 locazioni di memoria consecutive. Il primo elemento dell'array ha indice 0, e si indica con v[0], mentre l'ultimo ha indice 9, e si indica con v[9]. Questa dichiarazione è senza inizializzazione, quindi il contenuto degli elementi dell'array v non è definito.

Per dichiarare un array è necessario specificare il tipo dei dati in esso contenuti, il nome

dell'array e le dimensioni. Ad esempio, mediante la dichiarazione:

float v[10];

il compilatore costruisce un array di float, chiamato v, e gli assegna 10 locazioni di memoria consecutive. Il primo elemento dell'array ha indice 0, e si indica con v[0], mentre l'ultimo ha indice 9, e si indica con v[9]. Questa dichiarazione è senza inizializzazione, quindi il contenuto degli elementi dell'array v non è definito, se vogliamo inizializzare il valore degli array useremo questa sintassi:

int a[3] { 123, 12, 60};

Usare array è vantaggioso, perchè si risparmia la digitazione delle variabili, e righe di dichiarazioni di variabili, però perchè un dato in una allocazione di memoria, per essere letto deve prima essere estratto, quindi, oltra ad rallentare il programma e il computer, è anche difficile (per il PC) da eseguire.

Strutture di controllo

Le strutture di controllo sono dei comandi che permettono di non eseguire, eseguire,eseguire più volte o eseguire solo una certa parte di programma, le strutture di controllo sono: **IF, ELSE, FOR, WHILE E SWITCH**, che esemineremo tra poco.

If – else

If permette di dire se è così esegui quest e else,che va usato dopo if, altrimenti (se nessun if può essere applicato) esegui questo, ecco la sintassi:

if (condizione espressa dagli operatori relazionali) [istruzione] ;

else [istruzione] ;

se invece bisogna eseguire più istruzioni si userà questa sintassi:

```
if ( condizione espressa dagli operatori relazionali )
{
[ istruzione ] ;
[ istruzione ] ;
…....
}

else
{
```

[istruzione] ;

[istruzione] ;

…....

}

dopo le stutture di controllo non vana mai messo il “ ; “ , infatti nel caso di una istruzione singola vanno messi dopo l' istruzione, e non dopo la struttura di controllo!

While

While è la stessa cosa di if, solo che continua a ripetere la/e istruzione/i fino a che la condizione non è più valida.

For

la sintassi del for è questa:

for (espressione1, espressione2, espressione3)

viene eseguita l' espressione 1, ma viene eseguita l' espressione2 solo se la espressione1 può essere eseguita, e la stessa cosa vale per l' espressione3 con l' espressione2.

I commenti

I commenti, detti anche note sono tutte le righe iniziate con // e vengono completamente ignorate dal compilatore e servono solo a chi legge il codice sorgente, servono da appunti o da annotazioni.

Riferimenti

Un riferimento è una specie di collegamento tra variabili, per esempio dando un nome lunghissimo alla variabile, è scomodo

doverla riscrivere, così invece si dichiara una variabile, e si crea un riferimento, per esempio y, e ogni volta che verrà digitato y sarà come digitare la variabile, esempio;

```
int variabile12345678910ciao;
int&  variabile12345678910ciao = y;
```

Così ogni volta che verrà digitato y sarà come digitare la variabile dal nome lunghissimo.

Puntatori

Non trattati in questo corso.

Classi

Le classi servono a dare il diritto di accesso a quelle istruzioni e il diritto di accesso è dato da due keyword:

PRIVATE: Tutto ciò che è private può essere visto solo da quella classe

PUBBLIC: Tutto ciò che è pubblic può essere letta da tutte le classi e da tutte le parti si programma,

la sintassi è questa:

```
class <NomeClasse> {
    public:
      <membri pubblici>
    protected:
      <membri protetti>
    private:
      <membri privati>
 };
```