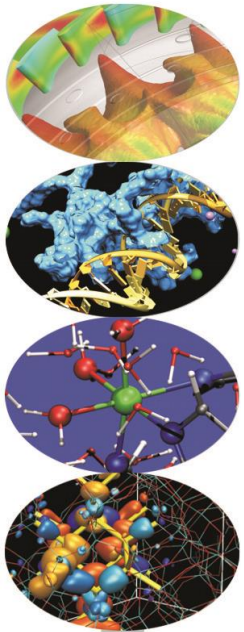
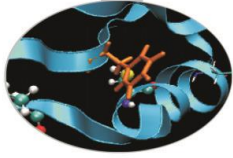


Sintassi II Parte



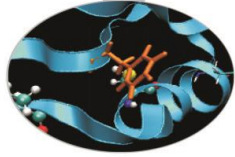
Indice



Durante l'esecuzione di un codice, un programma può ripetere segmenti di codice, o di prendere decisioni e biforcarsi. A tal fine, il C fornisce istruzioni per il controllo di flusso che servono per specificare cosa deve essere fatto dal nostro programma e sotto che quali le circostanze.

- **costrutti decisionali**
- **I cicli**

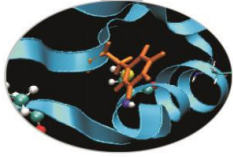
Il costrutto if/else



Il costrutto **if** consente di svolgere una o più operazioni se una particolare condizione (enunciata con un'espressione booleana) risulta verificata.

```
if (boolean expression a){  
statement1a;  
statement2a;  
...  
}  
else if (boolean expression b){  
statement1b;  
statement2b;  
...  
}  
else {  
statement1c;  
...  
}
```

Esempio

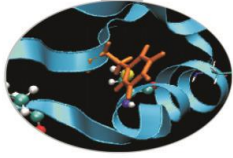


Programma per il calcolo delle radici di un'equazione quadratica:

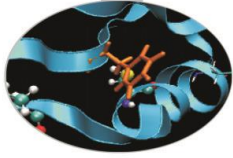
$a x^2 + b x + c = 0$ usando la formula: $x = (-b \pm \sqrt{b^2 - 4 a c}) / (2 a)$

```
double a, b, c, d, e, x1, x2;
/* Test for complex roots */
e = b * b - 4. * a * c;
if (e < 0.)
{
    printf("\nError: roots are complex\n");
    exit(1);
}
/* Test for a = 0. */
if (a == 0.)
{
    printf("\nError: a = 0.\n");
    exit(1);
}
```

Esempio



```
/* Perform calculation */  
d = sqrt(e);  
x1 = (-b + d) / (2. * a);  
x2 = (-b - d) / (2. * a);  
  
/* Display output */  
printf("\nx1 = %12.3e    x2 = %12.3e\n", x1, x2);
```



Errori comuni con il costrutto if/else

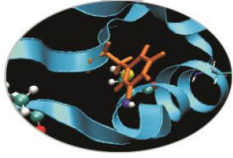
Scriviamo un costrutto if che assegni alle variabili a e b i valori 2 e 4 rispettivamente, quando la variabile x è uguale a 3:

```
if (x == 3){  
    a = 2;  
    b = 4;  
}
```

è la forma corretta. Se dimenticassimo di racchiudere i due assegnamenti in un blocco avremmo:

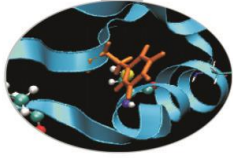
```
if(x == 3)  
    a = 2;  
    b = 4;
```

ciò significa che il costrutto if termina con il ; della prima espressione, dunque b assumerà sempre il valore 4, *indipendentemente* dal valore di x



Errori comuni con il costrutto if/else

- Un blocco di istruzioni, chiamato anche compound statement, è un gruppo di istruzioni che viene trattato dal compilatore come se fosse una singola istruzione.
- I blocchi sono racchiusi da `{}`. Le parentesi contengono le istruzioni da eseguire.
- I blocchi di istruzioni possono essere utilizzati in qualsiasi luogo in cui è consentita una singola istruzione.



Errori comuni con il costrutto if/else

Scrivendo invece:

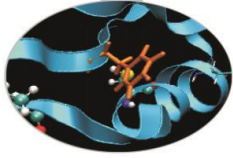
```
if (x = 3){
```

```
    a = 2;
```

```
    b = 4;
```

```
}
```

la condizione del costrutto if risulta sempre vera (si è usato l'operatore di assegnamento = al posto di quello di uguaglianza ==) dunque le istruzioni di assegnamento vengono eseguite *indipendentemente* dal valore di x.



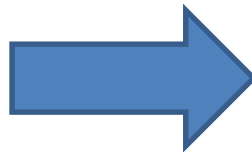
Operatore ternario

L'operatore ternario “?:” è una forma sintetica dell'istruzione if-else, e per questo viene usata per ragioni di comodità e sinteticità del codice.

```
condition?expression1:expression2
```

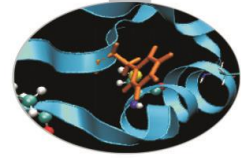
- if/else

```
if (x<y) min=x;  
else min=y;
```



- Operatore ternario

```
min= (x<y)?x:y
```



Il costrutto switch

Come il costrutto *if/else* rappresenta una struttura di selezione. Le etichette **case** individuano i valori costanti della variabile o dell'espressione che controlla il costrutto e sono seguite da una lista di operazioni che il programma svolge finché non incontra la prima istruzione **break**.

```
switch(integer-variabile) {
```

```
  case(a):
```

```
    statement1a;
```

```
    ...
```

```
  break;
```

```
  case(b):
```

```
    statement1b;
```

```
    ...
```

```
      case(c):
```

```
        statement1c;
```

```
    ...
```

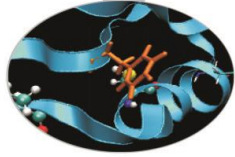
```
  break;
```

```
    default:
```

```
      statement;
```

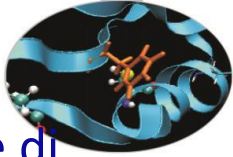
```
}
```

Il costruito switch



```
int decision;
int a, b, c;
scanf("%d", &decision);
switch (decision) { // decision è la variabile di controllo;
case (1) : // se decision == 1 vengono eseguite le due seguenti
            istruzioni;
            a = 20;
            b = 12;
            break; // se decision == 1 qui il costruito si interrompe;
case (2) :
            a = 10;
            b = 12; // queste due istruzioni sono eseguite solo se decision
                    == 2;
case (3) :
            c = 32; // questa istruzione viene eseguita sia che decision == 2 o
                    // decision == 3;
                    break;
default :
            a = 0; // queste istruzioni sono eseguite in tutti i casi non
            b = 1; // contemplati in precedenza;
}
```

Il ciclo while



E' una struttura iterativa che permette di svolgere ripetutamente una serie di operazioni fin tanto che la condizione da cui dipende rimanga vera.

```
while( boolean expression ){
```

```
  statement1;
```

```
  statement2;
```

```
  ...
```

```
}
```

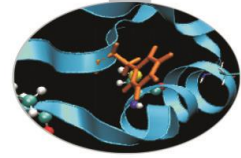
Esempio:

```
while(x > 0)
```

```
  x -= 2;
```

N.B.: il ciclo while potrebbe non essere mai eseguito se la condizione risultasse falsa in partenza.

N.B. il ciclo deve terminare ad un certo punto!

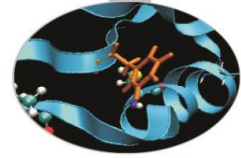


Il ciclo while

```
#include <iostream>
using namespace std;
int main () {
    int n;
    cout << "Enter the starting number > ";
    cin >> n;
    while (n>0)
    { cout << n << ", "; --n; }
    cout << "FIRE!\n";
    return 0; }
```

OUTPUT

Enter the starting number > 8
8, 7, 6, 5, 4, 3, 2, 1, FIRE!



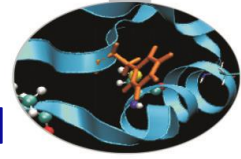
Il ciclo do/while

Si differenzia dal ciclo while per il fatto che la condizione è posta in fondo alla lista di istruzioni: questo assicura che il ciclo venga eseguito almeno una volta.

```
do{  
  statement1;  
  statement2;  
  ...  
} while(boolean expression);
```

Esempio:

```
do{  
    x = x+10;  
} while (x < 100);
```



Il ciclo for

Il ciclo **for** si differenzia dal ciclo while perché contiene obbligatoriamente al suo interno un contatore in modo da svolgere un numero prefissato di iterazioni. E' del tutto analogo al costrutto *do/end do* del Fortran90.

for (*starting expression; boolean expression; counter*) {

statement1;

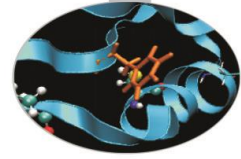
statement2;

...

}

esempio:

```
int main() {  
int x=10;  
for(int i=0; i<10; i++){  
x += 2;  
cout << "X = " << x << " quando i = " << i+1 << endl;  
}
```



Il ciclo for

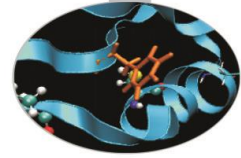
- N.B.: è sempre possibile costruire un ciclo while che contenga un contatore.

```
int main() {  
int x=10;  
for(int i=0; i<10; i++)    cout<<i<<endl;  
}
```

OPPURE

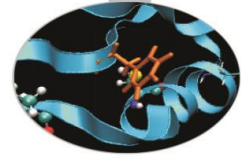
```
int main() {  
int i=0;  
while(i<10) {  
    cout<<i<<endl;  
    i++  
}
```


Esempio



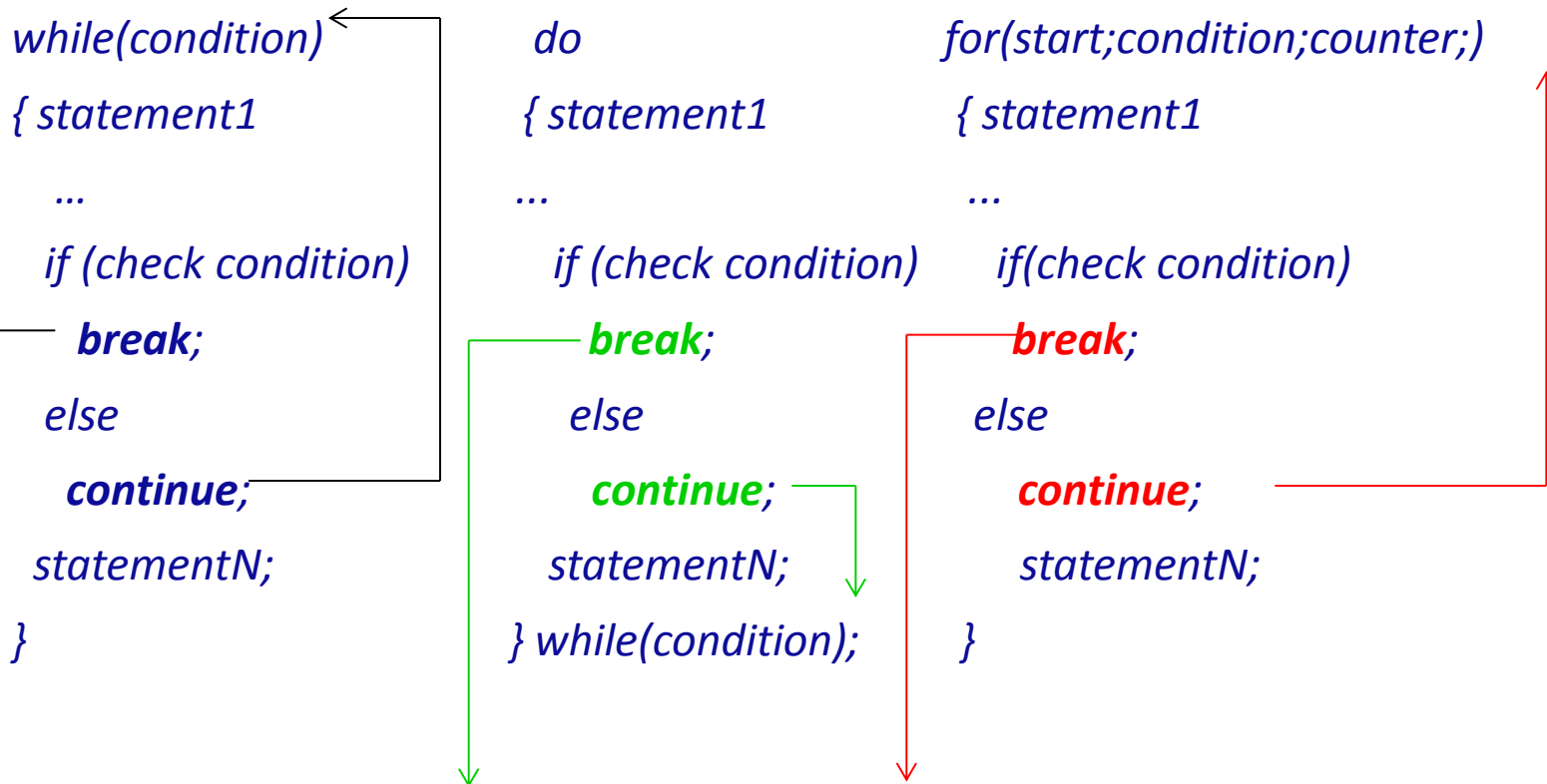
- Calcolo del fattoriale di un numero n

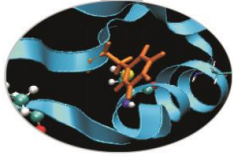
```
int main()
{
    int n,i;
    int fact=1;
    printf(“Inserisci un numero: “)
    scanf(“%d”,&n);
    if(n<0)
    {
        printf(“Error, il numero è negativo\n”);
        exit(1);
    }
    for(i=n;i>0;i--) fact*=double(i);
    printf(“Il fattoriale di %d è %d”,n,fact);
    return 0;
}
```



Le istruzioni *break* e *continue*

Al verificarsi di particolari condizioni è possibile interrompere l'esecuzione di un ciclo attraverso l'istruzione ***break*** oppure passare all'iterazione successiva grazie all'istruzione ***continue***.



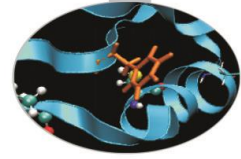


Esempio

calcolo della somma dei primi dieci numeri naturali; uso di break

```
#include<iostream>
using namespace std;

int main(){
    int sum=0, i=0;
    while(true){
        i++;
        sum +=i;
        if(i==10){
            cout << "Sum is: " << sum << endl;
            break;
        }
        cout << "i: " << i << endl;
    }
    return 0;
}
```



Le istruzioni break e continue

In output abbiamo:

```
i: 1
```

```
i: 2
```

```
i: 3
```

```
i: 4
```

```
i: 5
```

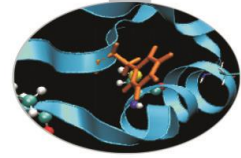
```
i: 6
```

```
i: 7
```

```
i: 8
```

```
i: 9
```

```
Sum is: 55
```



Le istruzioni break e continue

calcolo della somma dei primi cinque numeri pari; uso di continue

```
#include <iostream>
using namespace std;

int main(){
    int sum=0;
    for(int i=1; i<11; i++){
        if(i%2 != 0)
            continue;
        sum +=i;
        cout << "i: " << i << endl;}
    cout << "Sum is: " << sum << endl;
    return 0;
}
```

output:

```
i: 2
i: 4
i: 6
i: 8
i: 10
Sum is: 30
```