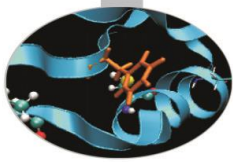
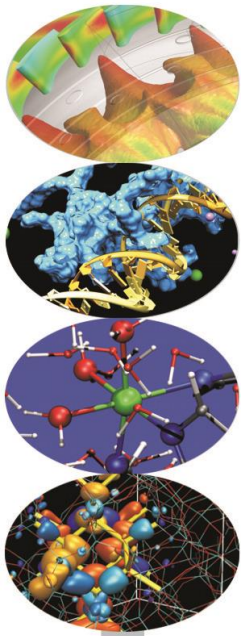
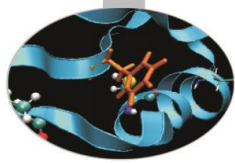


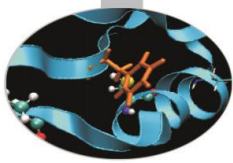
Sintassi II Parte





Indice

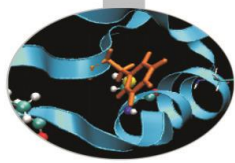
- **I costrutti decisionali**
- **I cicli**



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.

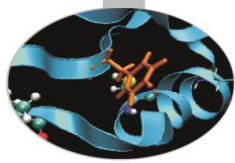
```
if (boolean expression a){
        statement1a;
        statement2a;
        ...
 }
else if (boolean expression b){
        statement1b;
        statement2b;
        ...
}
```



Esempio

```
int sign, x;
cin >> x;
if(x > 0){
        sign =1;
        cout << " x è un intero positivo " << endl;
}
else if(x < 0){
        sign = -1;
        cout << " x è un intero negativo " << endl;
}
else if(x==0){
        sign=0;
        cout << " x è uguale a zero " << endl;}
```

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:

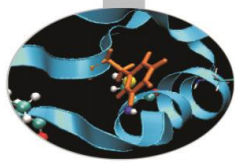
```
if (x == 3){
    a = 2;
    b = 4;
}
```

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

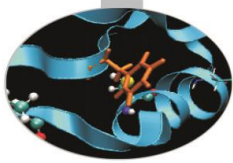
Errori comuni con il costrutto if/else



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.

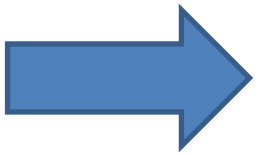


Operatore ternario

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

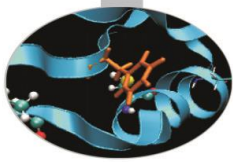
- if/else

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



- Operatore ternario

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

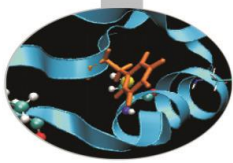


Il costrutto switch

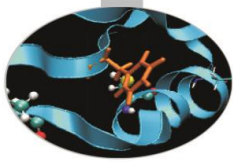
Come il costrutto *if/else* rappresenta una struttura di selezione. Le etichette ***case*** individuano i valori costanti della variabile o dell'espressione che controlla il costrutto e sono seguite da una lista di operazioni che il programma svolge finché non incontra la prima istruzione ***break***.

```
switch(variabile) {
        case(a):
           statement1a;
           ...
        break;
        case(b):
           statement1b;
           ...
        case(c):
           statement1c;
           ...
        break;
        default:
           statement;
}
```

Il costrutto switch



```
int decision;
int a, b, c;
cin >> 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 costrutto 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
          break;       // decision == 3;
      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;

...

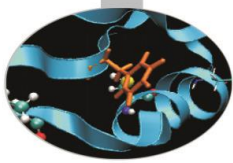
}

Esempio:

```
while(x > 0)
    x -= 2;
```

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

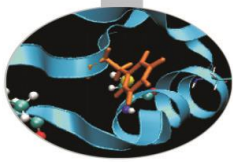
Il ciclo while



```
#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

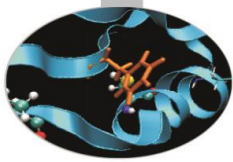
Si differenzia dal ciclo while per il fatto che la condizione è posta in fondo alla lista di istruzioni: questo assicura che il ciclo venga eseguito almeno una volta.

```
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;

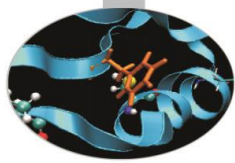
statement2;

…

}

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

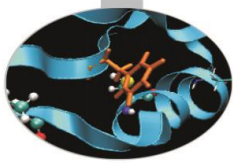
- N.B.: è sempre possibile costruire un ciclo while che contenga un contatore.

```
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++
}
```

Le istruzioni break e continue

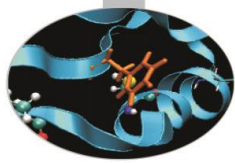


Al verificarsi di particolari condizioni è possibile interrompere l'esecuzione di un ciclo attraverso l'istruzione ***break*** oppure passare all'iterazione successiva grazie all' istruzione ***continue***.

```
while(condition)
{ statement1
  …
  if (check condition)
    break;
  else
    continue;
  statementN;
}
```

```
do
{ statement1
…
  if (check condition)
    break;
  else
    continue;
  statementN;
} while(condition);
```

```
for(start;condition;counter;)
 { statement1
 …
   if(check condition)
     break;
   else
     continue;
     statementN;
 }
```

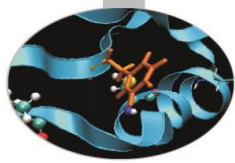


Esempio

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

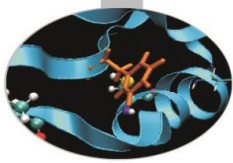
```
#include<iostream>
using namespace std;

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
```