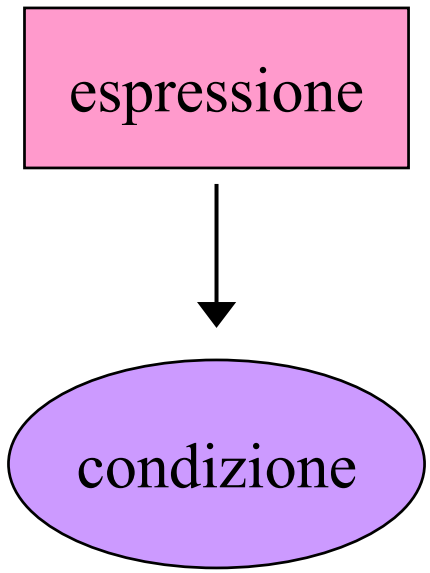
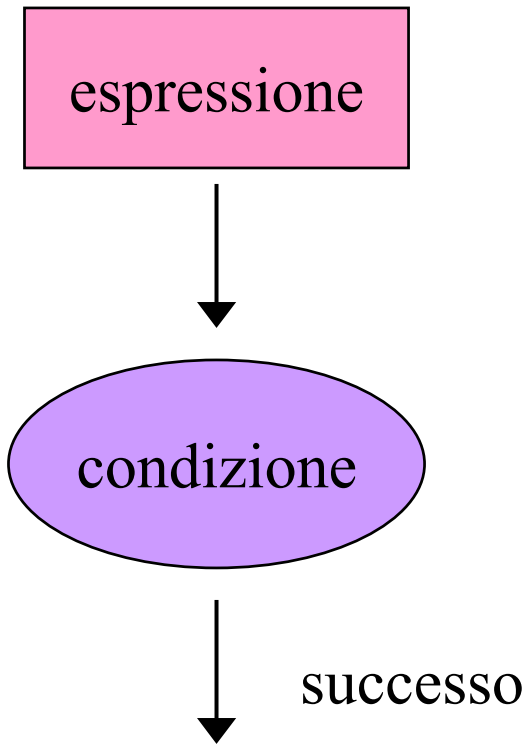
Istruzioni di selezione (If/else/switch)

espressione


espressione
condizione


espressione
condizione
successo

Sintassi di if

```
if (espressione) istruzione
```

dove espressione è una qualsiasi espressione C++ e istruzione può essere una singola istruzione o una sequenza di istruzioni racchiusa tra { e }.

Semantica di if

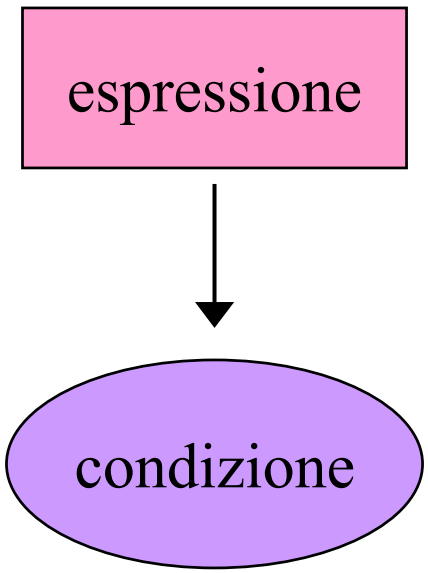
- Nell'esecuzione di un'istruzione if viene valutata l'espressione tra parentesi
  - Se il suo valore è non nullo allora viene eseguita l'istruzione
  - Se il suo valore è nullo l'istruzione viene ignorata.
  - In entrambi i casi l'esecuzione procede con l'istruzione successiva all'if

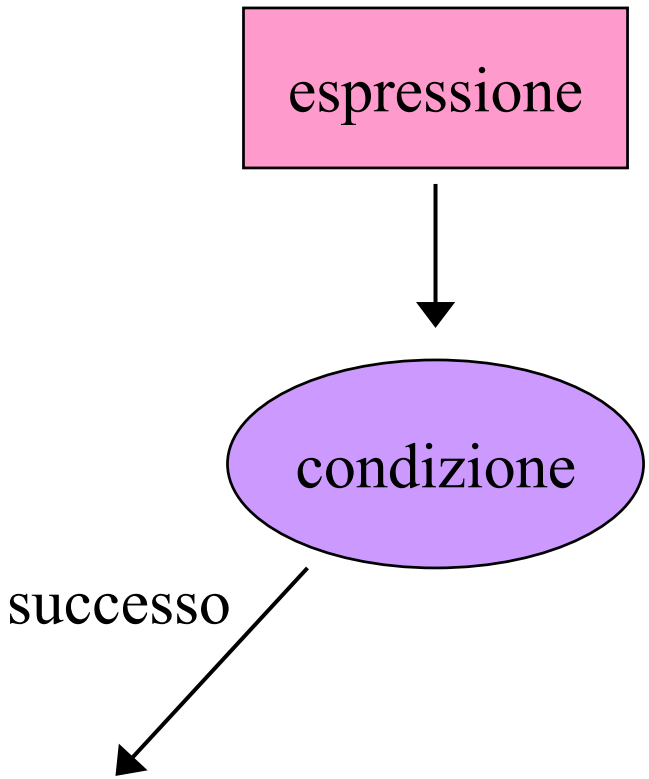
Esempi di uso di if

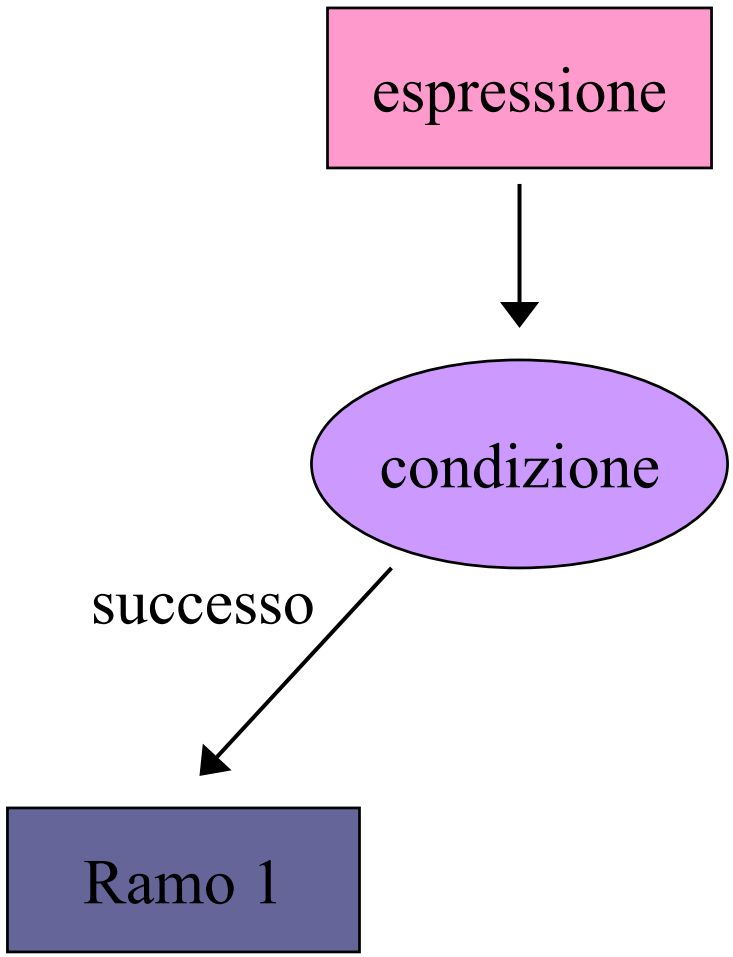
```
:
:
if ( x < y) x = x + 5;
:
:
if (x==0) {
   x = -1;
   y = y + 25;
}
:
```

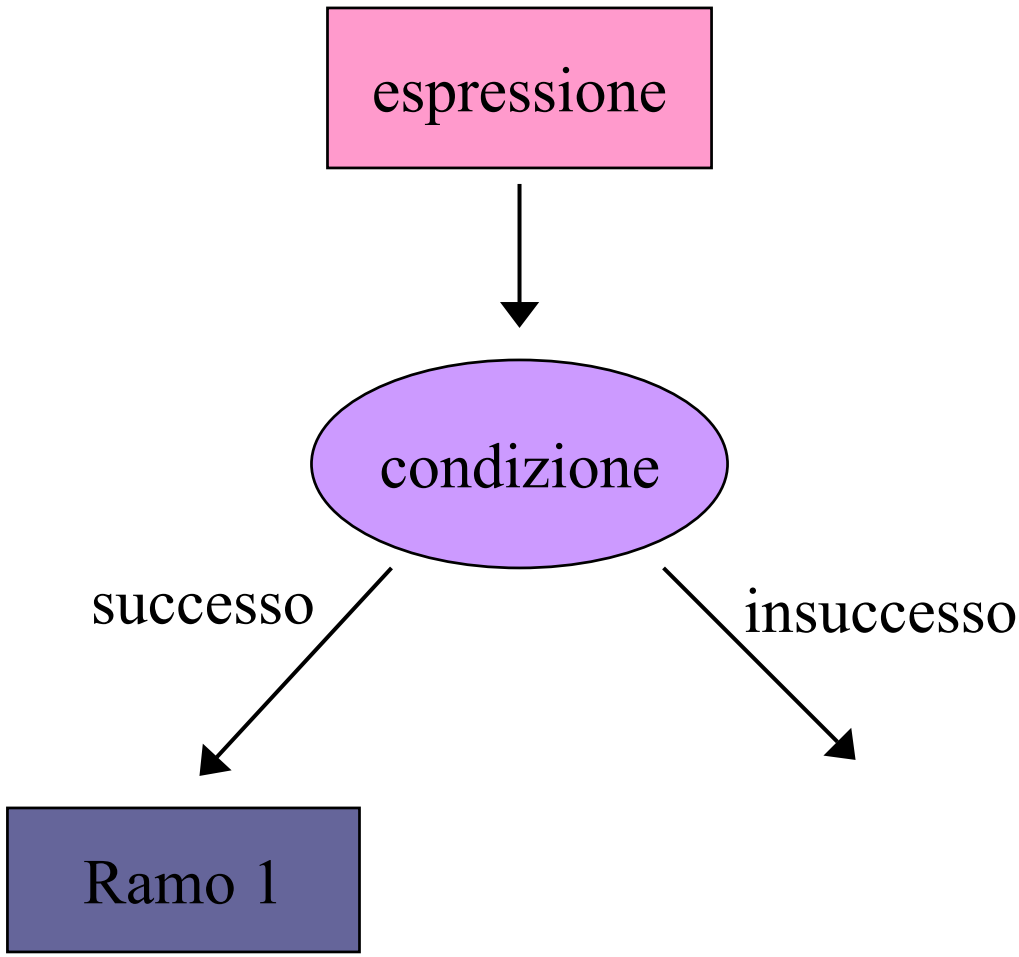
Esempio di programma

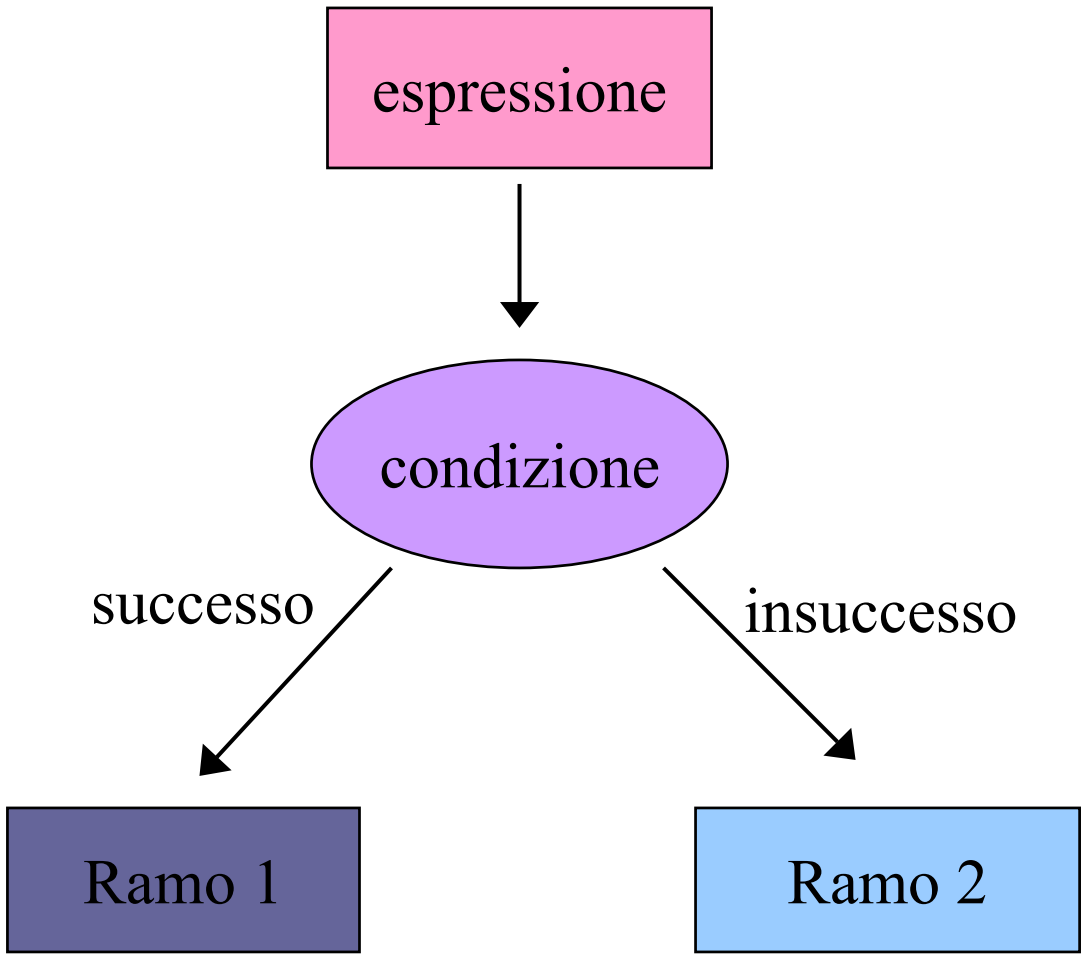
```
#include<iostream.h>
int main (  )
{
   int  x, y;
   cout<< " Inserire due numeri " ;
   cin>>x>>y;
   if(x > y) cout << "Il primo e piu grande.\n";
   if(x == y) cout<< "Sono uguali.\n";
   if(x < y) cout<<"Il secondo e piu grande.\n";
   cout<<" letti"<< x<<" e "<<y<<"\n";
   return 0;
}
```


espressione
condizione


espressione
condizione
successo


espressione
condizione
successo
Ramo 1


espressione
condizione
successo
insuccesso
Ramo 1


espressione
condizione
successo
insuccesso
Ramo 1
Ramo 2

Sintassi di if…else

```
if (espressione) istruzione1 else istruzione2
```

dove espressione è una qualsiasi espressione C++ e istruzione1(2) sono una istruzioni singole o sequenze di istruzioni racchiuse tra { e }.

Semantica di if…else

- Viene valutata l'espressione tra parentesi:
  - Se il valore è non nullo viene eseguita l'istruzione1 e non viene eseguita l'istruzione2
  - Se invece il valore è nullo l'istruzione1 non viene eseguita e viene eseguita l'istruzione2.

Esempi

```
:
if ( x != 0 ) {
        y = y / x;
        x = x - 1;
} else {
        x = 25;
        y = y / x;
}
:
```

```
:
if ( x != 0 ) {
        y = y / x;
        x = x - 1;
      }
else {
        x = 25;
        y = y / x;
      }
:
```

```
:
:
if ( x >y ) x = y - 6 ;
else y = x + 6;
:
```

Corretto ma di difficile lettura: quando si legge la riga dell'if non ci si aspetta che ci sia un seguito…

```
:
:
if ( x > 0 ) {
        x = x + 25;
        y = y / x;
        }
else x = x +1;
:
:
```

```
:
:
if ( x != 0 )  y = y / x;
else {
        x = 25;
        y = y / x;
        }
:
```

Esempio di programma

```
#include<iostream.h>
int main( ){
   int  x, y, small, large;
   cout << "Inserire due numeri" << endl;
   cin >> x >> y;
   if (x > y){
       large = x;
       small = y;
   } else {
       large = y;
       small = x;
   }
   cout<<"in ordine crescente:"
       <<small<<" "<<large<<endl;
   return 1;
}
```

if... else annidati

- Nei costrutti più complessi, in cui si vogliano porre delle condizioni all'interno di un if…else, si pone il problema di associare correttamente l'else ad un if.
- La regola è che ogni else è associato al più vicino if che lo preceda, sempre che questo non sia già stato associato ad un altro else.

Esempio:

Quanto vale y dopo l'esecuzione di questo blocco di istruzioni, dato x = 4.5?

```
y = 3;
if (x > 4)
if (x > 5)
if (x > 6) y = 3;
else y = 4;
else y = 5;
```

Utilizziamo l'incolonnamento per visualizzare I diversi blocchi logici:

```
y = 3;
if (x > 4)
   if (x > 5)
      if (x > 6) y = 3;
      else y = 4;
   else y = 5;
```

sono tutte istruzioni singole e quindi le parentesi graffe non sono necessarie, ma aiutano!

```
y=3;
if (x > 4) {                    // vero per x=4.5
    if (x > 5) {                // falso per x=4.5
        if (x > 6) {
            y = 3;
        } else {
            y = 4;
        }
    } else {                    // eseguito se x <= 5
       y = 5;                   // per x=4.5 y=5 !
    }
}
```

Visibilità di un identificatore

- Un blocco di codice è il codice contenuto all'interno di una coppia di parentesi *graffe*: {…}.

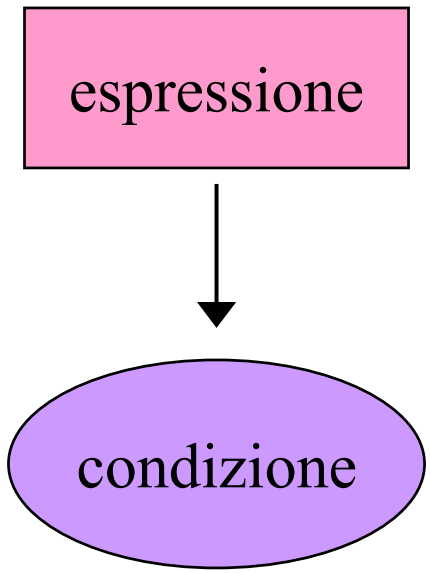
- La visibilità (*scope*) di un identificatore è data dal codice che lo dichiara e dalle istruzioni successive all'interno del più piccolo blocco contenente la dichiarazione: l'identificatore è valido (visibile) solo in questa parte del codice.

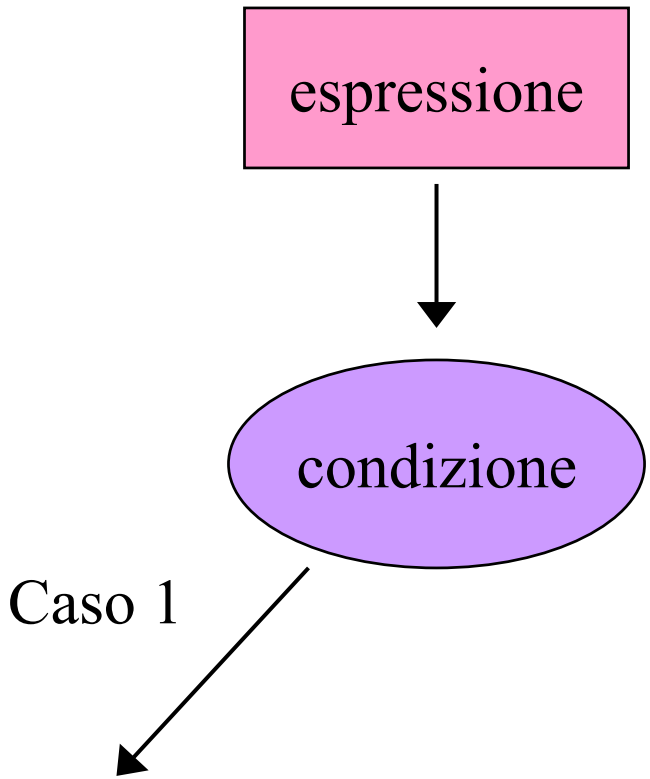
Esempi:

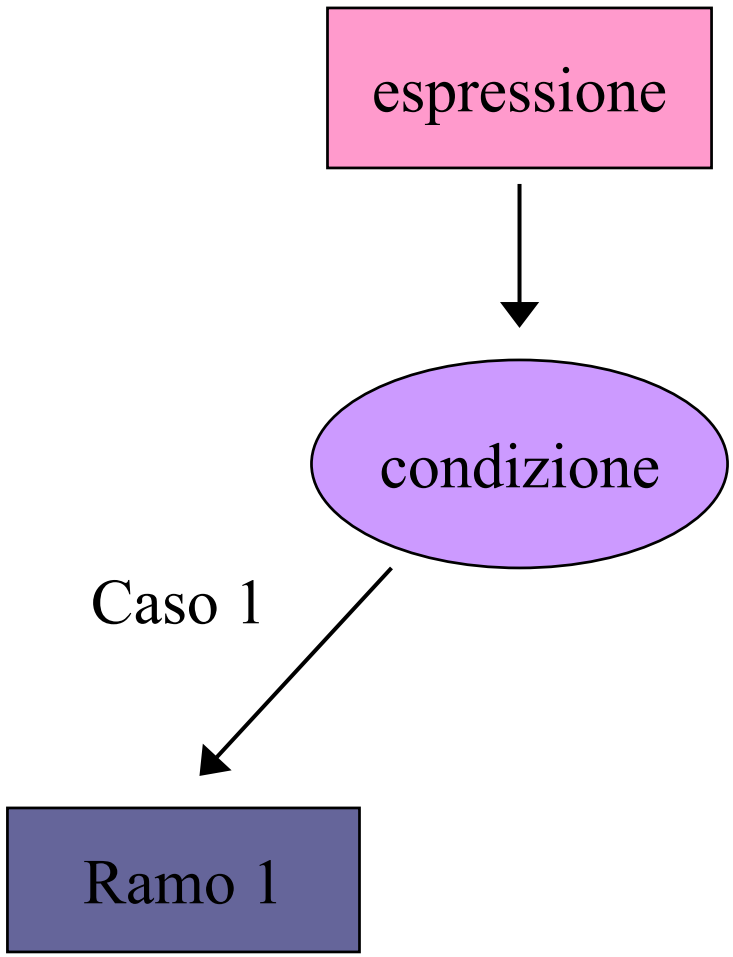
```
{  int x ;
       x = 5;
       cout << x << endl;
}
```

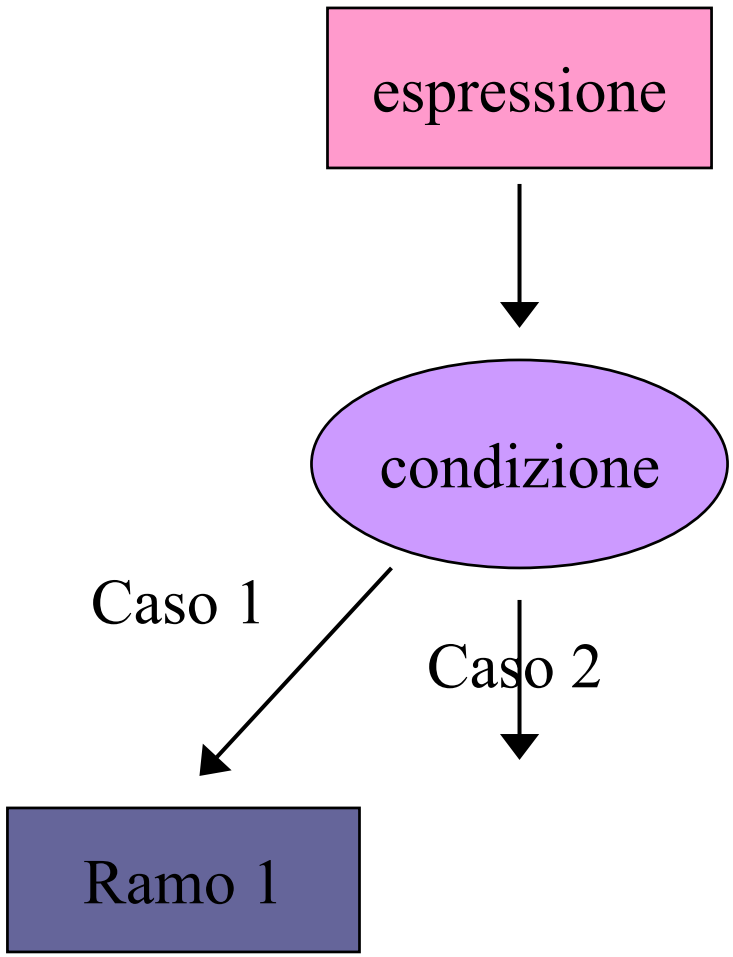
```
{  int  x, y;
       y = x = 25;
       {       double z;
               z = 20;
       }
   cout<< x + y <<endl;
}
```

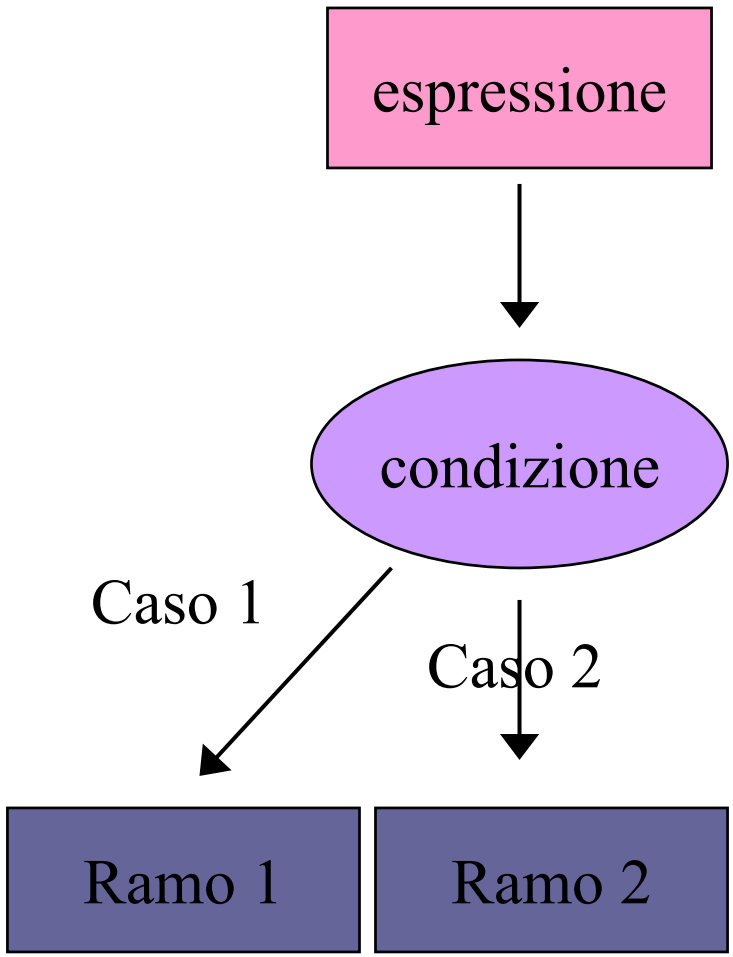
```
{ int  x, y;
   y = x = 25;
        {       double z;
                z = x + y;
        }
   cout<< x + y << z <<endl;  //errore!
   }
```

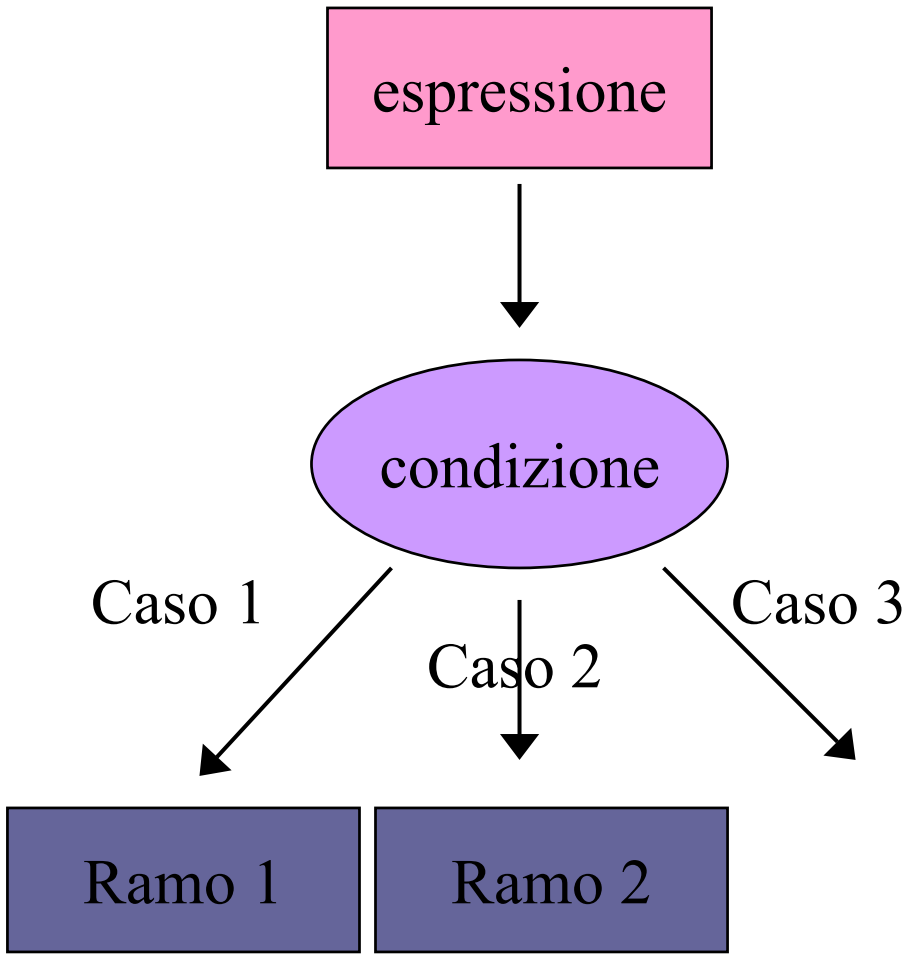

espressione
condizione

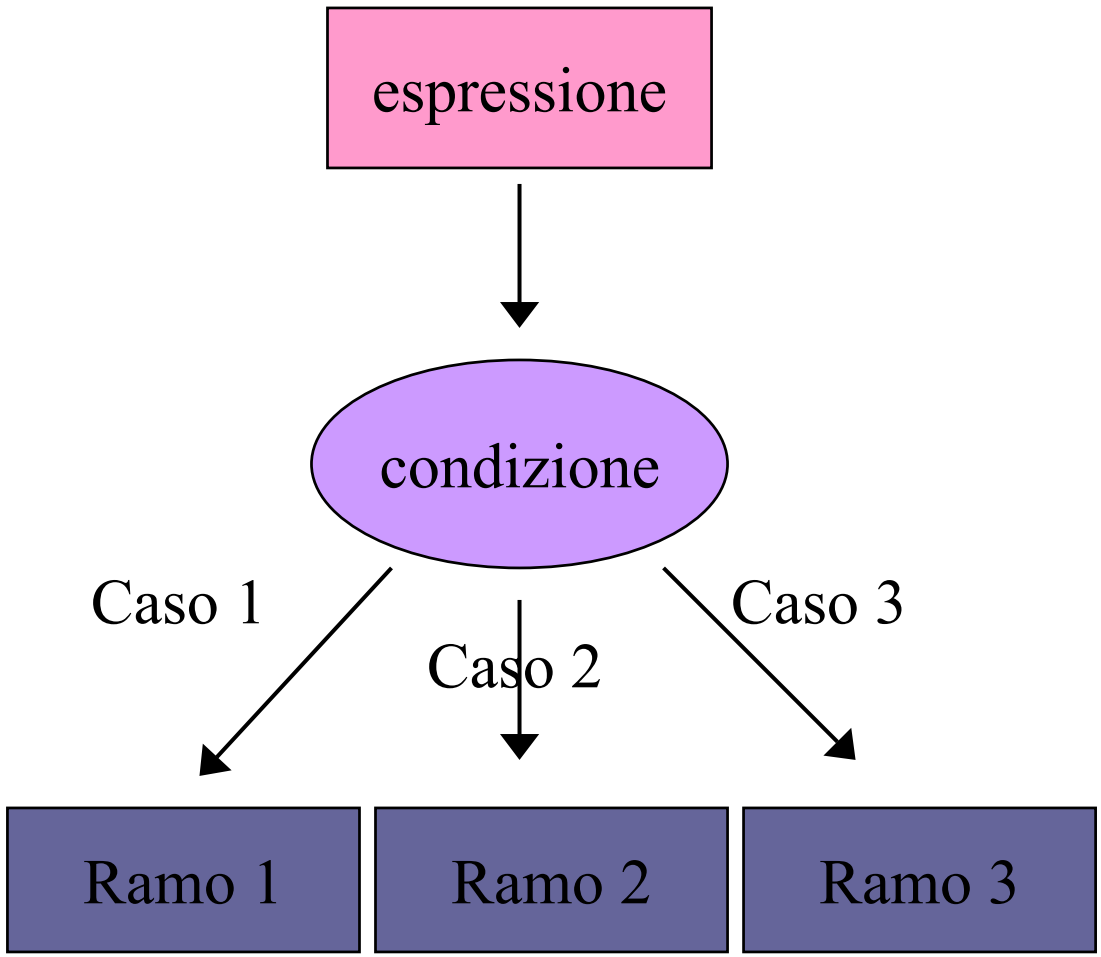

espressione
condizione
Caso 1


espressione
condizione
Caso 1
Ramo 1


espressione
condizione
Caso 1
Caso 2
Ramo 1


espressione
condizione
Caso 1
Caso 2
Ramo 1
Ramo 2


espressione
condizione
Caso 1
Caso 2
Caso 3
Ramo 1
Ramo 2


espressione
condizione
Caso 1
Caso 2
Caso 3
Ramo 1
Ramo 2
Ramo 3

Sintassi di switch

```
switch(espressione) {
                case val1: istruzione1
                case val2: istruzione2
                :
                case valn: istruzionen
        default: istruzione
}
```

Dove l'espressione assume valori interi (val1,… val*n*) e ogni istruzione può essere singola o composta da una sequenza di istruzioni.

Semantica di switch

Nell'esecuzione dell'istruzione switch

1. Viene valutata l'espressione
2. Il valore dell'espressione viene confrontato con I diversi valori dei case. Se è uguale ad uno di questi viene eseguito il blocco istruzione ad esso associato.
3. Se il valore dell'espressione non coincide con nessuno dei valori previsti si esegue il blocco di istruzioni associato a default.

NB: una volta trovato il caso giusto il programma esegue anche tutti i casi successivi!

Esempio

```
int dato;
cin >> dato;
switch(dato) {
      case 5: cout<< "dato == 5";
      case 4: cout<< "dato == 4";
      default:
            cout<< "dato diverso da 4 o 5 \n";
}
```

Se dato==6 viene eseguita solo la stampa di default, se è 4 vengono eseguite quelle del case 4 e del default, se è 5 vengono eseguite tutte!

break

Se si inserisce un break alla fine di ogni blocco istruzioni si ovvia al problema precedente:

```
int dato;
cin >> dato;
switch(dato) {
      case 5: { cout<< "dato == 5"; break ; }
      case 4: { cout<< "dato == 4"; break ; }
      default:
           cout<< "dato diverso da 4 o 5 \n";
}
```

else if

L'uso di switch con break corrisponde in definitiva a un uso di if…else if…else:

```
if(dato ==5) {
    cout<< "dato == 5";
} else if(dato==4) {
    cout<< "dato == 4";
} else {
    cout<< "dato diverso da 4 o 5 \n";
}
```

Preprocessore

- È possibile inserire delle scelte anche a livello del preprocessore, tipicamente per definire quali parti di codice vadano compilate e quali no.
- Ricordando che tutte le direttive del preprocessore iniziano con il simbolo # vediamo alcuni esempi.

define

- Con l'istruzione

  #define NOMEVAR

  si dichiara una variabile al preprocessore
- Con l'istruzione

  #define NOMEVAR valore

  le si assegna un valore
- Con l'istruzione

  #undef NOMEVAR

  la si cancella completamente

define

- Le variabili per il processore possono essere definite anche esternamente, quando si invochi il preprocessore (o il preprocessore + compilatore + ...)

g++ -DNOMEVAR programma.cc

g++ -DNOMEVAR=valore programma.cc

ifdef

- Per mettere nello stesso file sorgente del codice che è diverso a seconda del calcolatore utilizzato si procede generalmente così

```
#ifdef PCWINDOWS
        blocco di codice per PC con Windows
#else
        codice per tutti gli altri
#endif
```

ifndef

- Tutti gli *header file* iniziano con

```
#ifndef NOME_NON_AMBIGUO
#define NOME_NON_AMBIGUO
```

E finiscono con

```
#endif
```

- La prima istruzione verifica che NOME_NON_AMBIGUO non sia stato definito da nessuno, altrimenti vuol dire che il file è già stato incluso e che non serve, anzi potrebbe essere nocivo, includere nuovamente il suo contenuto.