Fondamenti di Informatica e Laboratorio T-AB
Ingegneria Elettronica e Telecomunicazioni

Lab 04

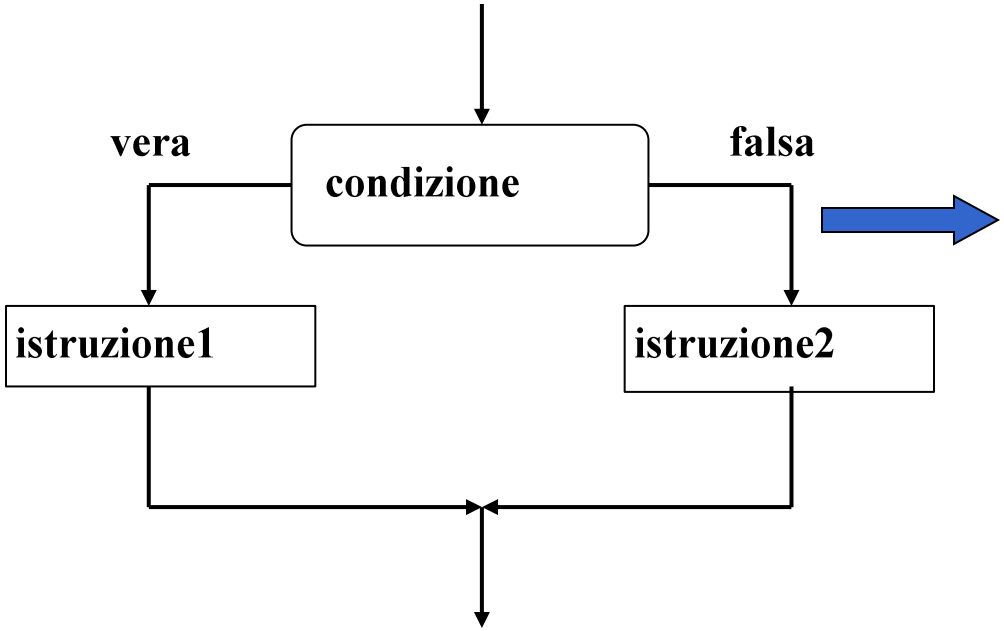
Ripasso Istruzioni
Funzioni semplici

Lab 04

Ripasso Istruzioni

ISTRUZIONE DI SCELTA SEMPLICE

```
<scelta> ::=  if (<cond>) <istruzione1>
                [ else <istruzione2> ]
```



La parte **`else`** è *opzionale*: se omessa, in caso di condizione falsa si passa subito all'istruzione che segue l'**`if`**.

ISTRUZIONI IF ANNIDATE

- Occorre attenzione *ad associare le parti else all' if corretto*

Regola semantica:
l'**`else`** è sempre associato all'**`if`** più interno

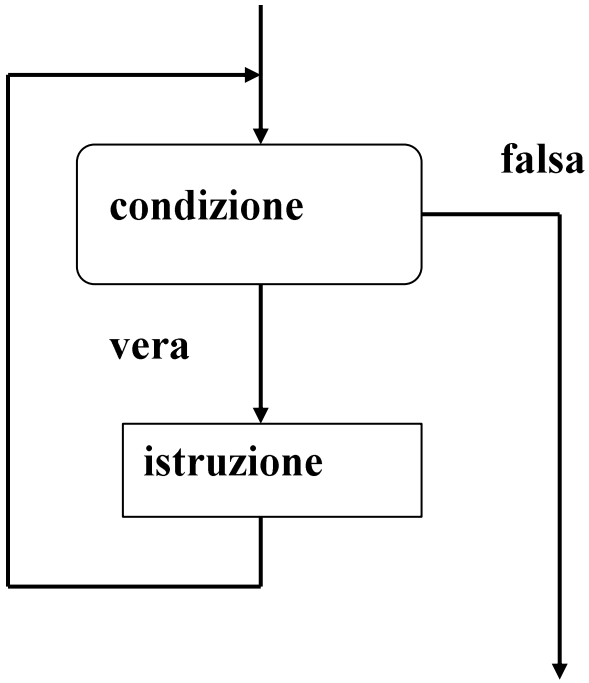
```
if (n > 0)
   if (a>b) n = a;
   else n = b; /* riferito a if(a>b) */
```

Se vogliamo cambiare questa semantica, dobbiamo inserire un blocco

```
if (n > 0)
   { if (a>b) n = a; }
else n = b; /* riferito a if(n>0) */
```

ISTRUZIONE `while`

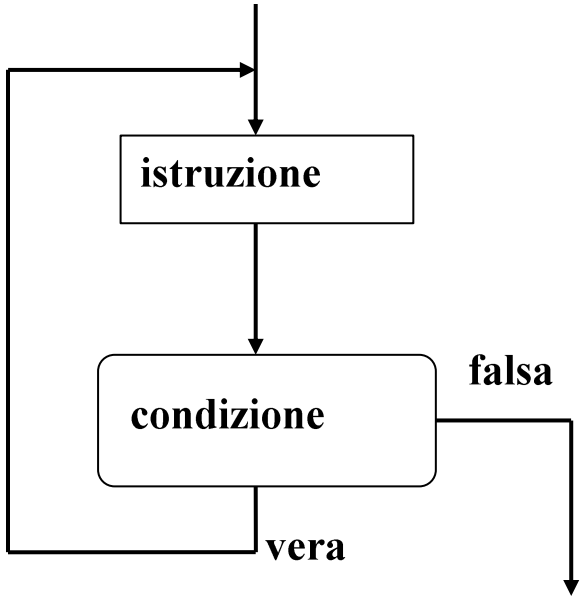
```
<while> ::=
    while(<condizione>) <istruzione>
```



- L'istruzione viene ripetuta *per tutto il tempo in cui la condizione rimane vera.*
- Se la condizione è **falsa**, l'iterazione **non viene eseguita *neppure una volta***.
- In generale, *non è noto* ***quante volte*** l'istruzione sarà ripetuta.

ISTRUZIONE `do..while`

```
<do-while> ::=
  do <istruzione>  while(<condizione>);
```



È una variante della precedente: la condizione viene verificata *dopo* aver eseguito l'istruzione.

Se la condizione è falsa, l'iterazione **viene comunque eseguita *almeno una volta***.

ISTRUZIONE `for`

- È una evoluzione dell'istruzione **`while`** che mira a eliminare alcune frequenti sorgenti di errore:
  - mancanza delle *inizializzazioni delle variabili*
  - mancanza della *fase di modifica del ciclo* (rischio di ciclo senza fine)
- In genere si usa quando e' noto il numero di volte in cui dovra' essere eseguito il ciclo.

```
<for> ::=
 for( <espr-iniz>;<cond>;<espr-modifica>) <istruzione>
```

Esercizio 1 – Cicli

Si realizzi un programma che, partendo da una base **a** ed un limite **n** **(inseriti da tastiera)**, calcoli la seguente funzione:

$$\sum_{i=0}^{n} a^{i}$$

Realizzare il programma in due modi diversi:
1. Utilizzando due cicli (uno per la sommatoria, ed uno per la potenza)
2. Utilizzando un ciclo solo (è possibile usare una condizione)

La soluzione sarà mostrata con cicli for

Esercizio 1 – Cicli - Soluzione

```
#include <stdio.h>

int main()
{       int a, n, i, j;
        int somma;
        int prod;

        somma = 0; //elemento neutro della somma
        printf("Inserisci la base ed il numero di cicli: ");
        scanf("%d%d", &a, &n);

        for (i=0; i <= n; i++) {
                prod  = 1; //elemento neutro del prodotto
                for (j=1; j <= i; j++) {
                        prod = prod * a;
                }
                somma = somma + prod;
        }
        printf("Il risultato della funzione e' %d\n: ", somma);

        return 0;
}
```

Esercizio 1 – Cicli – Soluzione(variante)

```
#include <stdio.h>

int main()
{       int a, n, i;
        int somma;
        int prod;

        somma = 0; //elemento neutro della somma
        prod  = 1; //elemento neutro del prodotto
        printf("Inserisci la base ed il numero di cicli: ");
        scanf("%d%d", &a, &n);

        for (i=0; i <= n; i++) {
                if (i>0) {
                        prod = prod * a;
                }
                somma = somma + prod;
        }

        return 0;
}
```

Esercizio 2 – if innestati

Stampa di caratteri in ordine alfabetico

- Realizzare un programma che legge da input tre caratteri, e li stampa in ordine alfabetico. Si assuma di inserire 3 caratteri diversi.
- Utilizzando l'istruzione if
  - Per determinare il secondo carattere, devo per forza utilizzare degli if innestati

ESEMPIO

Specifica:

se *car1* < car2
    primo = car1
    secondo = car2
altrimenti
    primo = car2
    secondo = car1

se car3 < primo
    …
altrimenti {
    se car3 < secondo
        …
    altrimenti
        …
    }

Esempio con i caratteri 't' 'c' 'g' :

Controllo i primi due caratteri ’t’ e ‘c’

Condizione non verificata

Condizione verificata

primo = 'c' secondo = 't'

Controllo il terzo carattere ‘g’

Condizione non verificata

Condizione verificata

primo = 'c' secondo = 'g'
terzo = 't'

Esercizio 2 – if innestati - soluzione

```
#include <stdio.h>

int main ()
{
   char c1, c2, c3, first, second, third;
   printf("Inserire tre caratteri: \n");
   scanf("%c %c %c", &c1, &c2, &c3);
   if(c1 < c2)
   {
        first = c1;
        second = c2;
   }
   else
   {
        first = c2;
        second = c1;
   }
   …
```

Esercizio 2 – if innestati - soluzione

```
…
if(c3 < first)
{
      third = second;
      second = first;
      first = c3;
}
else
{
      if(c3 < second)
      {
            third = second;
            second = c3;
      }
      else
      {
            third = c3;
      }
}
printf("Characters: %c %c %c\n", first, second, third);
return 0;
}
```

Lab 04

Funzioni Semplici

ESEMPIO

```
int max (int x, int y ){
    if (x>y) return x;
        else return y;
}
```

- Il simbolo **max** denota il nome della funzione
- Le variabili intere **x** e **y** sono i parametri della funzione
- Il valore restituito è un intero **int**

```
int main(){
        int m;
        m = max(2,13);
        return 0;
}
```

Esercizio 1 - Funzioni

Codificare in C la funzione

```
int max(int a, int b)
```

che restituisce il massimo valore tra due interi.

Codificare in C la funzione

```
int max3(int a, int b, int c)
```

che restituisce il massimo valore fra tre interi, sfruttando la funzione max definita precedentemente.

Definire un possibile main che prende in ingresso i tre valori dall'utente (input da tastiera) e ne stampa il massimo.

Esercizio 1 – Le due funzioni

```
int max(int a, int b) {
  if (a>b)
       return a;
  else return b;
}

int max3(int a, int b, int c) {
  int max_di_due;
  max_di_due = max(a,b);
  return max(max_di_due,c);
}
```

Esercizio 1 – Il main

```
void main() {
  int a, b, c, d;

  printf("Inserire 3 tre interi: ");
  scanf("%d%d%d", &a, &b, &c);

  d = max3(a, b, c);
  printf("Il loro massimo e': %d\n", d);
}
```

Esercizio 1 – Il programma

```
#include <stdio.h>

void max(int a, int b) {
   ...
}

void max3(int a, int b, int c) {
   ...
}

void main() {
   ...
}
```

NOTA: Nel file sorgente **`main.c`** e' necessario definire le due funzioni **`max`** e **`max3`** per chiamarle nella funzione **`main`**

Esercizio 1 – Una variante

```
int max(int a, int b) {
   ...
}

int max3(int a, int b, int c) {
  return max(max(a,b),c);
}
```

Esercizio 2 - Funzioni

Si progettino e si realizzino due funzioni così definite:

```
float euro_to_dollari(float money)
float euro_to_lire(float money)
```

ognuna delle quali converte un valore in euro nella moneta corrispondente. A tal fine si supponga che:
1 € = 1.07 $
1 € = 1936.27 £

Si progetti poi un programma che legge da input un valore, inteso come quantità di euro, e stampa la conversione in dollari ed in lire.

Esercizio 2 – Funzioni - Soluzione

```
#include <stdio.h>

float euro_to_dollari(float money)
{return money*1.07;}

float euro_to_lire(float money)
{return money*1936.27;}

void main() {
  float val_euro, dollari, lire;

  printf("Inserire un valore: ");
  scanf("%f", &val_euro);

  dollari = euro_to_dollari(val_euro);
  printf("Il valore in dollari e': %f\n", dollari);

  lire = euro_to_lire(val_euro);
  printf("Il valore in lire e': %f\n", lire);
}
```

Esercizio 3 - Funzioni

Si scriva una funzione

```
int somma_potenze(int a,int n);
```

che dati **a** e **n** deve calcolare $\sum_{i=1}^{n} a^i$

A tal fine si scriva una funzione

```
int potenza(int x,int y);
```

che dati **x** e **y** deve calcolare x^y usando come operazione primitiva il prodotto.

Esercizio 3 – Funzioni - Soluzione

```
int potenza(int x,int y)
{ int i, P=1; /* P: accumulatore di prod.*/
  for(i=1; i<=y; i++)
       P = P * x;
  return P;
}

int somma_potenze(int a, int n)
{ int i, s=0;
  for(i=1; i<=n; i++)
       s = s + potenza(a,i);
  return s;
}
```

Esercizio 3 – Funzioni - Soluzione

Un possibile main

```
int main() {
 int N1,N2,SP;

 printf("Inserisci due interi");
 scanf("%d,%d", &N1,&N2);

 SP = somma_potenze(N1,N2);
 printf("La somma delle potenze vale %d",SP);

 return 0;
}
```

Esercizio 4 - Funzioni

Creare una funzione **`float square(float x)`** . La funzione deve restituire il quadrato del parametro **`x`**.

Creare un'altra funzione, di nome **`float cube(float x)`**, che restituisce invece il cubo del valore **`x`**.

Progettare quindi e codificare un programma che legge un float da tastiera e restituisce il suo quadrato ed il suo cubo. Per calcolare il quadrato ed il cubo si devono utilizzare le due funzioni sopra definite.

Esercizio 4 – Funzioni - Soluzione

```
float square(float x)
{
        return x*x;
}

float cube(float x)
{
        return x*x*x;
}
```

Esercizio 5 - Funzioni

Si scriva una funzione

```
int somma2(int n);
```

che dato **n** deve calcolare $\sum_{i=1}^{n} \sum_{j=1}^{i} j$

A tal fine si sfrutti una funzione^{n}

```
int somma(int n);
```

che dato **n** deve calcolare $\sum_{k=1} k$

Esercizio 5 – Funzioni - Soluzione

```
int somma(int n)
{ int k, s=0;
  for (k=1; k<=n; k++)
       s = s + k;
  return s;
}

int somma2(int n)
{int i, s2 = 0;
 for (i=1; i<=n; i++)
    s2 = s2 + somma(i);
 return s2;
}
```

NOTA: Nel file sorgente prima del **`main`** e' necessario definire la funzione **`somma2`** e prima di **`somma2`** bisogna definire **`somma`**

Esercizio 5 – Funzioni - Soluzione

Un possibile main

```
int main() {
 int N, S;

 printf("Inserisci un intero");
 scanf("%d", &N);

 S = somma2(N);
 printf(”La somma vale %d",S);

 return 0;
}
```

Esercizio 6 - Funzioni

Codificare in C la funzione
`int min_to_sec(int a)` che considera il parametro a come minuti e restituisce il numero di secondi corrispondente.

Codificare in C la funzione
`int ore_to_sec(int a)` che considera il parametro a come ammontare di ore, e restituisca il numero di secondi corrispondente. Si utilizzi la funzione definita precedentemente.

Definire un possibile main che prende in ingresso tre valori interi, rappresentanti ore, minuti e secondi della durata di un CD Audio. Il programma deve stampare il valore corrispondente in secondi.

Esercizio 6 – Funzioni - Soluzione

```
int min_to_sec(int a)
{
	return a*60;
}


int ore_to_sec(int a)
{
	return a*60*60;
}
```

Esercizio 7 - Funzioni

Codificare in C la funzione
`int ipotenusa(int a, int b)` che, dati i cateti **`a`** e **`b`** di un triangolo rettangolo, restituisce il valore dell'ipotenusa.

A tal scopo si utilizzi il Teorema di Pitagora:

$$Ipotenusa = \sqrt{a^2 + b^2}$$

Per calcolare la radice quadrata si utilizzi la funzione di libreria **`sqrt(x)`**. Per utilizzare quest'ultima si aggiunga l'istruzione
`#include <math.h>` in testa al file.

Definire un possibile main che legga da tastiera due valori che rappresentino i cateti di un triangolo rettangolo, e stampi il valore dell'ipotenusa.

Esercizio 7 – Funzioni - Soluzione

```
float ipotenusa(float a, float b)
{
        return sqrt(a*a + b*b);
}
```

Esercizio 8 - Funzioni

Codificare in C la funzione
`int perimetro(int a, int b, int c)` che, dati i lati a,b,c di un triangolo, ne calcola il perimetro.

Codificare in C la funzione
`float area(int a, int b, int c)`
che restituisce l'area di un triangolo i cui lati misurano a, b, c. A tal scopo si usi la formula di Erone:

$$Area = \sqrt{p(p-a)(p-b)(p-c)}$$

Dove p è la metà del perimetro. A tal scopo si includa l'header **`<math.h>`** e si utilizzi la funzione **`sqrt(x)`**.

Definire un possibile main che prende in ingresso i tre lati di un triangolo e stampa perimetro ed area.

Esercizio 8 – Funzioni - Soluzione

```
float perimetro(float a, float b, float c)
{
      return a + b + c;
}


float area(float a, float b, float c)
{
      float p;
      float area;

      p = perimetro(a, b, c) / 2;
      area = sqrt(   p * (p-a) * (p-b) * (p-c)    );
      return area;
}
```

Esercizio 9 - Funzioni

Codificare in C la funzione **`int primo(int x)`** che restituisce:
1 se x è un numero primo
0 altrimenti.
Si utilizzi a tal proposito l'operatore modulo (%).

Si progetti un programma che legge da tastiera un numero N, e stampa a video tutti i numeri primi compresi tra 0 e N.

Esercizio 9 – Funzioni - Soluzione

```
int primo(int x) {
        int i, resto;

        if ((x == 1) || (x == 2))
                return 1;
        else {
                i= 2;
                do
                {
                        resto = x % i;
                        i++;
                }
                while ((resto != 0) && (i<x));

                return (resto != 0);
        }
}
```