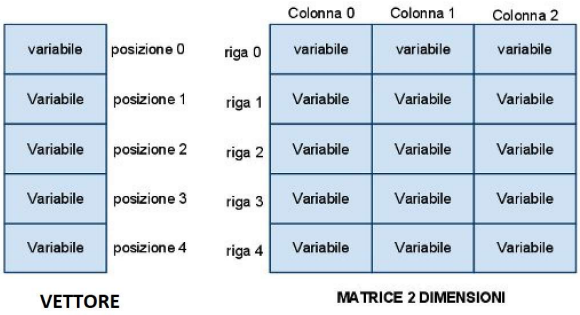
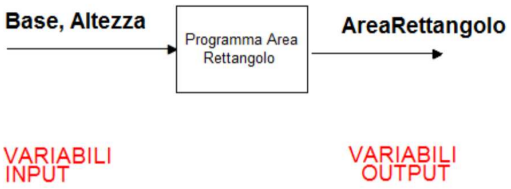
ITIS-LS "Francesco Giordani" Caserta

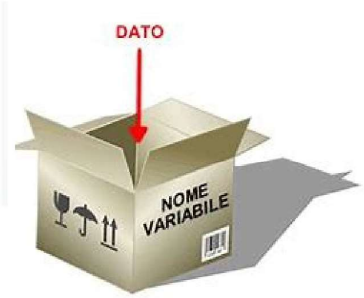
prof. Ennio Ranucci

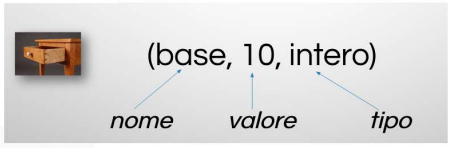
a.s. 2020-2021

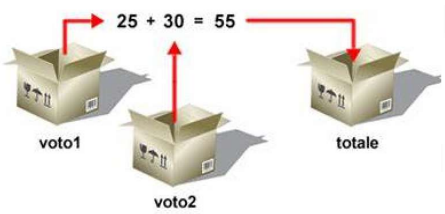
variabili costanti array record











```
/*
ITIS-LS F.Giordani Caserta
Anno scolastico 2020/2021
Classe 3^ sez.B spec. Informatica
Data: 22/01/2021
Numero es: 1
Versione:1.0
Programmatore/i: Lezione collettiva
Sistema Operativo:Windows 10
Compilatore/Interprete: Code::Blocks Release 17.12 rev 11256
Obiettivo didattico: output di una espressione numerica;
Obiettivo del programma: Scrivere a video il risultato di 12*6

TABELLA DELLE VARIABILI
IDENTIF.      I/O/LAVORODESC. TIPO   DIMENSIONE

*/

#include <iostream>
using namespace std;

int main()
{
  cout << 12*6 << endl;
  return 0;
}

/*
ITIS-LS F.Giordani Caserta
Anno scolastico 2020/2021
Classe 3^ sez.B spec. Informatica
Data: 22/01/2021
Numero es: 2
Versione:1.0
Programmatore/i: Lezione collettiva
Sistema Operativo:Windows 10
Compilatore/Interprete: Code::Blocks Release 17.12 rev 11256
Obiettivo didattico: uso di una variabile;
Obiettivo del programma: Scrivere a video il perimetro di un esagono. La
misura del lato è contenuta in una variabile;

TABELLA DELLE VARIABILI
IDENTIF.      I/O/LAVORO          DESC.                TIPO          DIMENSIONE
lato                Input               lato dell'esagono      intero              2  BYTE
*/


#include <iostream>
using namespace std;
int lato; // dichiarazione della variabile
int main()
{
   lato=12;//assegnazione del valore della variabile da programma
   cout << lato*6 << endl;
   return 0;
}
```

```
/*
ITIS-LS F.Giordani Caserta
Anno scolastico 2020/2021
Classe 3^ sez.B spec. Informatica
Data: 22/01/2021
Numero es: 3
Versione:1.0
Programmatore/i: Lezione collettiva
Sistema Operativo:Windows 10
Compilatore/Interprete: Code::Blocks Release 17.12 rev 11256
Obiettivo didattico: uso di una variabile;
Obiettivo del programma: Scrivere a video il perimetro di un esagono. La misura del
lato viene acquisita da tastiera e memorizzata in una variabile;

TABELLA DELLE VARIABILI
IDENTIF.      I/O/LAVORO          DESC.                TIPO          DIMENSIONE
lato                Input              lato dell'esagono      intero              2  BYTE
*/


#include <iostream>

using namespace std;

int lato;

int main()

{

  cout<<" inserire il valore del lato: ";

  cin>>lato;  //assegnazione del valore della variabile da tastiera

  cout << lato*6 << endl;

  return 0;

}
```

```
/*
ITIS-LS F.Giordani Caserta
Anno scolastico 2020/2021
Classe 3^ sez.B spec. Informatica
Data: 22/01/2021
Numero es: 4
Versione:1.0
Programmatore/i: Lezione collettiva
Sistema Operativo:Windows 10
Compilatore/Interprete: Code::Blocks Release 17.12 rev 11256
Obiettivo didattico: uso di una variabile e di una costante
Obiettivo del programma: Calcolare il perimetro di un poligono. La misura del lato
viene acquisita da tastiera e memorizzata in una variabile;

TABELLA DELLE VARIABILI
IDENTIF.      I/O/LAVORO          DESC.                  TIPO          DIMENSIONE
lato                Input              lato dell'esagono      intero            2  BYTE
*/


#include <iostream>
using namespace std;
int lato;
int const numlati=8;
int main()
{
  cout<<" inserire il valore del lato: ";
  cin>>lato;//assegnazione del valore della variabile da tastiera
  cout <<"il perimetro del poligono che ha "<<numlati<<" lati e' "<< lato*numlati<<
endl;
  return 0;
}
```

```
/*
ITIS-LS F.Giordani Caserta
Anno scolastico 2020/2021
Classe 3^ sez.B spec. Informatica
Data: 22/01/2021
Numero es: 5
Versione:1.0
Programmatore/i: Lezione collettiva
Sistema Operativo:Windows 10
Compilatore/Interprete: Code::Blocks Release 17.12 rev 11256
Obiettivo didattico: uso di una funzione
Obiettivo del programma: Scrivere a video il perimetro di un esagono utilizzando la
funzione perimetroEsagono.

TABELLA DELLE VARIABILI
IDENTIF.      I/O/LAVORO        DESC.                TIPO          DIMENSIONE
lato              Input          lato dell'esagono    intero          2  BYTE
*/


#include <iostream>
using namespace std;
int lato,perimetro;
int perimetroEsagono();
int main()
{
   perimetro=perimetroEsagono();
   cout <<"il perimetro dell' esagono e' "<< perimetro<< endl;
   return 0;
}
int perimetroEsagono ()
{
  cout<<" inserire il valore del lato: ";
  cin>>lato;
  return lato*6;
}
```

```
/*
ITIS-LS F.Giordani Caserta
Anno scolastico 2020/2021
Classe 3^ sez.B spec. Informatica
Data: 22/01/2021
Numero es: 6
Versione:1.0
Programmatore/i: Lezione collettiva
Sistema Operativo:Windows 10
Compilatore/Interprete: Code::Blocks Release 17.12 rev 11256
Obiettivo didattico: uso di una funzione
Obiettivo del programma: Scrivere a video il perimetro di un esagono utilizzando la
funzione perimetroEsagono (eliminando l'interfaccia con l'utente interna alla
funzione che è un errore frequente dei principianti).

TABELLA DELLE VARIABILI
IDENTIF.      I/O/LAVORO        DESC.                  TIPO            DIMENSIONE
lato              Input          lato dell'esagono      intero          2  BYTE
*/


#include <iostream>
using namespace std;
int lato,perimetro;
int perimetroEsagono();
int main()
{
   cout<<" inserire il valore del lato: ";
    cin>>lato;
    perimetro=perimetroEsagono();
   cout <<"il perimetro dell' esagono e' "<< perimetro<< endl;
   return 0;
}
int perimetroEsagono ()
{
  return lato*6;
}
```

```
/*
ITIS-LS F.Giordani Caserta
Anno scolastico 2020/2021
Classe 3^ sez.B spec. Informatica
Data: 22/01/2021
Numero es: 7
Versione:1.0
Programmatore/i: Lezione collettiva
Sistema Operativo:Windows 10
Compilatore/Interprete: Code::Blocks Release 17.12 rev 11256
Obiettivo didattico: uso di una funzione
Obiettivo del programma: Scrivere a video il perimetro di un esagono utilizzando la
funzione perimetroEsagono parametrizzata.

TABELLA DELLE VARIABILI
IDENTIF.      I/O/LAVORO        DESC.                TIPO          DIMENSIONE
lato              Input          lato dell'esagono     intero          2  BYTE
*/


#include <iostream>
using namespace std;
int lato;
int perimetroEsagono(int latoPar);
int main()
{
   cout<<" inserire il valore del lato: ";
   cin>>lato;
   cout <<"il perimetro dell' esagono e' "<< perimetroEsagono(lato)<< endl;
   return 0;
}
int perimetroEsagono (int latoPar)
{
  return latoPar*6;
}
```

```
/*
ITIS-LS F.Giordani Caserta
Anno scolastico 2020/2021
Classe 3^ sez.B spec. Informatica
Data: 22/01/2021
Numero es: 8
Versione:1.0
Programmatore/i: Lezione collettiva
Sistema Operativo:Windows 10
Compilatore/Interprete: Code::Blocks Release 17.12 rev 11256
Obiettivo didattico: uso di una funzione
Obiettivo del programma: Scrivere a video il perimetro di un poligono utilizzando la
funzione perimetroPoligono parametrizzata.

TABELLA DELLE VARIABILI
IDENTIF.      I/O/LAVORO      DESC.                  TIPO          DIMENSIONE
lato              Input          lato del poligono     intero        2  BYTE
numLati           Input          numero lati poligono  intero        2  BYTE
*/


#include <iostream>
using namespace std;
int lato,numLati;
int perimetroPoligono(int latoPar, int numLatiPar);
int main()
{
   cout<<" inserire il numero dei lati del poligono: ";
   cin>>numLati;
   cout<<" inserire il valore del lato: ";
   cin>>lato;
   cout <<"il perimetro del poligono e' "<< perimetroPoligono(lato,numLati)<< endl;
   return 0;
}
int perimetroPoligono (int latoPar, int numLatiPar)
{
  return latoPar*numLatiPar;
}
```

PARAMETRI

I **parametri** (o **argomenti**) sono uno o più dati che una funzione/procedura/modulo/sottoprgramma si aspetta di ricevere per eseguire il suo compito..
Le variabili che compaiono nella definizione della funzione (oppure procedura/modulo/sottoprogramma) vengono dette parametri **formali**; i dati che vengono sostituiti ai parametri formali al momento del richiamo della funzione (oppure procedura/modulo/sottoprgramma) vengono detti parametri **attuali** (o argomenti **attuali**).

Nella definizione della procedura vengono descritti i **parametri formali**, cioè le variabili utilizzate all'interno della procedura.
I parametri formali dal punto di vista dell'ambito di visibilità si comportano come variabili locali: sono visibili all'interno della funzione (oppure procedura/modulo/sottoprgramma) in cui sono definiti e di tutti i suoi sottomoduli e nascondono eventuali variabili globali con lo stesso nome.
Al momento del richiamo della procedura si devono fornire i **parametri attuali**, cioè i valori che andranno a sostituirsi ai parametri formali.
L'assegnazione dei valori avviene in base alla **posizione**, associando il primo parametro attuale al primo parametro formale, il secondo parametro attuale al secondo parametro formale e così via.
I parametri attuali nell'istruzione di chiamata di un modulo e i parametri formali nella definizione del modulo stesso devono essere uguali per numero e tipo.

I parametri formali sono identificati da:

- ***un identificatore;***
- ***un tipo;***
- ***una posizione;***
- ***una direzione (input/output) che dipende dalla modalità di passaggio del parametro.***

Nel momento in cui vengono usati, cioè il richiamo della procedura e la sostituzione dei parametri attuali, sono inoltre caratterizzati da un valore.

PASSAGGIO DEI PARAMETRI

Il **passaggio dei parametri** può avvenire per valore e quella per riferimento o indirizzo.

Nella **trasmissione per valore** al parametro formale viene trasmesso il valore del parametro attuale, ma i due dati sono distinti, in locazioni diverse di memoria.
Il parametro formale è una copia temporanea del dato; quindi, anche se il parametro viene modificato all'interno della funzione (oppure procedura/modulo/sottoprogramma), non si ha alcun effetto sul parametro attuale, una volta che la funzione (oppure procedura/modulo/sottoprogramma) è terminato (le modifiche non hanno effetto sulle variabili del modulo chiamante).

Nella **trasmissione per riferimento o indirizzo** il parametro formale e il parametro attuale fanno riferimento alla stessa locazione di memoria.
Alla funzione (oppure procedura/modulo/sottoprogramma) chiamata viene trasmesso l'**indirizzo** del parametro (puntatore); la funzione (oppure procedura/modulo/sottoprogramma) accede direttamente alla variabile e quindi qualsiasi modifica della variabile all'interno della funzione (oppure procedura/modulo/sottoprogramma) ha effetto anche dopo che la funzione (oppure procedura/modulo/sottoprogramma) è terminata; si deve usare il passaggio per riferimento per restituire valori alla funzione (oppure procedura/modulo/sottoprogramma) chiamante.

Passaggio di parametri per valore

```
/*    Scambio del valore di due variabili mediante  valore */
#include <iostream>
using namespace std;
/* Scambio riceve due parametri di tipo float */
void Scambio(float , float );   // prototipo
int main()
{
  float var1 = 3.14, var2 = 1.41;   // variabili locali
  cout << "Prima dello scambio: var1 = " << var1 << " var2 = " << var2 << endl;
  // A Scambio sono passati i valori contenuti in var1 e var2
  Scambio (var1, var2);        // chiamata di funzione
  cout << "Dopo lo scambio....:      var1 = " << var1  << " var2 = " << var2 << endl;
}

// Funzione Scambio che effettua lo scambio
void Scambio(float  x, float  y)
{
   float  temp = x;
   x = y;
   y = temp;
}
```

RAM				
MAIN		FUNZIONE		
var1	var2	x	y	temp
3.14	1.41	3.14	1.41	3.14
		1.41	3.14	

Schermo

```
Prima dello scambio....: var1 = 3.14   var2 = 1.41
Dopo lo scambio....:     var1 = 3.14   var2 = 1.41
```

Es9

```
#include <iostream>
using namespace std;
void Scambio (float, float);
int main()
{
        float var1, var2;
        var1=3.14;
        var2=1.41;
        cout<<"prima dello scambio: var1= "<<var1<<" var2= "<<var2<<endl;
        Scambio(var1,var2);
        cout<<"dopo lo scambio: var1= "<<var1<<" var2= "<<var2<<endl;
        return 0;
}
void Scambio(float x, float y)
{
  float temp;
  temp = x;
  x = y;
  y = temp;
}
```

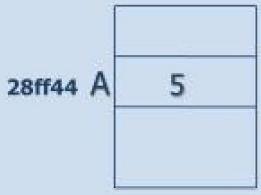
Passaggio di parametri per indirizzo

Il **passaggio di parametri per indirizzo** (o per riferimento) passa alla funzione **l'indirizzo di memoria** del parametro **anziché il valore**.
Non ci sono quindi variabili distinte, una nel main ed una nella funzione, ma la funzione opera sulla stessa area di memoria del main.

Data la variabile intera **A** per sapere qual è l'*indirizzo di memoria di* **A** si usa l'operatore **&**.

Osserviamo le istruzioni seguenti:

```
int A;
A=5;
cout <<"A vale "<<A <<endl;
cout <<"L'indirizzo di A vale "<<&A<<endl;
```



Sullo schermo viene prodotto:

```
A vale 5
L'indirizzo di A vale   0x28ff44
```

Che significa che la variabile A è allocata all'indirizzo di memoria 28ff44 che è espresso in base 16 (0x)

Passaggio di parametri per indirizzo

Vediamo il seguente esempio in cui la funzione "aggiungi" aggiunge 1 ai due parametri ricevuti. Il primo parametro viene passato per valore il secondo per indirizzo.

```
void aggiungi(int, int &);
int main()
{  int a,b;
   a=0;
   b=5;
   aggiungi(a, b);
   cout<<"nel main a vale"<<a;
   cout<<"nel main b vale"<<b;
}
void aggiungi(int x, int &y)
{x++;
 y++;
 cout<<"nella funzione x vale"<<x;
 cout<<"nella funzione y vale"<<y;
 }
```

Schermo

```
nella funzione: x vale 1
nella funzione: y vale 6

nel main: a vale 0
nel main: b vale 6
```

Es9

```
#include <iostream>
using namespace std;
void Scambio (float &, float &);
int main()
{
        float var1, var2;
        var1=3.14;
        var2=1.41;
        cout<<"prima dello scambio: var1= "<<var1<<" var2= "<<var2<<endl;
        Scambio(var1,var2);
        cout<<"dopo lo scambio: var1= "<<var1<<" var2= "<<var2<<endl;
        return 0;
}

void Scambio(float &x, float &y)
{
  float temp;
  temp = x;
  x = y;
  y = temp;
}
```