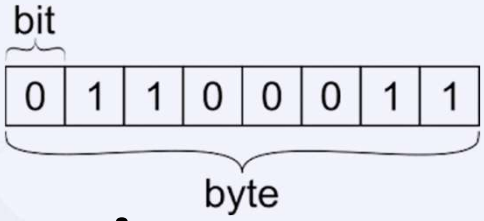
La macchina come comunica?

In BIT, cioè con sequenze di 0 e 1.



Ma allora com'è possibile l'interazione tra Uomo-Macchina?

La comunicazione tra uomo e macchina deve avere regole ben precise per far sì che le intenzioni dell'uno siano comprese dall'altro. Ci vengono in soccorso i linguaggi di programmazione, i compilatori e gli interpreti.

Ma cos’è un linguaggio di programmazione?

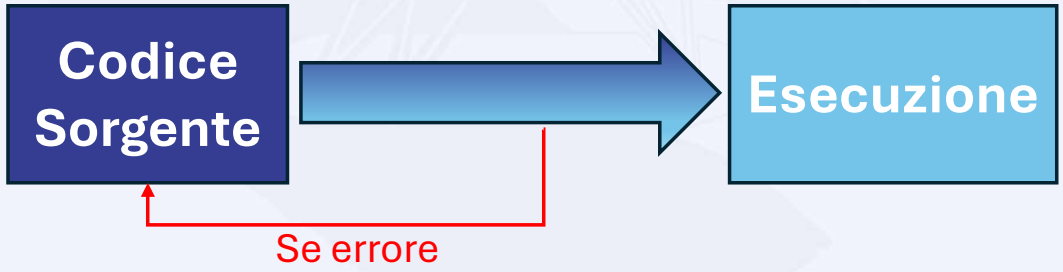
Sono linguaggi formali che tramite un insieme di algoritmi, regole sintattiche e logiche permettono di comunicare con la macchina.

Cos ‘è un Compilatore?

È un programma per computer che converte il codice scritto dal programmatore in linguaggio macchina e successivamente lo esegue.

Cos ‘è un Interprete?

È un programma per computer che traduce istruzioni di alto livello in linguaggio macchina per poi eseguire nella macchina delle azioni specifiche.
Gli interpreti vengono spesso utilizzati come strumenti di *DEBUG* perché possono eseguire un singolo codice alla volta.



Compilatore VS Interprete

Compilatore	Interprete
Esegue la scansione dell'intero programma prima di tradurlo	Esegue la scansione e traduce il programma riga per riga
Accetta l'intero programma come input	Accetta una singola istruzione come input
Genera un oggetto intermedio	Non genera nessun oggetto intermedio
Richiede meno tempo di esecuzione	Richiede più tempo di esecuzione
Linguaggio C	Linguaggio Python
Se si modifica il codice sorgente bisogna scansionarlo per intero	Se si modifica il codice sorgente bisogna solo scansionare l'istruzione
Non serve ricompilare il programma per l'esecuzione	Ogni volta bisogna scansionare e tradurre al momento dell'esecuzione
Fornisce una lista di tutti gli errori del programma	Interrompe la traduzione al rilevamento di un errore e continuerà una volta risolto
Lento nel Debug	Veloce nel Debug

Linguaggio C

Esso è uno dei linguaggi più famosi e utilizzati al mondo.
Esso viene usato in:

Arduino e Board IoT

Sistemi Operativi

App e Videogiochi

Perché si studia il linguaggio C solitamente?

Il C insegna a programmare, a pensare e a come impostare il codice. Permette di imparare a gestire la memoria e i problemi di più basso livello. Una volta imparato C, esso getterà le basi per tutti gli altri linguaggi di programmazione semplificandone lo studio.

Quali sono le principali caratteristiche del Linguaggio C?

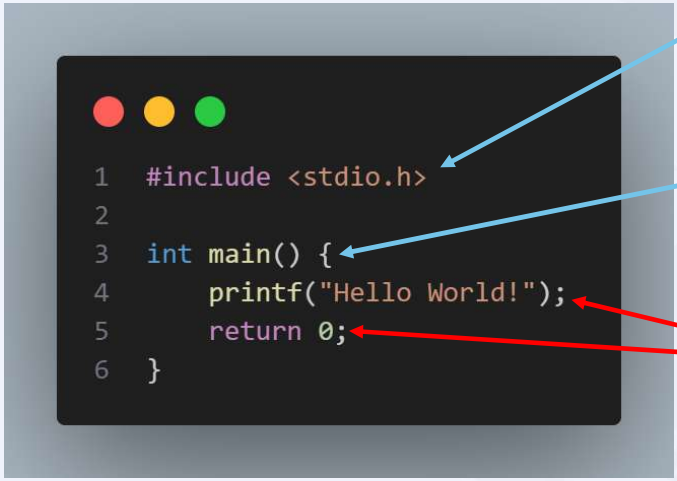
Dimensioni del codice e dell'eseguibile ridotte (Kb)

Efficienza dei programmi

Alta compatibilità di compilazione

Attività di basso livello (ES: Puntatori e memoria)

Iniziamo finalmente a programmare!



Libreria:
Il comando #include permette di richiamare le librerie standard.

Funzione:
int main() è la funzione principale di qualsiasi programma in C.

Punto e virgola:
Serve per concludere un'istruzione e fa capire al compilatore che dopo quel simbolo c'è una nuova istruzione.

```
gcc nomeFile.c –o nomeFile.out
./nomeFile.out
```

Un commento critico è sempre meglio di niente!

```c
#include <stdio.h>

int main() {
    //printf("Hello World!");
    int a = 5;
    /*
    Ciao, sono un commento!
    */
    return 0;
}
```

Una pratica molto comune nella programmazione è il commentare il codice, significa aggiungere frasi ignorate dal compilatore che servono al programmatore per leggere il codice.

Commento //:
Il comando // commenta la riga che segue.

Commento /* Testo */:
Il comando /* commenta tutto ciò che segue, quindi anche più righe, fino al */.

Quali librerie dovrei usare?

Le librerie standard di C forniscono al programmatore delle funzioni utili per facilitare la scrittura del programma.

<stdio.h>:
Fornisce le funzionalità basilari di input/output, come printf e scanf.

<math.h>:
Fornisce funzioni matematiche più avanzate.

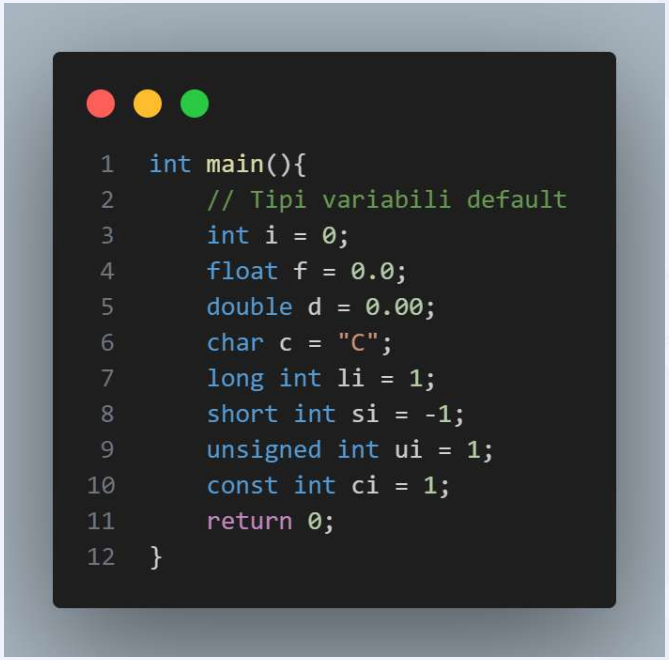
<string.h>:
Aggiunge il tipo String e metodi per la loro manipolazione.

<time.h>:
Aggiunge metodi per gestire date e orari.

<stdlib.h>:
Aggiunge metodi per gestire la memoria locale, svolgere conversioni ed altro ancora.

<windows.h> o <unistd.h>:
Aggiungono funzioni per sfruttare le Windows/Linux API, come la funzione sleep();

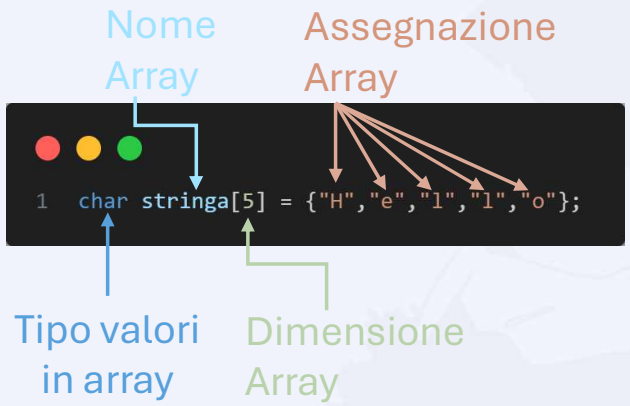
Tipi di Variabile



```c
int main(){
    // Tipi variabili default
    int i = 0;
    float f = 0.0;
    double d = 0.00;
    char c = "C";
    long int li = 1;
    short int si = -1;
    unsigned int ui = 1;
    const int ci = 1;
    return 0;
}
```

int:
Numeri interi a 16 Bit.
float:
Numeri a virgola mobile a precisione singola, 32 Bit.
double:
Numeri a virgola mobile a precisione doppia, 64 Bit.
char:
variabile che contiene un carattere, 8 Bit.
long:
Amplia il dominio dei numeri, ad esempio per gli int da 16 a 32 Bit.
short:
Riduce il dominio dei numeri.
unsigned:
Indica che la variabile ha solo segno positivo.
const:
Dichiara la variabile come costante, cioè non modificabile durante l'esecuzione.

Array o Vettori:




```c
int main(){
    // Tipi di array
    int array[3]; // I valori vanno da 0 a 2
    char stringa[] = {"H","e","l","l","o"};
    i[2] = 45;
    int a = array[2];
    return 0;
}
```

Un array, detto anche vettore in italiano, è un insieme di variabili indipendenti e si dichiara definendo il tipo che contiene (non può avere variabili miste) e la sua dimensione. Si possono assegnare direttamente gli elementi, in quel caso l'array prenderà la dimensione di assegnazione.
La dimensione dell'array non è modificabile durante l'esecuzione, per questo viene detto **statico**.

REGOLA DEL PROGRAMMATORE:

```c
char stringa[5] = {"H","e","l","l","o"};
    stringa[0] = "H";
    stringa[1] = "e";
    stringa[2] = "l";
    stringa[3] = "l";
    stringa[4] = "o";
```

Stringhe:

```c
int main(){
    // Tipi di array
    char stringa[] = {"H","e","l","l","o"};
    return 0;
}
```

Se sfruttiamo la libreria <*string.h*>

```c
#include <string.h>
int main(){
    // Tipi di array
    int array[3];
    string stringa = "Hello";
    return 0;
}
```

Come avrete notato non esistono le stringhe in C, obbligando l'utilizzo di Array ed algoritmi iterativi. Tuttavia ci viene in soccorso la libreria <string.h>

```c
//Compara due stringhe
    int Ris = strcmp(Stringa1,Stringa2);
```

```c
//Concatenare due stringhe
    string Concat = strcat(Stringa1,Stringa2);
```

```c
//Copiare una stringa in un'altra
    Stringa = strcpy(StringaDest,StringaSRC);
```

```c
//Lunghezza stringa
    int dim = strlen(Stringa);
```

Libreria <stdio.h>:

```c
#include <stdio.h>

int main(){
    int a = 0;
    int b = 0;
    int somma = 0;
    printf("Inserire primo valore: ");
    scanf("%d",&a);
    printf("Inserire secondo valore: ");
    scanf("%d",&b);

    somma = a + b;
    printf("La somma è %d\n", somma);
    return 0;+
}
```

printf()

Istruzione che stampa in console il messaggio tra ()

```c
printf("La somma è %d\n", somma);
```

Messaggio da inviare in output

Specificatore di formato

Sequenze di Escape

scanf()

Istruzione che richiede in input il valore della variabile

```c
scanf("%d",&a);
```

Specificatore di formato

Andress Operator

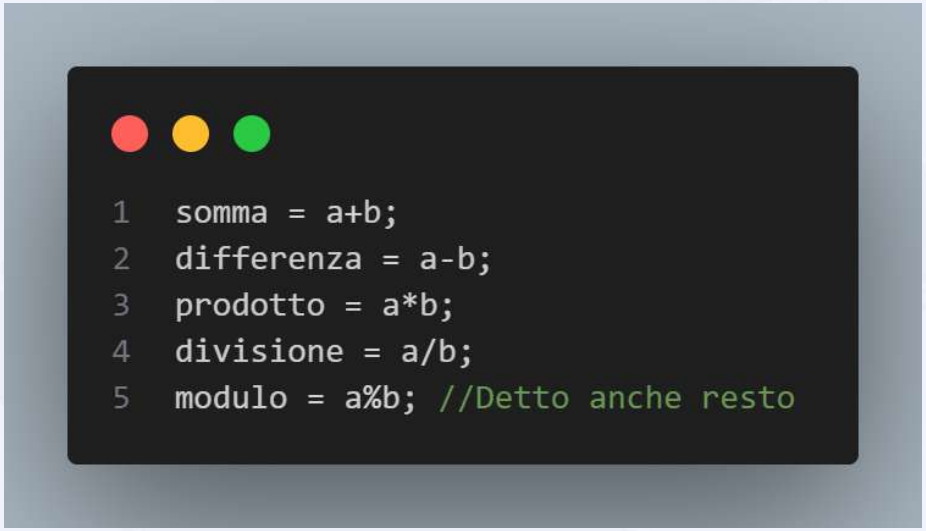
Specificatori di Formato

```
scanf("%c",&a); // Caratteri
scanf("%s",&a); // Stringhe
scanf("%hi",&a); // Short int
scanf("%hu",&a); // Short unsigned int
scanf("%Lf",&a); // Long double
scanf("%d",&a); // int
scanf("%f",&a); // float
scanf("%u",&a); // unsigned int
```

Sequenze di Escape

```
printf("\b"); // Backspace
printf("\n"); // Newline
printf("\t"); // Tab orizzontale
printf("\v"); // Tab verticale
printf("\\"); // Backslash
printf("\'"); // Apostrofo
printf("\""); // Virgolette
printf("\?"); // Punto interrogativo
```

Operatori Matematici



```
1  somma = a+b;
2  differenza = a-b;
3  prodotto = a*b;
4  divisione = a/b;
5  modulo = a%b; //Detto anche resto
```

Esercizio 1: Scrivere un programma che dati in input 2 numeri faccia le 4 operazioni fondamentali

Esercizio 2: Compilare il programma ed eseguirlo con 2 numeri a vostra scelta, successivamente date come secondo valore = 0

```
gcc nomeFile.c –o nomeFile.out
./nomeFile.out
```

Esercizio 1: Scrivere un programma che dati in input 2 numeri faccia le 4 operazioni fondamentali

```c
#include <stdio.h>

int main(){
    float a = 0, b = 0, s = 0, dif = 0, p = 0, div = 0;
    printf("Inserire primo valore: ");
    scanf("%f",&a);
    printf("Inserire secondo valore: ");
    scanf("%f",&b);
    s = a+b;
    dif = a-b;
    p = a*b;
    div = a/b;
    printf("a+b=%f\na-b=%f\na*b=%f\na/b=%f\n",s,dif,p,div);
    return 0;
}
```

Esercizio 2: Compilare il programma ed eseguirlo con 2 numeri a vostra scelta, successivamente date come secondo valore = 0

```
Inserire primo valore: 2
Inserire secondo valore: 3
a+b=5.000000
a-b=-1.000000
a*b=6.000000
a/b=0.666667
```

```
Inserire primo valore: 0
Inserire secondo valore: 0
a+b=0.000000
a-b=0.000000
a*b=0.000000
a/b=-nan
```

gcc nomeFile.c –o nomeFile.out
./nomeFile.out

I costrutti Condizionali

```
1  if(/*Condizione*/) //Istruzione
2  else //Istruzione
```

```
1  if(/*Condizione*/) {
2      //Istruzioni
3  }
4  else {
5      //Istruzioni
6  }
```

```
1  if(/*Condizione*/) {
2      //Istruzioni
3  }
4  else if(/*Condizione*/) {
5      //Istruzioni
6  }
7  else {
8      //Istruzioni
9  }
```

Operatori Logici

```
1  a>b // Maggiore
2  a>=b // Maggiore Uguale
3  a<b // Minore
4  a<=b // Minore Uguale
5  a==b // Uguale
6  a!=b // Diverso
7  a>b && b>c // AND
8  a>b || b>c // OR
9  !(/*Condizione*/) // NOT
```

Esempio:

```c
//Trovare la relazione che c'è tra due numeri in input
#include <stdio.h>
int main(){
    float a = 0, b = 0;
    printf("Inserire primo valore: ");
    scanf("%f",&a);
    printf("Inserire secondo valore: ");
    scanf("%f",&b);
    if(a>b) printf("a maggiore di b\n");
    else if(a<b) printf("a minore di b\n");
    else printf("a e b sono uguali\n");
    return 0;
}
```

Qual è la differenza tra i due?

```c
//Divisione di due numeri in input
#include <stdio.h>
int main(){
    float a = 0, b = 0;
    printf("Inserire primo valore: ");
    scanf("%f",&a);
    printf("Inserire secondo valore: ");
    scanf("%f",&b);
    if(b == 0) {
        if(a == 0) printf("Risultato Indefinito\n");
        else printf("Risultato impossibile da calcolare\n");
    }
    else printf("a/b=%f\n", a/b);
    return 0;
}
```

```c
//Divisione di due numeri in input
#include <stdio.h>
int main(){
    float a = 0, b = 0;
    printf("Inserire primo valore: ");
    scanf("%f",&a);
    printf("Inserire secondo valore: ");
    scanf("%f",&b);
    if(b == 0 && a == 0) printf("Risultato Indefinito\n");
    else if(b == 0) printf("Risultato impossibile da calcolare\n");
    else printf("a/b=%f\n", a/b);
    return 0;
}

```

I costrutti Iterativi

```
1  do {
2      //Istruzioni
3  }while(/*Condizioni*/);
```

```
1  while(/*Condizioni*/){
2      //Istruzioni
3  }
```

```
1  for(/*Assegnazione Variabile*/;/*Condizioni*/;/*Operazioni Matematiche*/) {
2      //Istruzioni
3  }
```

I seguenti codici fanno la stessa cosa?

```c
#include <stdio.h>
int main(){
    int i = 0;
    do {
        printf("i = %d\n",i);
        i++;
    }while(i<10);
    return 0;
}
```

```c
#include <stdio.h>
int main(){
    int i = 0;
    while(i<10){
        printf("i = %d\n",i);
        i++;
    }
    return 0;
}
```

```c
#include <stdio.h>
int main(){
    for(int i = 0; i<10; i++) {
        printf("i = %d\n",i);
        i++;
    }
    return 0;
}
```

I seguenti codici fanno la stessa cosa?

```c
#include <stdio.h>
int main(){
    int i = 0;
    do {
        printf("i = %d\n",i);
        i++;
    }while(i<10);
    return 0;
}
```

```
i = 0
i = 1
i = 2
i = 3
i = 4
i = 5
i = 6
i = 7
i = 8
i = 9
```

```c
#include <stdio.h>
int main(){
    int i = 0;
    while(i<10){
        printf("i = %d\n",i);
        i++;
    }
    return 0;
}
```

```
i = 0
i = 1
i = 2
i = 3
i = 4
i = 5
i = 6
i = 7
i = 8
i = 9
```

```c
#include <stdio.h>
int main(){
    for(int i = 0; i<10; i++) {
        printf("i = %d\n",i);
        i++;
    }
    return 0;
}
```

```
i = 0
i = 2
i = 4
i = 6
i = 8
```

Esercizio 1: Scrivere la calcolatrice migliorata

Esercizio 2: Scrivere un programma che sommi n numeri in input dall'utente, dove n viene sempre scelto dall'utente

Esercizio 3: Scrivere un programma che sommi numeri in input dall'utente finché non digita 0

```
gcc nomeFile.c –o nomeFile.out
./nomeFile.out
```

Esercizio 1: Scrivere la calcolatrice migliorata

```c
#include <stdio.h>

int main(){
    float a = 0, b = 0, s = 0, dif = 0, p = 0, div = 0;
    printf("Inserire primo valore: ");
    scanf("%f",&a);
    printf("Inserire secondo valore: ");
    scanf("%f",&b);
    s = a+b;
    dif = a-b;
    p = a*b;
    printf("a+b=%f\na-b=%f\na*b=%f\n",s,dif,p);
    if(b == 0 && a == 0) printf("a/b è indefinito");
    else if(b == 0) printf("a/b non calcolabile");
    else {
        div = a/b;
        printf("a/b=%f",div);
    }
    return 0;
}
```

Esercizio 2: Scrivere un programma che sommi n numeri in input dall'utente, dove n viene sempre scelto dall'utente

```c
#include <stdio.h>

int main(){
    int n = 0;
    int somma = 0;
    int a = 0;
    printf("Quanti numeri vuoi sommare?\n");
    scanf("%d",&n);
    for(int i=0;i<n;i++) {
        printf("Numero %d: ", i+1);
        scanf("%d",&a);
        somma+=a;
    }
    printf("Somma = %d\n",somma);
    return 0;
}
```

Esercizio 3: Scrivere un programma che sommi numeri in input dall'utente finché non digita 0

```c
#include <stdio.h>

int main(){
    int n = 0;
    int somma = 0;
    do{
        printf("Numero da sommare: ");
        scanf("%d",&n);
        somma+=n;
    } while(n != 0);
    printf("Somma = %d\n",somma);
    return 0;
}
```

Le funzioni

Nel linguaggio C le funzioni ritornano sempre un valore, se avessimo necessità di non dover ritornare nulla allora si userà il tipo *void*.
Per definire il valore di ritorna si usa l'istruzione *return*.
Non è necessario che la funzione accetti parametri di input.

```
1  ValoreDiRitorno NomeFunzione (/*Parametri*/) {
2      //Istruzioni
3      return Variabile;
4  }
```

```
1  int somma (int a, int b) {
2      s = a+b;
3      return s;
4  }
```

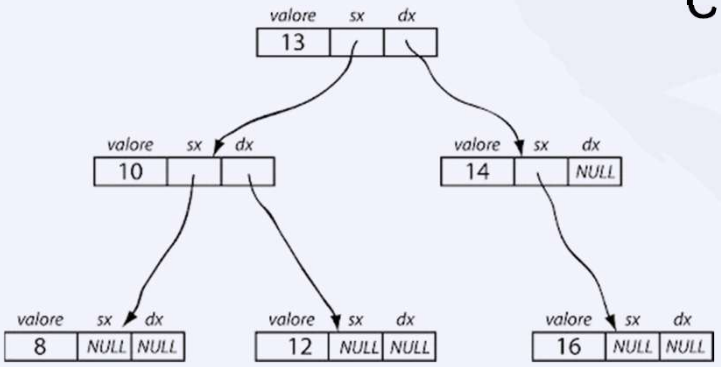
```
1  void somma (int a, int b) {
2      s = a+b;
3      printf("Somma = %d\n",s);
4      return;
5  }
```

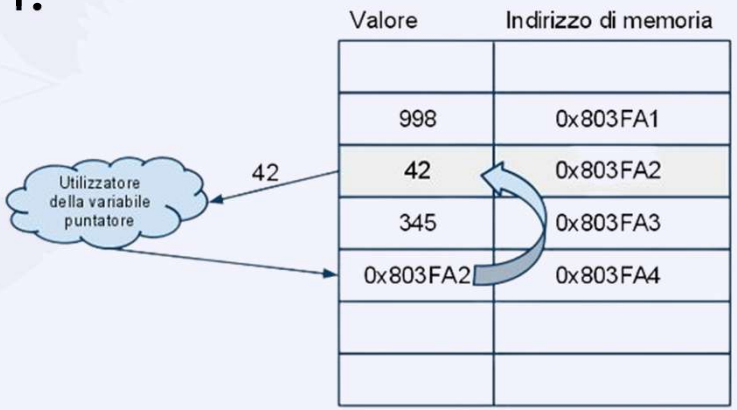
```
1  int somma () {
2      s = 3+5;
3      return s;
4  }
```

Puntatori

Uno dei fondamenti su cui si basa la programmazione in C sono i puntatori: variabili che salvano l'indirizzo di memoria di altre variabili.

Esse permettono di lavorare a più basso livello con gli indirizzi fisici della macchina e di creare strutture dati più complesse come liste ed alberi.





Puntatori

```c
int variabile = 0;

int *puntatore = &variabile;
```

```c
#include <stdio.h>

int main(){
    int variabile = 0;

    int *puntatore = &variabile;

    printf("Variabile = %d\n", variabile);
    printf("*Puntatore = %d\n", *puntatore);
    printf("&Variabile = %d\n", &variabile);
    printf("Puntatore = %d\n", puntatore);
    printf("&Puntatore = %d\n", &puntatore);

    return  0;
}
```

```
Variabile = 0
*Puntatore = 0
&Variabile = 635235580
Puntatore = 635235580
&Puntatore = 635235584
```

Puntatori ed Array

Una caratteristica molto utile dei puntatori è con gli array, essi permettono di lavorare gli array nelle funzioni tramite l'algebra dei puntatori (che non approfondiremo).

```c
int array[5] = {0,1,2,3,4};
int *puntatore;
puntatore = &array[0];
/*Tramite l'algegra dei puntatori avremo che:
puntatore = array[0]
puntatore + 1 = array[1]
puntatore + 2 = array[2]
puntatore + 3 = array[3]
puntatore + 4 = array[4]
Quindi generalizzando:
usare il puntatore+i equivale a usare array[i]
*/
```

Esercizio 1: Scrivere la calcolatrice migliorata sfruttando le funzioni

Esercizio 2: Scrivere un programma che sommi n numeri in input dall'utente, dove n viene sempre scelto dall'utente, sfruttando le funzioni

Esercizio 3: Scrivere un programma che sommi numeri in input dall'utente finché non digita 0 sfruttando le funzioni

gcc nomeFile.c –o nomeFile.out
./nomeFile.out

Esercizio 1: Scrivere la calcolatrice migliorata sfruttando le funzioni

```c
#include <stdio.h>
float Somma(float a, float b);
float Differenza(float a, float b);
float Prodotto(float a, float b);
void Divisione(float a, float b);
int main(){
    float a = 0, b = 0, somma = 0, differenza = 0, prodotto = 0;
    printf("Inserire primo valore: ");
    scanf("%f",&a);
    printf("Inserire secondo valore:");
    scanf("%f",&b);
    somma = Somma(a,b);
    differenza = Differenza(a,b);
    prodotto = Prodotto(a,b);
    printf("Somma = %f\nDifferenza = %f\nProdotto = %f\n",somma,differenza,prodotto);
    Divisione(a,b);
    return 0;
}
float Somma(float a, float b){
    return a+b;
}
float Differenza(float a, float b){
    return a-b;
}
float Prodotto(float a, float b){
    return a*b;
}
void Divisione(float a, float b){
    if(b==0 && a == 0) printf("Divisione non definita\n");
    else if(b==0) printf("Divisione non calcolabile\n");
    else printf("Divisione = %f\n",a/b);
}
```

Esercizio 2: Scrivere un programma che sommi n numeri in input dall'utente, dove n viene sempre scelto dall'utente, sfruttando le funzioni

```c
#include <stdio.h>
int Sommatore(int n){
    int s = 0;
    int a = 0;
    for(int i = 0; i<n; i++) {
        printf("Numero %d: ", i+1);
        scanf("%d",&a);
        s+=a;
    }
    return s;
}
int main(){
    int n = 0;
    printf("Quanti numeri vuoi sommare?\n");
    scanf("%d",&n);
    printf("Somma = %d\n",Sommatore(n));
    return 0;
}
```

Esercizio 3: Scrivere un programma che sommi numeri in input dall'utente finché non digita 0 sfruttando le funzioni

```c
#include <stdio.h>
void Sommatore (int *somma) {
    int n = 0;
    do{
        printf("Numero da sommare: ");
        scanf("%d",&n);
        *somma += n;
    }while(n != 0);
}

int main(){
    int somma = 0;
    Sommatore(&somma);
    printf("Somma = %d\n",somma);
    return 0;
}
```

Esercizi per casa

Esercizio 1: Scrivere la calcolatrice con un menù iterativo e che si chiuda una volta che l'utente sia soddisfatto
Esercizio 2: Scrivere le funzioni che restituiscano il minimo, il valor medio e il massimo di un array. Scrivere un menù che sfrutti tali funzioni e le funzioni utili all'uso.
Esercizio 3: Scrivere una funzione che ordini un array dato in input di 10 valori.
Esercizio 4: Scrivere una funzione che suddivida un array in due, nel primo array andranno i valori sotto al valor medio, nel secondo array andranno i valori sopra al valor medio.

‘’Fine’’ Lezione sul linguaggio C

Grazie per aver seguito fino alla fine.

La prossima lezione sarà su Python con il nostro carissimo **Gabriele**, buona fortuna!

Se avete dubbi o volete qualche approfondimento, chiedete pure, mi potete contattare anche su Telegram a **@Diego_C3**