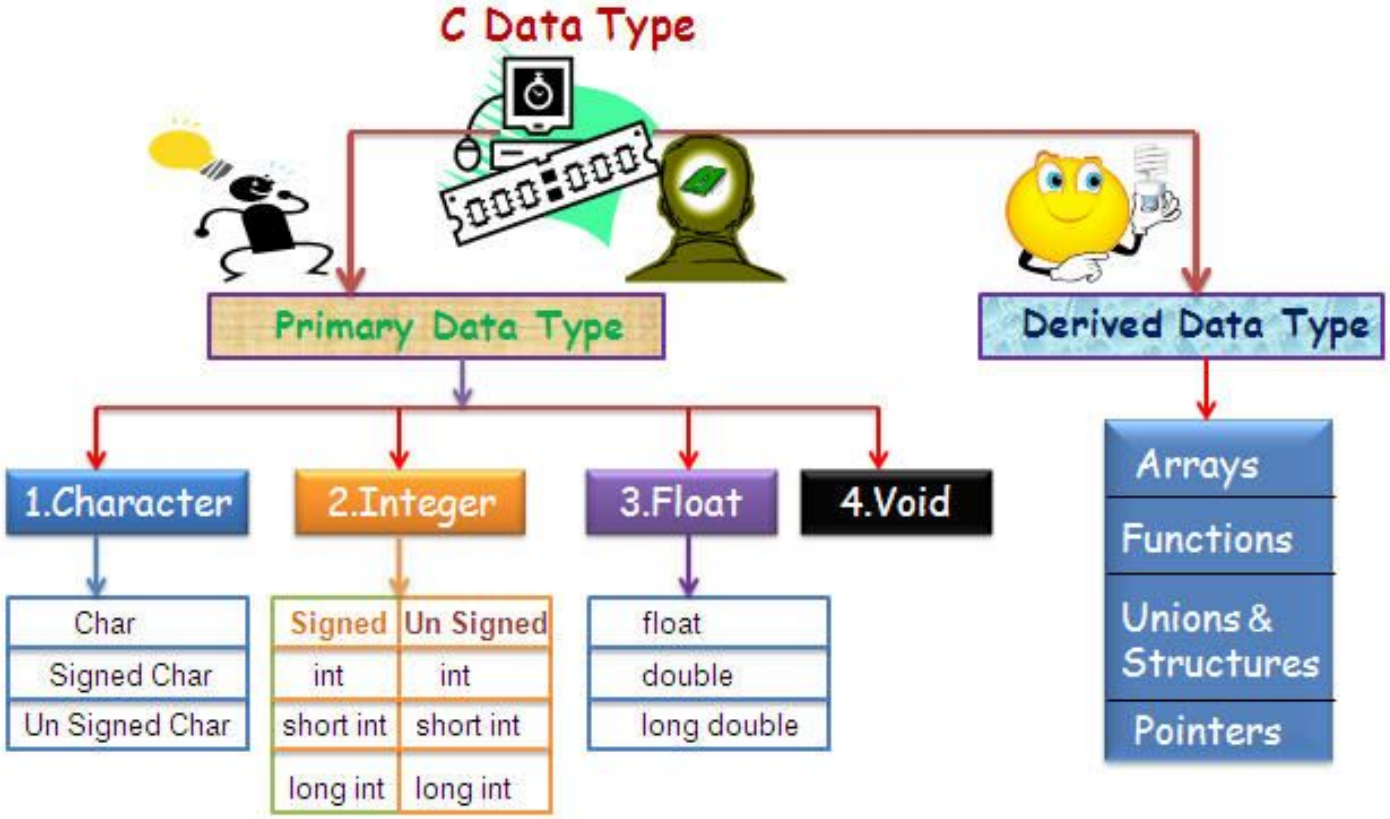
Dati del C

e funzioni di I/O

Tipi del C: [tutorialspoint.com]



A ogni Tipo la sua Dimensione

DATA TYPE	SIZE (IN BYTE)
char	1
short int	2
int	4
long int	4
float	4
double	8
long double	12
void	MEANING LESS

```c
#include <stdio.h>
int main()
{ /****************** DECLARATIVE SECTION ****************************/
    int a;          //Declaration
    a = -1;         //Assignment
    char b ='G';    //Initialization = Declare + Assign
    double c = 3.14;

  /***************** PROGRAM *********************************/
                              //printing the variables defined above along with their sizes
    printf("Hello! I am a character. My value is %c and "
           "my size is %zu byte.\n", b,sizeof(char));       //can use sizeof(b) as well

    printf("Hello! I am an integer. My value is %d and "
           "my size is %zu  bytes.\n", a,sizeof(int));      //can use sizeof(a) as well

    printf("Hello! I am a double floating point variable."
           " My value is %lf and my size is %zu bytes.\n",
                                c,sizeof(double));       //can use sizeof(c) as well

    printf("Bye! See you soon. :)\n");

    return 0;
}
```

`int printf ( const char * format, ... );`

const char * format → stringa

… → Altri parametri (0 o più)

Es:

```
printf ("I'm a String :)\n");      //solo il primo parametro
printf ("Decimals: %d \n", 1977);
printf ("Characters: %c %c \n", 'a', 65);
printf ("Preceding with blanks: %10d \n", 1977);   //      1977
printf ("Some different radices: %d %x %o %#x \n", 100, 100, 100, 100);
printf ("floats: %4.2f %+.0e %E \n", 3.1416, 3.1416, 3.1416);
printf ("%s \n", "A string");
```

Belli da paura

```
printf ("I'm a Str\bing :)\n");
printf ("Risultato: %10d \n");
printf ("Percentage character: %%",10);
printf ("%d + %d = %d\n", number, number*2);
printf ("Foo: %c", 37);
printf ("The value of number is %q\n", number);
printf ("What is % \bd format specifier ?\n");
printf ("%*d%*d\n", width, 10, width, 12);
```

Tipi Elementari del C e Specifiche di Conversione

Data Type	Memory (B)	Range	Format Specifier	
signed char	1	-128 to 127	%c	(%d)
unsigned char	1	0 to 255	%c	(%u o %d)
short int	2	-32,768 to 32,767	%hd	(%d)
unsigned short int	2	0 to 65,535	%hu	(%u o %d)
int	4	-2,147,483,648 to 2,147,483,647	%d	
unsigned int	4	0 to 4,294,967,295	%u	
long int	4	-2^31 to 2^31-1	%ld	
unsigned long int	4	0 to 2^32-1	%lu	
long long int	8	-(2^63) to (2^63)-1	%lld	(%I64d)
unsigned long long int	8	0 to 18,446,744,073,709,551,615	%llu	(%I64u)
float	4		%f	
double	8		%lf	
long double	10/12		%Lf	(!)

Esercizi

1. Studiare il codice *Es04_printf_demo.c* e facendo riferimento al manuale comprendere il significato di ciascuna printf e scriverlo a commento riga per riga.

2. Partendo dalla soluzione del sorgente *Es02_variabili.c* riorganizzare il codice in modo da ottenere una visualizzazione tabellare come quella a fianco.

```
**************************************************
          Description of every variable:
**************************************************
```

Tipo	Dim [bit]	Min	Max
Char	8 Bit	-128	127
Unsigned char	16 Bit	0	255
Short	16 Bit	-32768	32767
Unsigned short	16 Bit	0	65535
Int	32 Bit	-2147483648	2147483647
Unsigned int	32 Bit	0	4294967295
Long	32 Bit	-2147483648	2147483647
Unsigned long	32 Bit	0	4294967295
Long long	64 Bit	-9223372036854775808	9223372036854775807
Unsigned long long	64 Bit	0	18446744073709551615
Float	32 Bit	1.175494e-038	3.402823e+038
Double	64 Bit	2.225074e-308	1.797693e+308
Long double	128 Bit	3.464199e-317	3.464191e-317

`int scanf ( const char * format, ... );`

Es:

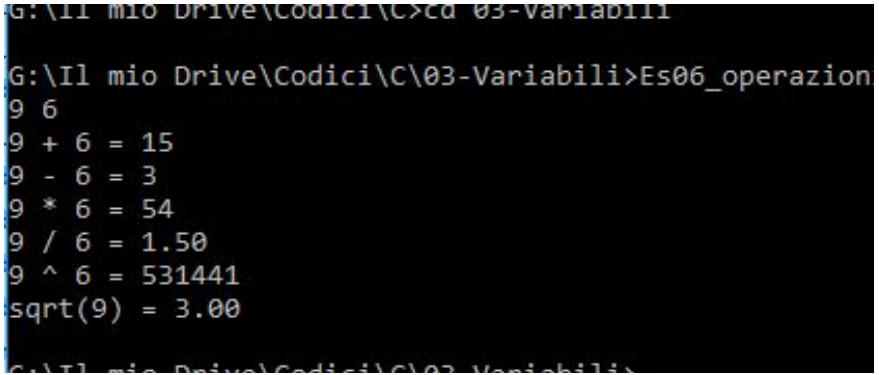
```
char name[100];     //alloca 100B contigui in memoria
int i;              //se la stringa e' + lunga il programma va in crash!!!
printf ("Enter your name and your age: ");
scanf ("%s",&name);
scanf ("%d",&i);
printf ("Mr. %s , %d years old.\n",str,i);
printf ("Enter a hexadecimal number: ");
scanf ("%x",&i);
printf ("You have entered %#x (%d).\n",i,i)   //come è memorizzato il numero?
```

Come sono dichiarati **`name`** and **`i`**?

Esercizi

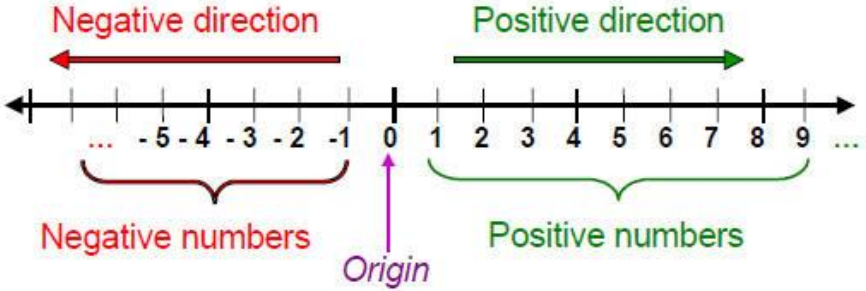
1. Scrivere programma C *Es05_messaggio.c* che:
   - richiede nome e età (per esempio Ugo 16)
   - stampa il messaggio: **`Ugo è nato nel 2002`**

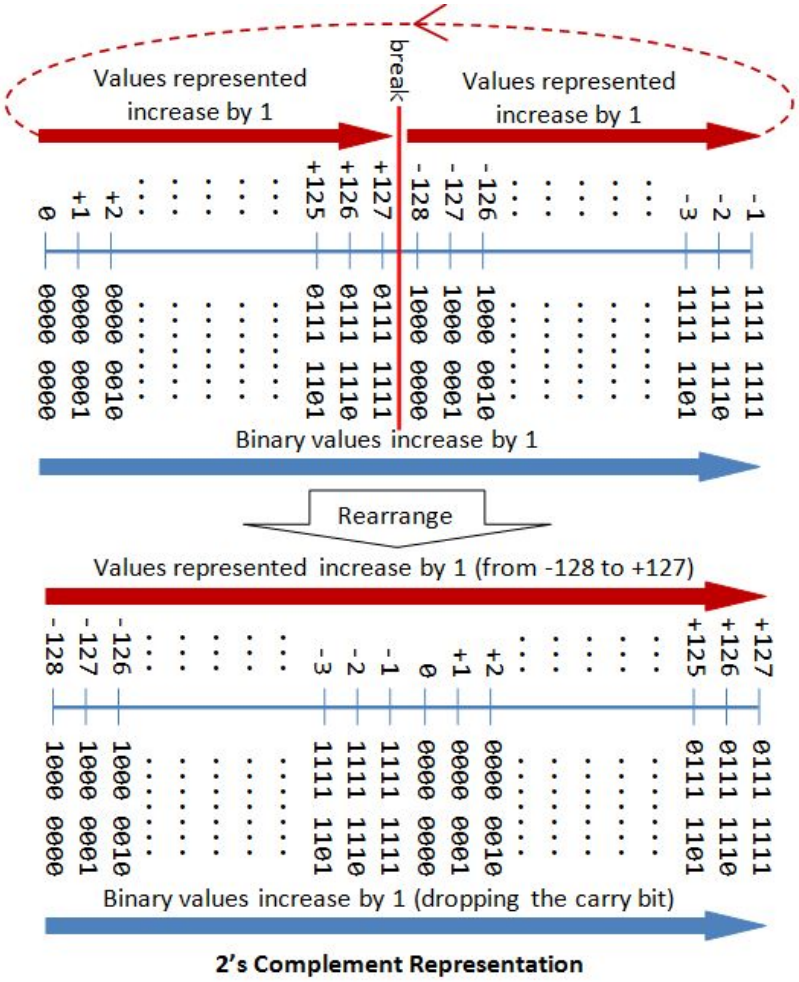
2. Scrivere programma C *Es06_operazioni.c* che:
   - richiede due numeri A e B
   - stampa le 6 operazioni possibili con i relativi risultati (A+B, A-B, A*B, A/B, A^B, sqrt(A))
     NB: per potenza e radice dovrete attingere da Math.h (lascio a voi di scovare le giuste funzioni)



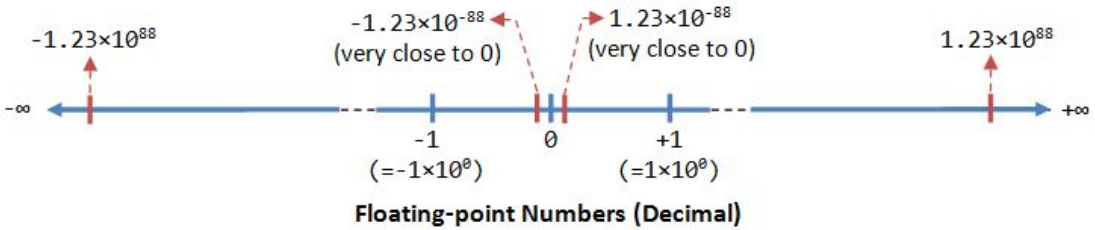
Ripasso sulle codifiche

Attenti ai CHAR:

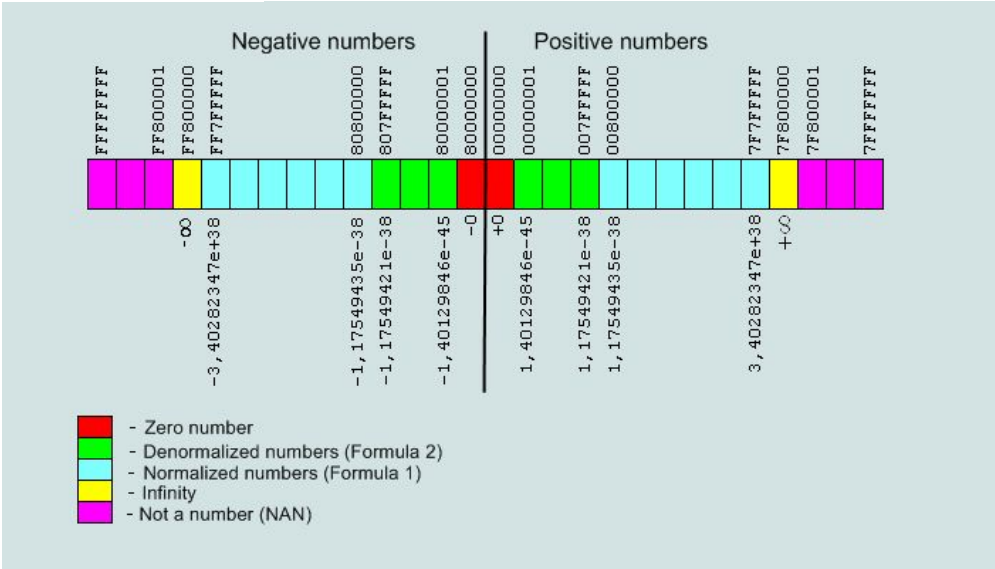




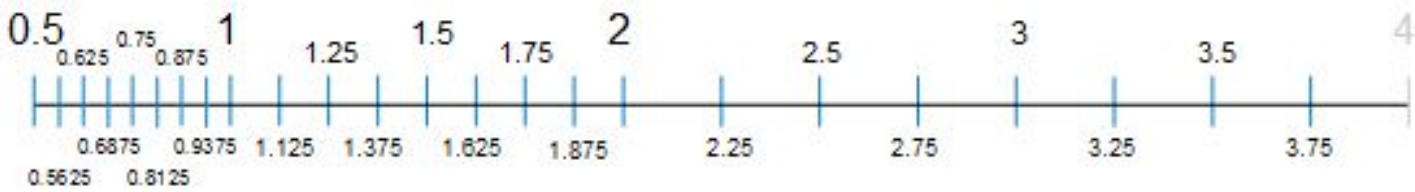
Ripasso sulle codifiche: ATTENTI A QUEI FLOAT!!



Floating-point Numbers (Decimal)

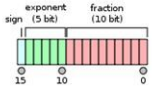


IEEE 754 Density

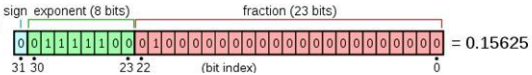


Other IEEE 754 Formats

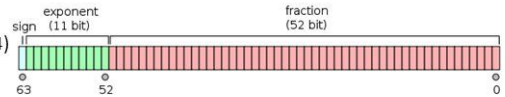
Half precision (binary16)



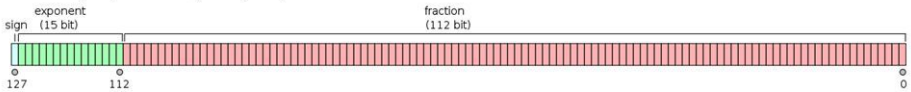
Single precision (binary32)



Double precision (binary64)



Quadruple precision (binary128)



Altre funzioni per l'I/O del C

Per l'input:

- `int getchar ( void );`
- `char * gets ( char * str );`

Per l'output:

- `int putchar ( int character );`
- `int puts ( const char * str );`

Esempio

```
#include <stdio.h> //
int main()
{
      char ch;

      puts("Inserire un carattere e premere INVIO:");
      while((ch=getchar())=='\n');

      printf("Il carattere inserito e':");
      putchar(ch);   //....    warning?
      printf("(%#02x)", ch);
      return 0;
```

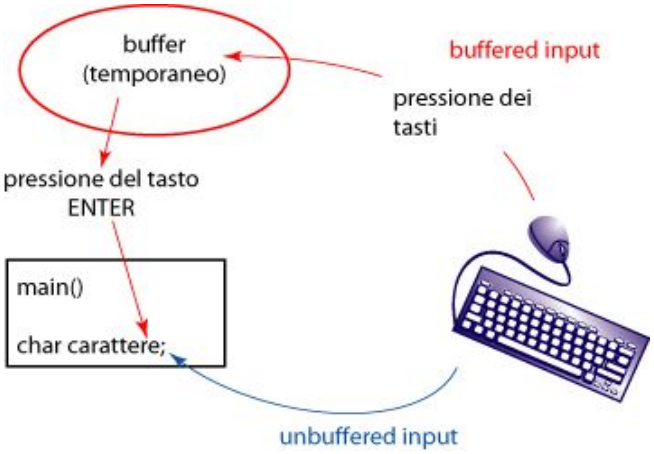
Modificare il sorgente in modo che continui a richiedere un carattere **all'infinito**.

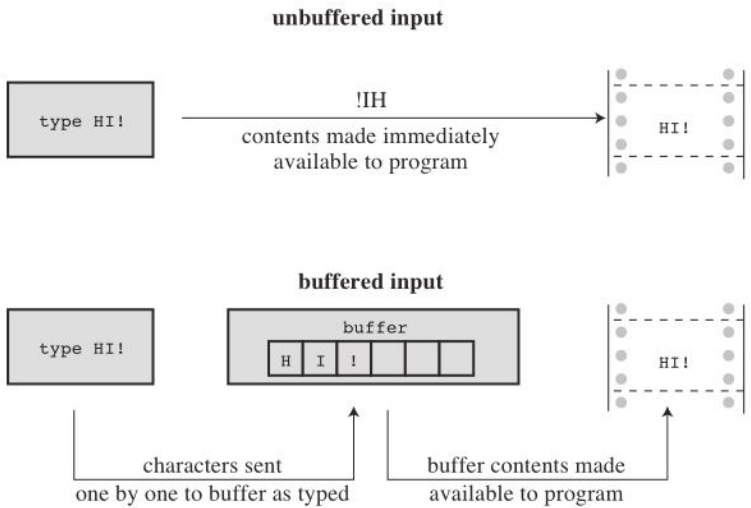
Input bufferizzato

```
#include <stdio.h> //
int main(void)
{
    int ch;

    while((ch = getchar()) != EOF)
        getchar();

    return 0;
}
```





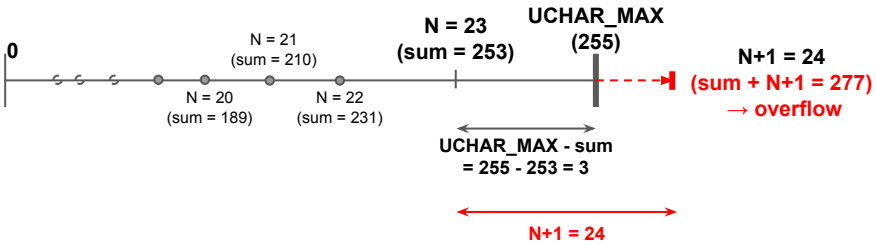
Esercizi

1. Dato un carattere in input stampa il relativo codice ASCII (loop).
2. Stampa la tavola ASCII riportando per ciascun carattere: simbolo, valore hex, valore dec.
3. Dato un numero N in input, stampare la somma dei primi N numeri interi.
4. Stampare il numero massimo Nmax di interi sommabili per ciascun tipo di dato.
5. Dato un numero intero N, stampare tutti i suoi divisori.

```
H:\MAXSUM.exe
--------------------------------------------------------
                      TYPE ANALYZER
--------------------------------------------------------
[1] Char
[2] Unsigned Char
[3] Short
[4] Unsigned Short
[5] Int
[6] Unsigned Int
[7] Long Long
[8] Unsigned Long Long
[0] ... to Exit
Please, select the C data type to analyze: 1

Nmax CHAR = 16
Somma CHAR = 120
```

Suggerimento: consideriamo UCHAR



quindi, la condizione di termine è `UCHAR_MAX - sum < N+1`