Introduzione alla Programmazione

claudio.lucchese@unive.it

I tipi fondamentali

1.	**char**	1 Byte	da -128 a +127
2.	signed char	1 Byte	da -128 a +127
3.	unsigned char	1 Byte	da 0 a 255
4.	signed short int o short int o **short**	2 Byte	da -32768 a 32767
5.	unsigned short int o unsigned short	2 Byte	da 0 a 65535
6.	signed int o **int**	4 Byte	da -2147483648 a +2147483647
7.	unsigned int o unsigned	4 Byte	da 0 a 4294967295
8.	signed long int o long int o **long**	8 Byte	da -2^{63} a $2^{63}-1$
9.	unsigned long int o unsigned long	8 Byte	da 0 a $2^{64}-1$
10.	**float**	4 Byte	da -3.40×10^{-38} a 3.40×10^{38}
11.	**double**	8 Byte	da -1.80×10^{-308} a 1.80×10^{308}
12.	long double	dipende dall'implementazione …	

Language Reference

https://en.cppreference.com/w/c/language/basic_concepts

Output Stream

Documentazione: https://en.cppreference.com/w/c/io/fprintf

printf (“formato”, expr_1, expr_2, …);

“formato” consiste in una stringa scelta dell’utente che viene scritta in output senza modifiche tranne per le codifiche che iniziano per %:

- **%c** per char e unsigned char
- **%d** per short, int; **%ld** per long
- **%u** per unsigned short, unsigned int, **%lu** per unsigned long
- **%f** per float e double

Ad ogni codifica %? in “formato” deve corrispondere una espressione in expr_1, expr_2, ...

Output Stream

```c
#include  <stdio.h>

int main(void)
{
    int i = -1;
    unsigned int ui = 100;
    float f = 3.1415927;

    printf("intero %d, senza segno %u, float %f", i, ui, f);
    printf("\n");

    printf("intero %4d, senza segno %4u, float %.2f", i, ui, f);
    printf("\n");

    return 0;
}
```

Input Stream

Documentazione: https://en.cppreference.com/w/c/io/fscanf

scanf (“formato”, receiving_argument_1, receiving_argument_2, …);

“formato” consiste in una stringa scelta dell’utente che viene “matchata” in input, i valori da memorizzare nei receiving arguments sono identificati con la codifica:

- **%c** per char e unsigned char
- **%d** per int; **%hd** per short; **%ld** per long
- **%f** per float, **%lf** per double

Ad ogni codifica %? in “formato” deve corrispondere un receiving_argument.

I receiving arguments si specificano con il prefisso **&**, es. &var.

Input Stream

```c
#include <stdio.h>

int main(void) {

  char c;
  int num;
  float value;

  scanf("%c", &c);
  scanf("%d", &num);
  scanf("%f", &value);

  printf("char : %d \n", c);
  printf("char : %c \n", c);
  printf("integer : %d \n", num);
  printf("float   : %f \n", value);

  return 0;
}
```

Espressioni

Le espressioni sono combinazioni valide di costanti, variabili, operatori e chiamate di funzione.

Costanti numeriche: 10 (int) 10**u** (unsigned) 10**l** (long)
10.0 (double) 10.0**f** (float) 10.0**d** (double)

Operatori aritmetici: **() + - * / %**

Operatori relazionali e di uguaglianza: **< <= > >= == !=**

Operatori logici: **&& || !**

Per ogni operatore è definita la precedenza e l'associatività.

Es. di espressione: (-2 + (2*3-4*3+2)/7) / (2*1)

Esercizio

- Scrivere un programma C che dato un numero di giorni specificato dall’utente, calcola a quanti anni, settimane, giorni corrispondono
  - Esempio: 373 giorni => 1 anno, 1 settimana e 1 giorno.

if-then-else statement

```c
/* Il mio primo programma */

#include <stdio.h>
#include <math.h>

int main(void)
{

    int x = 99;

    if (x>0) {
        printf("X è positivo! \n");
    } else {
        printf("Y è negativo! \n");
    }
    printf ("fine!\n");

  return 0;
}
```

if (*expr****)*** *statement_1*

oppure

if (*expr****)*** *statement_1* ***else*** *statement_2*

Se *expr* viene interpretata come **vera**,
viene eseguito lo *statement_1*

Quando il ramo else è presente,
se expr viene interpretata come **falsa**,
viene eseguito lo *statement_2*

In ogni caso l'esecuzione continua con gli **statement successivi** allo if.

Lo statement può essere un compound statement, ovvero un "blocco" delimitato da graffe di dichiarazione e statements.

Operatori relazionali, di uguaglianza, logici

Operatori relazionali e di uguaglianza: x**<**y x**<=**y x**>**y x**>=**y x**==**y x**!=**y

Operatori logici: exrp_1 **&&** exrp_2 (and logico) exrp_1 **||** exrp_2 (or logico)
! exrp (not)

In C **0** significa **falso**, e ogni valore **diverso da 0** viene interpretato come **vero**.

Esiste il tipo **_Bool**

La libreria <stdbool.h> fornisce i sinonimi **bool** (_Bool), **true** (1) e **false** (0).

Esempio: è la variabile x compresa tra 0 e 10 ?

bool is_included = (x>=0) && (x<=10);

Esercizi

1. Trovare il massimo tra 3 numeri.
2. Verificare se l'espressione (0.1+0.1+0.1)==0.3 è vera.
3. L'assegnamento è un'espressione, qual è il suo valore?
4. Date le lunghezze di tre segmenti, dire se questi possono formare un triangolo.
5. Date le lunghezze di tre segmenti, dire se questi possono formare un triangolo rettangolo.
6. Dato in input un numero intero <=255, visualizzare la sua rappresentazione binaria