Introducción:

hello world:

`stdio.h` es un archivo que contiene definiciones, declaraciones de variables, tipos, funciones para input y output.

Usamos `\n` para indicar salto de línea. Es un "format specifier"

Con `#include<filename>` se incluye los contenidos de `filename`

```
#include<stdio.h>
main(){
    //comentario
    printf("Hello world!\n"); // Al abrir y cerrar comitas indicamos un tipo string constante
    /*
    otro comentario
    */
}
```

Compilamos:

```
$gcc hello_world.c -o hello_world.out
```

Ejecutamos:

```
$./hello_world.out
```

Definición y declaración e inicialización de variables

Una variable es un nombre que será utilizado para almacenar un tipo de dato y que reconoce nuestro programa.

Debemos definir o declarar a las variables. Para la definición y declaración de variables utilizamos palabras reservadas en C `keywords`, como: char, int, float, double. Es buena práctica inicializarlas al definirlas. Inicializar es asignar un valor inicial a la variable o al inicio del programa. Para inicializar a las variables utilizamos el símbolo de "="

```
#include<stdio.h>
/*Definición y declaración de variables*/
main(){
    char variable_char; //8 bits, entera
    int variable_int; //32 bits, entera
    float variable_float; //32 bits, punto flotante
    double variable_double; //64 bits, punto flotante
}
```

Inicialización de variables:

```
#include<stdio.h>
/*Definición y declaración  e inicialización de variables*/
main(){
    char variable_char; //8 bits, entera
    int variable_int; //32 bits, entera
    float variable_float = -2.0; //32 bits, punto flotante. Esta línea representa
                                 //definición e inicialización de variable_float.
    double variable_double; //64 bits, punto flotante
    variable_char = 'b';
    variable_int = 3;
```

```
    variable_double = 5.0;

    //Imprimir los valores
    printf("Valor de variable char: %c", variable_char);
}
```

Al definir o declarar variables indicamos su tamaño, el rango de valores que puede tomar y las operaciones que es posible aplicarles.

Al realizar operaciones debemos revisar que la definición de nuestras variables son del tipo que nosostros queremos.

```
#include<stdio.h>
main(){
    int variable_int1, variable_int2;
    double variable_double1, variable_double2, variable_double3;
    //Inicialización de variables:
    variable_int1 = 3;
    variable_int2 = -1;
    variable_double1 = 5.0;
    variable_double2 = -3.0;
    variable_double3 = 0.6789281736281947
    variable_int1 = variable_int1/variable_int2;
    printf("Variable entera divida por -1: %d\n", variable_int1);
    variable_double1 = variable_double1/variable_double2;
    printf("Variable double1 entre variable double2: %1.9f\n", variable_double1);
    //Notación exponencial
    printf("Variable double 1 entre variable double 2 notación exponencial: %1.9e\n", variable_double1)

}
```

Número más grande positivo. Incluimos el archivo `float.h`

```
#include<stdio.h>
#include<float.h>
main(){
    printf("Numéro más grande positivo: %e\n", DBL_MAX);
}
```

Underflow, Overflow. Observa el uso del tipo `L` que significa `long double`

```
#include<stdio.h>
main(){
    long double variable1 = 2.22e-326;
    long double variable2 = 1e309;
    printf("Valor de variable 1: %Le\n", variable1);
    printf("Valor de variable2 %Le\n", variable2);
}
```

Obtención del epsilon de la máquina con un while.

Observa la estructura `while`. En este ejemplo no es necesario el uso de llaves `{}` pues el statement dentro del while sólo involucra una línea.

```
#include<stdio.h>
main(){
    double variable = 1.0;
```

```
    while(1.0+variable != 1.0){
        variable = variable/2.0;
    }

    printf("Valor de epsilon de la máquina %e\n", variable);
}
```

Ejemplo para ligar librería math al compilar:

```
#include<stdio.h>
#include<math.h>
main(){
    double x_8 = 0.71428571;
    double y_8 = 0.33333333;
    double x_5;
    double y_5;
    double variable_double = -3.0;
    //función: truncf
    printf("Valor x_8 con truncf: %1.5f", truncf(x_8));
    x_5 = truncf(x_5*1e5)*1e-5;
    printf("Valor x_5: %e\n",x_5);
    printf("fabs: %f\n", fabs(variable_double));
}
```

Compilamos:

```
$gcc ejemplo_truncf.c -o ejemplo_truncf.out -lm
```