INTRODUCCIÓN RÁPIDA A C

Introducción Rápida a C

- Tarea para la siguiente clase:

http://www.learn-c.org/

Introducción Rápida a C

```
#include <stdio.h>
#define valor 75.0
int funcion(float b);
int main(int argc, char**argv){
        printf("Hola Mundo!");
        funcion(valor);
        return 0;
}
int funcion(float b){
        printf("%.2f", b);
        return 0;
}
```

Introducción Rápida a C

- Para compilar:

gcc hola.c –o hola

Introducción Rápida a C

Archivos cabecera

Terminados en .h. Declarados con la diretiva #include:

```
#include <stdio.h>

#include “milib.h”
```

Introducción Rápida a C

Archivos de implementación

Terminan en .c. Usualmente contienen código de implementaciones.

`programa.c`

Introducción Rápida a C

Definiciones

Directiva #define nos permite dar nombre a valores

```
#define NOMBRE VALOR
```

Introducción Rápida a C

Definiciones

Directiva #define nos permite dar nombre a valores

```
#define NOMBRE VALOR
```

Introducción Rápida a C

Prototipo de funciones

Toda funcion tiene un nombre y un tipo de dato que devuelve.

Pueden tomar argumentos

```
<tipo> nombreFucion(arg1, arg2, …)
```

Introducción Rápida a C

El main

Todo programa debe tener una funcion main.

```
int main(int argc, char **argv)
```

Introducción Rápida a C

Definición de funciones

La implementación de la función se hace después de de declarar prototipo:

```
<tipo> nombreFuncion(arg1, arg2, …){…}
```

Introducción Rápida a C

Variables

C usa tipado estático. Toda variable declarada debe tener un tipo:

```
<tipo> nombre;

<tipo> nombre1, nombre2;
```

Introducción Rápida a C

Variables

C usa tipado estático. Toda variable declarada debe tener un tipo:

```
<tipo> nombre;

<tipo> nombre1, nombre2;
```

Introducción Rápida a C

Variables locales y globales

- Variables locales → Definidas dentro de una función. Existen mientras se ejecuta la función.
- Variables globales→ Definidas fuera de una función. Pueden ser vistas por todas las funciones.

Introducción Rápida a C

Tipos de datos atómicos

- Enteros → **`byte`**, **`short`**, **`int`**, **`long`**. Podemos usar **`unsigned`**.
- Punto flotante→ **`float`**, **`double`**, **`long double`**.
- Caracteres → **`char`**.

Introducción Rápida a C

Arreglos

- No ofrecen verificación de limites:

```
tipo nombre[tamaño]
```

- Para acceder o cambiar un elemento

```
nombre[6] = …
```

Introducción Rápida a C

Arreglos Multidimensionales

- Especificamos los tamaños:

```
tipo nombre[tamaño1][tamaño2]
```

- Para acceder o cambiar un elemento

```
nombre[6][3] = …
```

Introducción Rápida a C

Strings

- Se declaran con punteros a char:

```
char *valor = “hola”
```

```
char valor[] = “hola”
```

- Siempre tienen carácter nulo al final (siempre tomar en cuenta para el tamaño!)
- Los podemos manejar como arreglos.

Introducción Rápida a C

Operadores

- Matematicos:
  - `+, -, *, /, %`
- Asignación
  - `=, operacion= (+=, /=, -=, *=)`
- Incremento, decremento
  - `++, --` → *var++, ++var, --var, var++*

Introducción Rápida a C

Operadores

- Relacionales:
  - `==, !=, >, <, >=, <=`
- Lógicos
  - `!, &&, ||, ?:`

```
Condicion? Valor si verdadero : valor si falso;
```

Introducción Rápida a C

Sentencias

- Toda sentencia debe terminar en punto y coma:

```
int = 2;
```

- Los bloques de sentecias se delimitan con llaves:

```
{
    sentencia 1;
    sentencia 2;
}
```

Introducción Rápida a C

Sentencias

- Toda sentencia debe terminar en punto y coma:

```
int = 2;
```

- Los bloques de sentecias se delimitan con llaves:

```
{
    sentencia 1;
    sentencia 2;
}
```

Introducción Rápida a C

Sentencia if

```
if(condicion){
    sentencia 1;
    sentencia 2;
}
else{
    sentencia 3;
}
```

Introducción Rápida a C

Sentencia if

```
if(condicion){
    sentencia 1;
    sentencia 2;
}
else{
    sentencia 3;
}
```

Introducción Rápida a C

Sentencia if

```
if(condicion){
        sentencia 1;
}
else if(condicion 2){
        sentencia 2
}
else{
        sentencia 3;
}
```

Introducción Rápida a C

Sentencia switch

```
switch(variable){
        case 1:
                sentencia 1;
                break;
        case 2:
                sentencia 2;
                break;
        default:
                break;
}
```

Introducción Rápida a C

Sentencia switch

- Switch es preferible a if anidado
- Todo caso debe tener un **break** al final (¿por qué?)
- Es buena práctica tener un **default** (manejar casos atípicos).

Introducción Rápida a C

Sentencia while

```
while( condicion) {
     sentencia 1;
     sentencia 2;
}
```

Introducción Rápida a C

Sentencia for

```
for(inicial; final; incremento) {
    sentencia 1;
    sentencia 2;
}
```

Introducción Rápida a C

Entrada/Salida

Usando la libería <stdio.h>:

```
printf(" … ", …);
```

Entrada:

```
char datos[100]

fgets(datos, 100, stdin);
```