Programación Sobre Redes

L1: Introducción a C/C++

Nicolás Mastropasqua, basado en clase Algoritmos I Dc-Fcen 2017

March 20, 2020

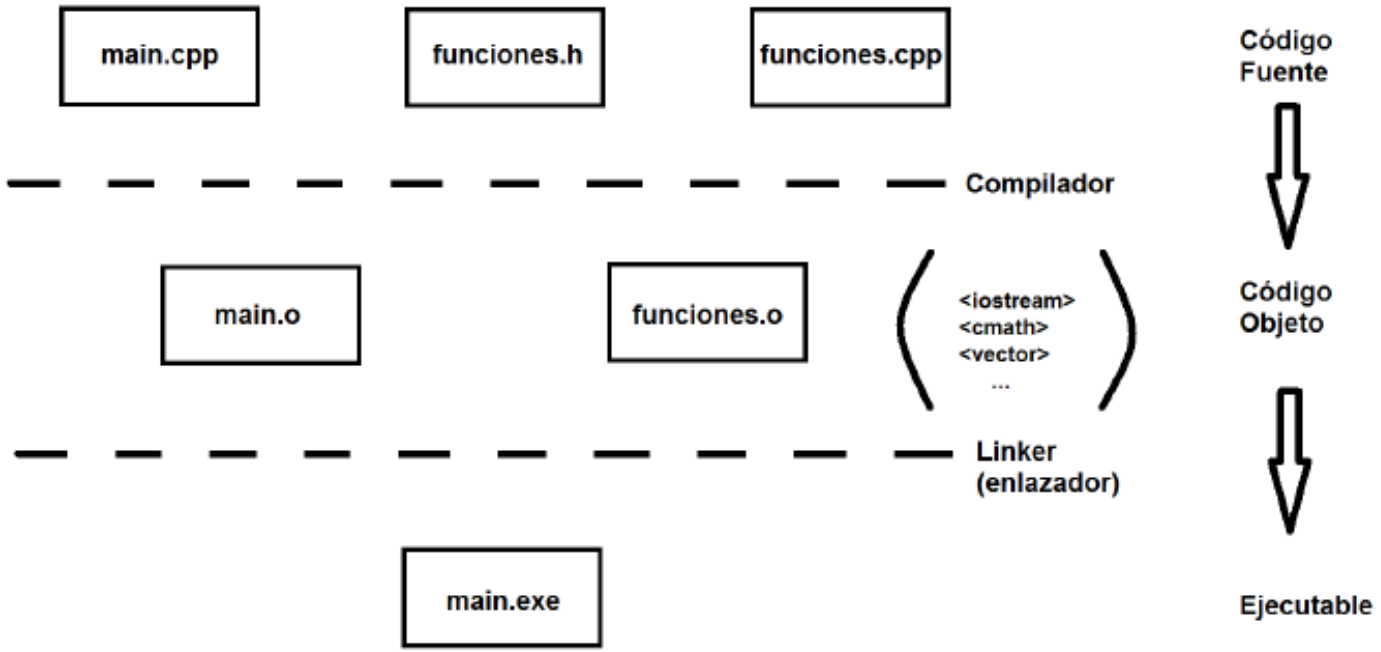
Instituto Industrial Luis A. Huergo

Contenidos

1. Compilación: Esquema general
2. Tipos y variables
3. Control de flujo
4. Funciones
5. Función Main
6. Namespaces
7. Recomendaciones

Compilación: Esquema general

Compilador y linker



Tipos y variables

Tipos

C++ cuenta con varios tipos básicos definidos

- int: Numeros enteros.
  - unsigned int: Entero no negativo.
  - long int: Entero, pero mas ”grande”
  - long long int, short int, etc...
- char: Representa un caracter.
- bool: Toma los valores verdadero (true) o falso (false).

Strings

Strings

Es posible trabajar con el tipo string en C++, haciendo el include de dicha librería. Los veremos con un poco más de cuidado en otro labo.

Más información: http://www.cplusplus.com/reference/string/string/

Variables

- En C++, las variables se referencian simplemente con una palabra, definida por el programador al momento de su declaración.

Variables

- En C++, las variables se referencian simplemente con una palabra, definida por el programador al momento de su declaración.
- Para usar una variable, primero tenemos que declararla. Al hacerlo, obligatoriamente tenemos que indicar el tipo y, opcionalmente, inicializarla. Por ejemplo:

Variables

- En C++, las variables se referencian simplemente con una palabra, definida por el programador al momento de su declaración.
- Para usar una variable, primero tenemos que declararla. Al hacerlo, obligatoriamente tenemos que indicar el tipo y, opcionalmente, inicializarla. Por ejemplo:

Ejemplo

```
int unEntero;
unsigned int unEnteroNoNegativo = 3;
char unaLetra = 'a';
```

Variables

- En C++, las variables se referencian simplemente con una palabra, definida por el programador al momento de su declaración.
- Para usar una variable, primero tenemos que declararla. Al hacerlo, obligatoriamente tenemos que indicar el tipo y, opcionalmente, inicializarla. Por ejemplo:

Ejemplo

```
int unEntero;
unsigned int unEnteroNoNegativo = 3;
char unaLetra = 'a';
```

Cuidado! Tipado en C++

C++,permite asignar valores a variables de otros tipo, realizando conversiones implicitas de manera silenciosa.

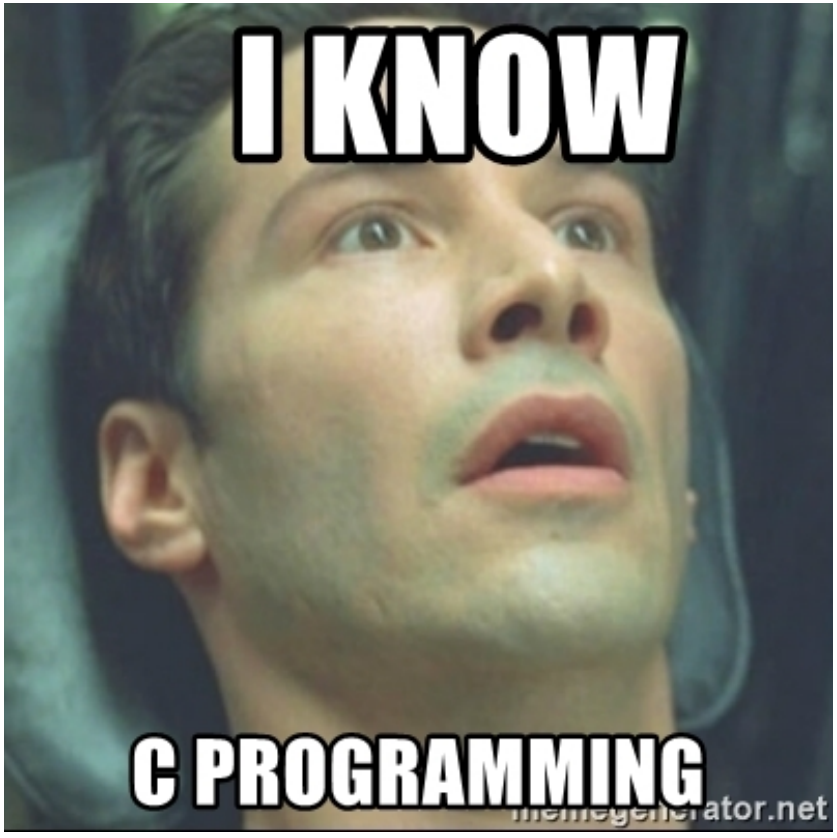
Control de flujo

Control de flujo

Las estructuras de condicionales (if) y ciclos(for,while) son muy similares a las de java

Ejemplo

```
int n = 100;
int res = 0;
for (unsigned int i = 0; i < n; i++) {
    if (i % 2 == 0) {
        res = res + i;
    }
}
```


I KNOW
C PROGRAMMING

Funciones

Definición de funciones

La definición de una función tiene la siguiente forma:

Esquema general

```
tipoRetorno nombreFuncion(tipoP1 nomP1, ...) {
   [cuerpo de la funcion]
   return valor;
}
```

Definición de funciones

La definición de una función tiene la siguiente forma:

Esquema general

```
tipoRetorno nombreFuncion(tipoP1 nomP1, ...) {
   [cuerpo de la funcion]
   return valor;
}
```

Importante

- Podemos declarar la función antes de definir su implementación.

Definición de funciones

La definición de una función tiene la siguiente forma:

Esquema general

```
tipoRetorno nombreFuncion(tipoP1 nomP1, ...) {
   [cuerpo de la funcion]
   return valor;
}
```

Importante

- Podemos declarar la función antes de definir su implementación.
- Es importante que las funciones **estén definidas antes de utilizarlas!**

Definición de funciones

La definición de una función tiene la siguiente forma:

Esquema general

```
tipoRetorno nombreFuncion(tipoP1 nomP1, ...) {
   [cuerpo de la funcion]
   return valor;
}
```

Importante

- Podemos declarar la función antes de definir su implementación.
- Es importante que las funciones **estén definidas antes de utilizarlas!**
- Esto se debe a que el compilador recorre el archivo de arriba hacia abajo

Función Main

La función Main

- Con lo que sabemos hasta ahora, podemos definir nuestro propio conjunto de funciones y usar librerías de terceros. Pero... Como hacemos para ejecutar una función determinada al correr nuestro programa?

La función Main

- Con lo que sabemos hasta ahora, podemos definir nuestro propio conjunto de funciones y usar librerías de terceros. Pero... Como hacemos para ejecutar una función determinada al correr nuestro programa?
- Para que el compilador sepa desde donde se debe comenzar a ejecutar nuestro código, por convención debemos definir una función llamada main.

La función Main

- Con lo que sabemos hasta ahora, podemos definir nuestro propio conjunto de funciones y usar librerías de terceros. Pero... Como hacemos para ejecutar una función determinada al correr nuestro programa?
- Para que el compilador sepa desde donde se debe comenzar a ejecutar nuestro código, por convención debemos definir una función llamada main.
- Deberá haber una única función llamada main en todos los archivos .cpp y .h que conforman nuestro programa (si no el compilador lanzará un error).

La función Main

- Con lo que sabemos hasta ahora, podemos definir nuestro propio conjunto de funciones y usar librerías de terceros. Pero... Como hacemos para ejecutar una función determinada al correr nuestro programa?
- Para que el compilador sepa desde donde se debe comenzar a ejecutar nuestro código, por convención debemos definir una función llamada main.
- Deberá haber una única función llamada main en todos los archivos .cpp y .h que conforman nuestro programa (si no el compilador lanzará un error).

Entrada/Salida: Salida por pantalla

Un ejemplo de esto es el operador « (incluido en la librera iostream de la STL) que lo podremos usar para visualizar valores en pantalla.

Entrada/Salida: Salida por pantalla

Un ejemplo de esto es el operador « (incluido en la librera iostream de la STL) que lo podremos usar para visualizar valores en pantalla.

Ejemplos

```
cout << "Hola mundo! ;
cout << "Hola mundo! << endl;
cout << "Mi edad es" << edad << endl;
```

Entrada/Salida: Salida por pantalla

Un ejemplo de esto es el operador « (incluido en la librera iostream de la STL) que lo podremos usar para visualizar valores en pantalla.

Ejemplos

```
cout << "Hola mundo! ;
cout << "Hola mundo! << endl;
cout << "Mi edad es" << edad << endl;
```

cout es una variable especial incluida en la libreria ostream que representa la salida estándar (stdout). Para nosotros, ”salida por consola”

Entrada/Salida: Entrada por por teclado

Equivalentemente, el operador » (también incluido en iostream) permitirá al usuario ingresar valores por teclado.

Ejemplos

```
int a;
cin >> a;
cout << a << endl;
```

Análogamente a **cout**, **cin** es la variable especial que representa la entrada estándar (stdin). Para nosotros, la ”entrada por teclado”.

Para digerir un poco lo visto

Ejercicio entre todos

a) Realizar una función máximo que reciba dos enteros y devuelva el máximo entre ellos. Luego, utilizar dicha función para pedirle al usuario dos enteros y luego mostrar en pantalla el máximo entre ellos.
b) Idem anterior, pero la función máximo debe estar en un .cpp distinto al del main

Namespaces

Namespaces

Motivación

Si hay dos funciones ”iguales” en librerías distintas, el compilador no tiene forma de saber cual debe seleccionar.

Namespaces

Motivación

Si hay dos funciones ”iguales” en librerías distintas, el compilador no tiene forma de saber cual debe seleccionar.

```
#include <iostream>
using namespace std;

// first name space
namespace first_space {
   void func() {
      cout << "Inside first_space" << endl;
   }
}

// second name space
namespace second_space {
   void func() {
      cout << "Inside second_space" << endl;
   }
}

int main () {
   // Calls function from first name space.
   first_space::func();

   // Calls function from second name space.
   second_space::func();

   return 0;
}
```

Figure 1: www.tutorialspoint.com

Namespaces

Motivación

Si hay dos funciones ”iguales” en librerías distintas, el compilador no tiene forma de saber cual debe seleccionar.

```
#include <iostream>
using namespace std;

// first name space
namespace first_space {
   void func() {
      cout << "Inside first_space" << endl;
   }
}

// second name space
namespace second_space {
   void func() {
      cout << "Inside second_space" << endl;
   }
}

int main () {
   // Calls function from first name space.
   first_space::func();

   // Calls function from second name space.
   second_space::func();

   return 0;
}
```

Figure 1: www.tutorialspoint.com

Recomendaciones

Algunas recomendaciones

- Si nuestros programas crecen en líneas de código, incluir todo en un mismo archivo de texto resulta poco practico.

Algunas recomendaciones

- Si nuestros programas crecen en líneas de código, incluir todo en un mismo archivo de texto resulta poco practico.
- Ademas, en C++ se acostumbra a separar las declaraciones de funciones de su implementación.

Algunas recomendaciones

- Si nuestros programas crecen en líneas de código, incluir todo en un mismo archivo de texto resulta poco practico.
- Ademas, en C++ se acostumbra a separar las declaraciones de funciones de su implementación.
- Las declaraciones irán en los archivos .h y las definiciones en un .cpp con el mismo nombre.

Algunas recomendaciones

- Si nuestros programas crecen en líneas de código, incluir todo en un mismo archivo de texto resulta poco practico.
- Ademas, en C++ se acostumbra a separar las declaraciones de funciones de su implementación.
- Las declaraciones irán en los archivos .h y las definiciones en un .cpp con el mismo nombre.
- Podemos entonces separar nuestras funciones en múltiples archivos de texto (.cpp y .h) que conformaran una librería.

Algunas recomendaciones

- Si nuestros programas crecen en líneas de código, incluir todo en un mismo archivo de texto resulta poco practico.
- Ademas, en C++ se acostumbra a separar las declaraciones de funciones de su implementación.
- Las declaraciones irán en los archivos .h y las definiciones en un .cpp con el mismo nombre.
- Podemos entonces separar nuestras funciones en múltiples archivos de texto (.cpp y .h) que conformaran una librería.
- Esto permitirá, ademas, utilizar la misma función en mas de un programa.

Algunas recomendaciones

- Si nuestros programas crecen en líneas de código, incluir todo en un mismo archivo de texto resulta poco practico.
- Ademas, en C++ se acostumbra a separar las declaraciones de funciones de su implementación.
- Las declaraciones irán en los archivos .h y las definiciones en un .cpp con el mismo nombre.
- Podemos entonces separar nuestras funciones en múltiples archivos de texto (.cpp y .h) que conformaran una librería.
- Esto permitirá, ademas, utilizar la misma función en mas de un programa.

Dudas?

¡A trabajar!

Palabras finales

- Con lo que vimos hasta ahora, sumado a sus conocimientos previos, deberían estar en condiciones de realizar el primer taller de la materia!
- Los ejercicios están indicados dentro de los archivos correspondientes del taller
- No duden en consultar :)