Estructuras de Datos

Introducción al Lenguaje C++

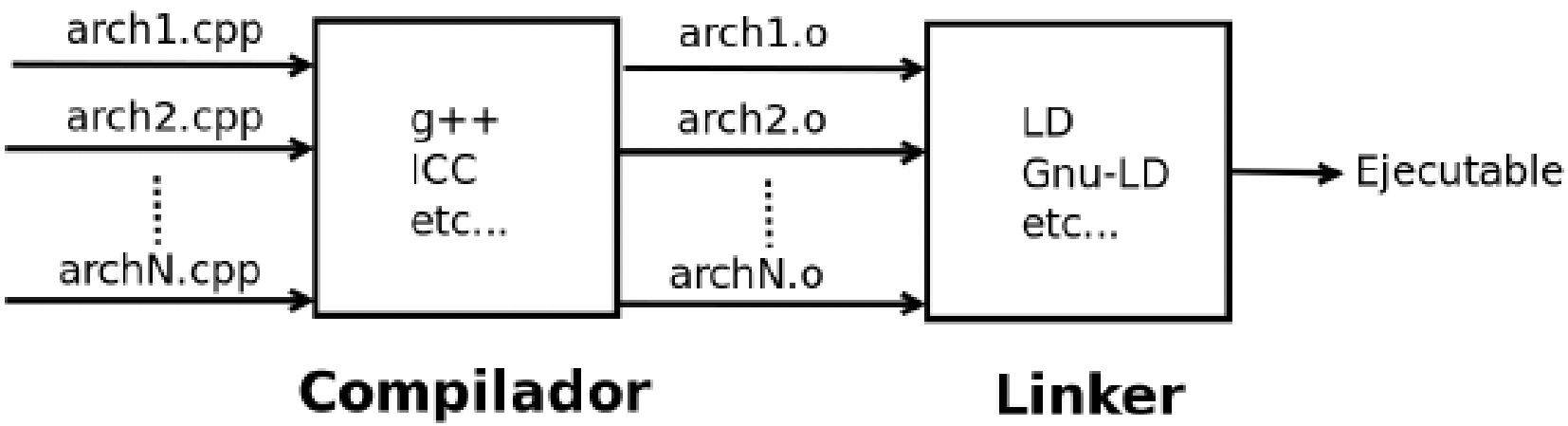
Prof. Luis Garreta

Ingeniería de Sistemas y Computación
Pontificia Universidad Javeriana – Cali

8 de agosto de 2017

C/C++ Es un lenguaje Compilado

- En C++ tenemos una serie de programas que toman nuestro código fuente y lo transforman en un archivo ejecutable.



Historia Lenguaje C++

- Fue creado en la década del 80 por Bjarne Stroustrup
- La idea principal fue extender el lenguaje C para poder trabajar con objetos.
- Lenguaje amado y odiado por muchos.
- Lo que vamos a usar en la materia es solo una pequeña porción de todo lo que ofrece.

Tipos Básicos (I)

- char.
- bool.
- int (Int, short int, long int, long long int).
- float (float, double, long double).

También se puede usar el término **void** ("vacio") para indicar que una función no devuelve nada y **const** para indicar que un parámetro no debe modificarse.

Tipos Básicos (II)

- Cuando declaremos una variable obligatoriamente tenemos que indicar el tipo, y opcionalmente inicializarla, por Ej.:
  - int a;
  - int a=3;
  - char a='a';
  - bool verdadero = true;
- Cuando trabajamos con booleanos también tenemos los operadores que ya conocemos de python:
  - !=, ==, >=, <=
- Los operadores AND y OR, se escriben:
  - &&, ||

Arreglos

- Los arreglos son similares a las listas, pero permiten acceder directamente a cada uno de sus elementos sin tener que pasar por todos los anteriores.
- Tienen longitud fija, la misma debe indicarse en el momento en que se declaran.
- Para acceder a una posición donde queremos guardar o leer un dato ponemos el subíndice entre corchetes.

Veamos ejemplos:

Ejemplos de Arreglos

```
char b[100];      // Declaro un arreglo de char de
                  // nombre b y de 100 posiciones.
b[13]   = 'a';    // En el lugar 13 guardo la a.
b[1000] = 'c';    // Ojo!! nadie chequea que
                  // me pase con los subindices!! y
                  // de paso recordemos que se empieza
                  // a numerar desde 0, o sea que el rango
                  // va desde b[0] a b[99].

int a[] = {4,8,15,16,23,42}; // Otra forma, declaramos
                             // e inicializamos.
int num = a[0]   // En num tenemos al 4
```

Funciones

- Cuando definimos una función tenemos que indicar la aridad:

```
int sumar(int a, int b){...}
bool espar(const int &a){...}
void incrementar(int &a){...}
```

- Cada línea de código debe terminar con un punto y coma.
- Toda función (que no devuelva **void**) debe terminar con un **return** que indica cual es el valor de salida.

```
int sumar(int a, int b){
  return a+b;
}

int sumar1(int a, int b){
  int res;
  res = a+b;
  return res;
}
```

Función *main*

- Todo programa en C++ tiene que tener una función llamada **main**.
- Es una función como cualquier otra pero indica el *"Entry Point"* del programa, es decir, desde donde tiene que empezar a ejecutar.

```
int main(int argc, char** argv) {
  ...
}
int main(int argc, char* argv[]){
  ...
}
int main(){
  ...
}
```

Preguntas

- Porqué devuelve un **int** ?
- Qué significan esos parámetros de entrada **argc** y **argv** ?

Librerías

- Muchas veces vamos a necesitar incluir librerías en nuestro programa.
- Una librería es un archivo donde hay definidas funciones que podemos usar,
- Asi nos ahorramos tener que estar escribiéndolas de nuevo cada vez.
- Con la instalación del compilador ya vienen varias.
- Para incluir una librería del sistema usamos la directiva

  #include <...>,

- Y si queremos incluir una librería o archivo ṕropiop, que está en el mismo directorio del proyecto principal usamos

  #include "..."

Ejemplo de uso de librerías

- Por ejemplo para mostrar algún un mensaje por pantalla necesitamos el operador << que está en la libreria **iostream**:

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

int main(int argc, char* argv[]) {
  cout << "Hola mundo!" << endl;
  return 0;
}
```

Preguntas

- Qué es *namespace?*
- Qué es *cout* y *endl?*
- Qué más tiene *iostream?*

Parámetros de las Funciones

- Podemos llamar funciones desde otras funciones.
- En C++ tenemos dos formas de pasar parámetros a las funciones, por **referencia** o por **copia**.
- **Por copia** significa que a la función se le pasa otra variable nueva con el valor de la original.
- **Por referencia** significa que se le pasa una referencia (valga la redundancia) a la variable, si la función le cambia el valor se lo está cambiando a la variable original, puede ser un comportamiento deseado o no por lo que hay que tener cuidado.
- En C++ se indica con un & delante del nombre de la variable.

Veamos un ejemplo:

Ejemplo Parámetros en las Funciones

```
void decrementar(int &a){
  a = a-1;
}

void incrementar(int a){
  a++;
}

int main(int argc, char* argv[]){
  int a = 10;
  incrementar(a);
  decrementar(a);
  incrementar(a);
  cout << a << endl;
  return 0;
}
```

Preguntas

- ¿Cuál es el valor que se muestra por pantalla?
- ¿Donde "vive" cada variable?

Cuando usar paso por referencia?

- Cuando necesite que las modificaciones de la variable se reflejen al final
  - Ejemplo: función swap
- Cuando necesite pasar a una función estructuras muy largas:
  - Ejemplo: pasar arreglos

Parámetros por referencia Ejemplo Función swap.c

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

// Intercambia los valores de a y b
void swap1 (int a, int b);
void swap2 (int &a, int &b);

int main () {
  int x = 10, y = 20;
  swap1 (x,y);
  cout << "Valor x: " << x << " Valor y: " << y << endl;

  swap2 (x,y);
  cout << "Valor x: " << x << " Valor y: " << y << endl;

  return 0;
}

void swap1 (int a, int b) {
  int tmp = a;
  a = b;
  b = tmp;
}
void swap2 (int &a, int &b) {
  int tmp = a;
  a = b;
  b = tmp;
}
```

Parámetros por referencia Ejemplo Función arreglos.c

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

// Muliplica por -1 cada elemento del arreglo
void negarArreglo (int arreglo [], int n);

int main () {
  int arregloEjemplo[] = {1,2,3,4,5};
  int n = 5;

  negarArreglo (arregloEjemplo, n);
  for (int i=0; i < n; i++)
    cout << arregloEjemplo [i] << ", ";

  cout << endl;

  return 0;
}

void negarArreglo (int *arreglo, int n) {
  for (int i=0; i < n; i++)
    arreglo [i] *= -1;
}
```

¿Qué pasa si quiero usar **negarArreglo** desde otro lado?

Separación en Archivos Diferentes

- La solución más elegante es separar en archivos diferentes:
  - Uno con la declaración de las funciones (.h).
  - Otro con la implementación de las funciones (.cpp).
  - Otro con la función princial (main) (.cpp)
- Luego incluimos el .h en todos los archivos que usen las funciones declararadas ahí.
- Para eso usamos la directiva:

  #include"archivo.h".

- Usando el ejemplo anterior nos quedarían 3 archivos:
  - funciones.h
  - funciones.cpp.
  - main.cpp,