Nociones básicas sobre C++

Agustín J. González
Patricio Olivares
ELO-329

C++ vs C

- C++ es un lenguaje más actual que C
- C++ mantiene todo el poder de C agregando características que facilitan la programación.
- C++ permite la programación orientada a objetos y programación genérica. A diferencia de Java, C++ permite este tipo de programación de forma opcional: Se pueden crear programas en el paradigma procedural u orientado a objetos según se necesite. ¡Incluso mezclas entre ellos!
- C++ posee una biblioteca de funciones mucho más amplia C. Las bibliotecas ya existentes de C son en su mayoría compatibles con C++

C++ vs Java

- C++ es plataforma dependiente (Librerías Qt buscan resolver este problema).
- C++ soporta herencia múltiple.
- C++ soporta sobrecarga de operadores (Ej: es posible darle nuevas funcionalidades al operador “+”).
- Mientras Java utiliza punteros, C++ soporta el uso de estos y operaciones sobre los mismos de forma explícita.
- C++ soporta tanto llamadas por valor como por referencia, mientras Java solo soporta llamadas por valor
- No existe un equivalente nativo para documentación en C++ como lo hay en Java (Javadoc)
- C++ crea nuevos árboles de herencia en cada programa. Java tiene un solo árbol de herencia, pues todo hereda de la clase Object.

Archivos de encabezado

- Son necesarios para declarar prototipos y definir constantes usadas en el programa.
- Son incluidos con la directiva del pre-procesador
  #include
- Ejemplo:
  #include <vector>
  #include <sys/socket.h>
  #include “setup.h”
- Al usar <....> la búsqueda del archivo se hace en lugares “estándares” definidos por el compilador.
- Los directorios estándares varían en cada instalación. Para ver cuáles son en tu sistema, puedes usar:
  - $ touch a.c
  - $ gcc -v -E a.c
  - Mostrará los directorios a buscar para archivos
    - #include <...> e
    - #include “...”

Comentarios (igual que Java)

- // Para comentarios de una línea
- /* ....
  ..... Para comentarios de múltiples líneas
  */
- No se permiten los comentarios anidados. Los comentarios son extraídos por el preprocesador, el cual no tiene capacidad de reconocer estas estructuras anidadas.
- #if 0
  código comentado
  #endif
- Hay mucho más que aprender sobre el preprocesador, ver: http://profesores.elo.utfsm.cl/~agv/elo329/miscellaneous/preprocessor.pdf

Tipos de Variable

- int
- short in (o short)
- long int (o long)
- unsigned int (o unsigned)
- unsigned long int (o unsigned long)
- unsigned short int (unsigned short)
- char
- float
- double
- long double
- bool

Acceso de Variable

- Las variables en C++ como en C, representan a los valores en sí y no referencias a éstos. En Java esto es así sólo para los tipos simples escalares como int, float, double, char y boolean.
- La diferencia entre C++ y Java se hace notar al manejar objetos.
- Objetos en Java son referencias a éstos y todos se encuentran en el heap. Mientras que en C++ los nombres de objetos siempre se refieren al objeto mismo.
- Ej: en C++
  Empleado juan, pedro; // al momento de crear la variable ya se crea el objeto invocando el constructor.
  juan=pedro; // hace que juan tome todos los atributos de pedro. Un cambio posterior a juan no afecta a pedro.
- Ésta es una gran diferencia con semántica en Java.

Salida de Datos

- #include <iostream>
- using namespace std; // para usar el objeto cout

```
int main (void){
   cout << “Hello, world” << endl;
   return 0;
}
```

- iostream debe ser incluido para hacer uso de las operaciones de entrada y salida.
- Es posible enviar datos a la salida estándar o a archivos:

```
#include <fstream>
ofstream os (“output.dat”);
os << “The value of pi is approx. “ << 3.14159 << endl;
....
```

Entrada de Datos

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

int main() {
   int i;
   ifstream fin;
   fin.open("test"); // test contains 3 ints
   for (int j=0;j<3;j++) {
      fin  >> i;
      cout << i  << endl;
   }
   fin.close();
}
```

Por documentación C++ ver cplusplus.com

Lectura desde archivo

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

int main()  {
  string s;
  ifstream fin;
  fin.open("/etc/passwd");
  while(getline(fin,s))
    cout << s << endl;
}
```

Operadores aritméticos

Asociatividad	Precedencia en orden decreciente hacia abajo
→	() [] -> .
→	! ~ ++ -- + (unario) - (unario) *(referencia) & (dirección) (tipo) sizeof
→	/ %
→	+ -
→	<< >>
→	< <= > >=
→	== !=
→	&
→	^
→	|
→	&&
→	||
←	? :
←	= += -= *= /= %= &= ^= |= >>= <<=
→	,

En principio podríamos usar and en lugar de && y or en lugar de ||; sin embargo, éstos no están soportados en todos los compiladores.

Asignaciones, Arreglos y Vectores

- Todas asignación tiene un valor, aquel asignado. Ej: a=b=c;
- ANSI C++ usa el mismo constructor de arreglo que C
- Como los arreglos de C no son particularmente poderosos, C++ incorpora vectores (no corresponde al concepto de vector geométrico). Éstos son análogos a los ArrayList de Java.
- Los vectores son una forma de plantilla (template). Su creación la veremos más adelante, pero su uso es muy simple:
  vector <X> a(n); // Ojo no usamos new como en Java...
  crea un arreglo “crecedor” de elementos de tipo X con espacio para n elementos.
- El acceso es con: a[i]

Vectores

- Pueden crecer según nuestra necesidad
  vector <double> a;
- En este caso **a** está vacío. Para hacerlo crecer:
  a.push_back(0.3);
  a.push_back(56.2);
- También podemos hacer que el vector crezca en varios elementos:
  a.resize(10);
- Podemos preguntar por el tamaño de un vector con a.size(); como en:
  for (int i=0; i < a.size(); i++)
  // ..... por más detalles ver www.cplusplus.com

Strings

- En ANSI C++ tenemos acceso a una clase poderosa para string.
- Ésta tiene definido el operador copia =, el operador concatenación + y operadores relacionales ==, !=, <, <=, >, >=, entre otros.
- El operador [ ] provee acceso a elementos individuales.
- Existen muchos métodos en esta clase como substr para extraer un substring:

```
string s = “Hola a todos”;
int n = s.length();          // asigna 12
char ch = s[0];
string t = s.substr(0,4);        // Substring de s[0] a s[4]
```

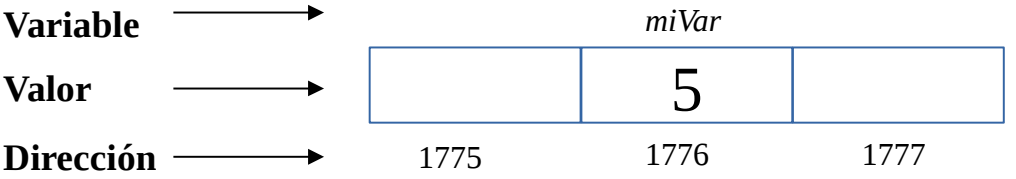
- Ver http://www.cplusplus.com/

Control de Flujo

- Se dispone de las opciones comunes en C.
- if (condición)
  block1 // Un bloque se delimita con { }
  else
  block2
- La parte else es opcional.
- While (condición) block
- do
  block
  while (condición);
- for(expresión; expresión2; expresión3)
  instrucción_a_repetir
- switch : análoga a C.

Punteros (resumen)

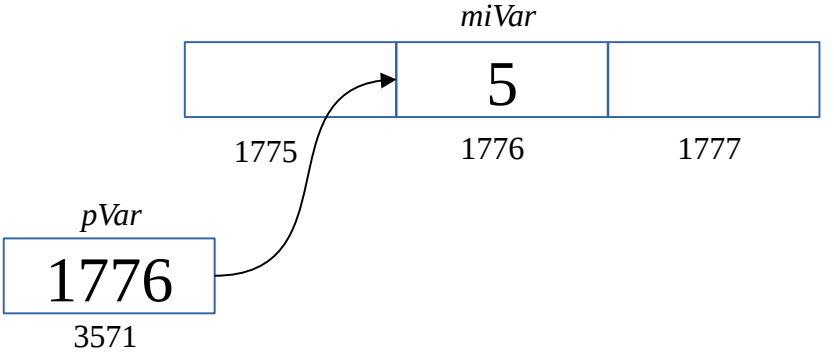
- Toda variable tiene un cierto valor o estado y una dirección de memoria asociada.
  - int miVar = 5;



- Las variables punteros, tienen almacenada la dirección en memoria de otra variable o dato.

Punteros (resumen)

- Definición en C/C++: Para definir una variable como puntero, se utiliza el operador “*”
  - int *pVar;
- A una variable de tipo puntero, se le pasa la dirección de memoria de otra variable. Para obtener la dirección de una variable en C/C++ se utiliza el operador “&”
  - pVar = &miVar;



Punteros (resumen)

- Es posible obtener el valor de la variable a la cual se apunta utilizando el operador “*”:
  - cout << &pVar <<endl; // Imprime la dirección de pVar (3571)
  - cout << pVar << endl; // Imprime el contenido de pVar (1776)
  - cout<< *pVar <<endl; // Imprime el valor del contenido apuntado (5)
- Operadores:
  - Operador de dereferencia (*): Permite acceder al valor de aquella variable a la que se apunta
  - Operador de referencia (&): Permite obtener la dirección de memoria de la variable solicitada

Paso por referencia

- En C++ tenemos un nuevo tipo de paso de argumentos, el paso por referencia.
- El efecto es equivalente al uso de punteros en C. La sintaxis cambia.
- En C se puede hacer:

```
void swap_en_C(int * px, int * py){
    int tmp = *px;
    *px = *py;
    *py=tmp;
}
```

- El llamado es swap_en_C(&a, &b)

- Ahora en C++, tenemos la opción anterior y una más simple:

```
void swap_en_Cplusplus (int & x, int & y){
    int tmp = x;
    x=y;
    y=tmp;
}
```

- El llamado es swap_en_Cplusplus(a,b);
- Obviamente:

```
swap_enCplusplus(4,x);        // no es legal
swap_enCplusplus(i+2,j-1);    // no es legal
```