

Algunas ideas básicas de C++

Agustín J. González
ELO-329

Archivos de encabezado

- Son necesarios para hacer uso de constantes predefinidas.
- Son incluidos con la directiva de l procesador
`#include`
- Ejemplo:
`#include <vector>`
`#include <sys/socket.h>`
`#include "setup.h"`
- <....> la búsqueda se hace en lugares “estándares”
- En Visual C++ \ MSDEV \ INCLUDE
- En Linux, Mirar man gcc.
 - `/usr/include` standard directory for `#include` files
 - `LIBDIR/include` standard gcc directory for `#include` files
 - `LIBDIR/g++-include` additional g++ directory for `#include`
 - `LIBDIR` es usualmente `/usr/local/lib/machine/version`. Buscar dónde la maneja Aragorn.

Comentarios

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

Tipos de Variable

- int
- short int (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, y char.
- La diferencia se produce en el manejo de objetos.
- Objetos en Java son referencias a éstos y todos se encuentran en el heap. Mientras que en C++ los nombres de los objetos siempre se refieren al objeto.
- 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.
- Gran diferencia con semántica en Java.

Salida de Datos

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

```
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 y 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();
}
```

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

-> () [] -> .
-> ! ~ ++ -- + (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).
- 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 `a[i]`

Vectores

- Pueden crecer según nuestra necesidad
vector <double> a; //variable automática (en stack)
- 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++)
// ¿Documentación? 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(); // n es 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.

Paso por referencia

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

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

- El llamado es `swap_en_C(&a, &b)`
- Ahora en C++, además de lo anterior, podemos hacerlo 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);`