Notas de programación en c++

Sergio Romero

27/12/2025

1. Introducción

Cuaderno de notas de C++ haciendo el curso: `https://ocw.mit.edu/courses/6-096-introduction-to-c-january-iap-2011/download/` lo iba a escribir en inglés pero siento que de momento se escapa a mis habilidades de inglés.

2. Lecture 1 Notes: Introduction

`https://ocw.mit.edu/courses/6-096-introduction-to-c-january-iap-2011/resources/lecture-1-notes-introduction/`

Los números *enteros* van de desde -2.147.483.648 hasta 2.147.483.647, los números *doubles* van hasta los 15 dígitos.

- Actualmente hay tres tipos de enteros: *short*, *int* y *long* a menos de que estés haciendo cosas muy especificas no te debes de preocupar por esto.
- Los tamaños y rangos de cada tipo no están completamente estandarizados; por lo general siempre se usan los estándares en la computadoras de 32 bits.

Las operaciones sólo se pueden hacer entre los mismos tipos, es decir, si haces una división entre enteros, el resultado será entero ignorando los decimales, ejemplo $\frac{9}{2} = 4$, los decimales son truncados, por que es una operación entre enteros.

La mejor praxis para declarar variables es:

```
1 int x = 4 + 2;
```

3. Lecture 2 Notes: Flow of Control

https://ocw.mit.edu/courses/6-096-introduction-to-c-january-iap-2011/resources/lecture-2-notes-flow-of-control/

Cuando un programa se ejecuta normalmente éste va del inicio al final. Es decir el primer comando se ejecuta, luego el segundo, luego el tercero y así hasta llegar al final.Sin embargo es muy útil tener control sobre los comandos de ejecución, ya sea por conveniencia o por circunstancia. Por ejemplo un juego de computadora debería de poder mover al personaje del jugador por todo el mapa cuando el jugador quiera. Necesitamos tener control sobre el orden en que se ejecutan los comandos del programa, debemos tener control sobre el flow.

Por lo general hay dos tipos de condicionamientos, llamados estructuras de control, condicionales y ciclos. Los condicionales permiten al programa revisar el valor de variables y determinar si o no ejecutar ciertos comandos, en C++ hay *if* y *switch*. Los condicionales tienen dos tipos de operadores especiales: relacionales y lógicos. Sirven para determinar si un condición es verdadera o falsa. Son operaciones tipo $<$, $<=$, $>$, $>=$, $==$, $!=$, funcionan como su contraparte aritmética sin embargo su resultado es un valor Booleano, es decir verdadero o falso. Ejemplo $4 < 5$, verdadero, $4 == 4.0$, falso. Además hay operadores lógicos como &&, ||, !, que representan, *y*, *o*, *not*, respectivamente.

El condicional*if* tiene la forma:

```
if(condition)
{
   statement1
   statement2
}
```

La condición es una expresión que va a hacer comprobada. Si la condición es verdadera, entonces los comandos posteriores serán ejecutados. si la condición es falsa, los comandos serán ignorados.

El condicional *if-else* es muy parecido al *if*, solamente en caso de que sea falso la condición, los comandos del *else* serán ejecutados. Nota que este condicional alguno de los dos condiciones se va a cumplir, por lo que los comandos ya sean del *if* o del *else* serán ejecutados.

```
if(condition)
{
   statementA1
   statementA2
}
else
{
   statementB1
   statementB2
}
```

El condicional *else if* es usado para decidir sobre dos o más bloques basado en múltiples condiciones.

```
if(condition1)
{
   statementA1
   statementA2
}
else if(condition2)
{
   statementB1
   statementB2
}
```

Si la condición 1 se cumple, el bloque correspondiente al *if* será ejecutado. Si no, sólo si la condición 2 se cumple los comandos de ese bloque serán ejecutados. Una vez un bloque se haya ejecutado, el resto será ignorado, así en este condicional podría no ejecutarse ningún bloque. Un *else* puede ser agregado al final de este condicional, así de esta forma un bloque será ejecutado forzosamente.

El condicional *switch-case* tiene la estructura de que puede o no ejecutarse cierto conjunto de comandos. Sin embargo el condicional tiene una sintaxis peculiar.

```
switch(expression)
{
   case constant1:
      statementA1
      statementA2
      ...
      break;
   case constant2:
      statementB1
      statementB2
```

```
        ...
        break;
    ...
    default:
        statementZ1
        statementZ2
        ...
}
```

El *switch-case* evalúa la expresión y si la expresión es igual a la *constant1* se ejecutarán los correspondientes comandos hasta encontrar el *break*, sino, procede a revisar la *constant2* de ser cierta, ejecutará los correspondientes comandos hasta encontrar el *break*, sino, procederá a revisar la *constant3* y así sucesivamente, en caso de que ninguna de las *constants* haya sido verdadera se ejecutará el *default*. Debido a la sintaxis del *switch-case* los corchetes no son necesarios en casa *case*, el *switch-case* tiene su equivalente en *if-else* pero el primero es mucho más ***elegante*** a la hora de expresar el mismo comportamiento.

3.1. Ciclos

Los condicionales ejecutan cierto tipo de comandos si las condiciones son cumplidas, los ciclos ejecutan cierto de comando ***mientras*** las condiciones se sigan cumpliendo. *C++* tiene tres tipo de ciclos: *while, do-while* y *for*.

El ciclo *while* tiene la forma del condicional *if*:

```
while(condition)
{
    statement1
    statement2
    ...
}
```

Mientras la *condition* se siga cumpliendo el segmento de bloque de los comando se seguirá ejecutando repetidamente.

El ciclo *do-while* es una variación del *while* que garantiza que al menos una vez el segmento de bloque de los comandos se ejecutará.

```
do
{
    statement1
    statement2
    ...
}
```

```
7 while(condition);
```

Primero se ejecuta el segmento de bloque de los comandos, si la condición se mantiene, entonces repite el segmento de bloque, sino, termina el ciclo de repetición.

3.2. For

El ciclo *for* funciona como el ciclo *while* pero con un cambio en la sintaxis:

```
1 for (initialization; condition; incrementation)
2 {
3    statement1
4    statement2
5 }
```

El ciclo *for* es diseñado para permitir a una variable contadora que se inicializa en el principio del ciclo y que se va incrementando o disminuyendo en cada interacción del ciclo. Nota que el ciclo *while* y el ciclo *for* son equivalentes.

```
1 initialization
2 while(condition)
3 {
4    statement1
5    statement2
6    ...
7    incrementation
8 }
```

Siendo su equivalente en ciclo *for*:

```
1 for (initialization; condition; incrementation)
2 {
3    statement1
4    statement2
5 }
```

El incremento puede ir en cualquier parte del bloque de segmento correspondiente, sin embargo es buena praxis dejarlo al final del bloque.

4. Nested Control Structures

Es posible colocar condicionales dentro de otros condicionales y ciclos dentro de otros ciclos, además es posible hacer cualquier combinación que se te ocurra de los temas de este resumen.

Ejemplo:

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

int main(){
for(int x=0; x<4; x=x+1){
   for(int y=0; y<4; y=y+1)
      cout << y;
   cout << "\n";
   }
}
```

5. Ejercicios

Lista de ejercicios de: `https://ocw.mit.edu/courses/6-096-introduction-to-c-january-resources/problem-set-1/`

5.1. Hola mundo

Escribe un programa que su salida sea Hello, world! usando una constante **const char *** con el valor de Hello, world!.