Laboratorio de Programación - Labo01

Parte 2: Entrada Salida y Variables

Algoritmos y Estructuras de Datos I

Departamento de Computación, FCEyN, Universidad de Buenos Aires.

Lenguajes de Programación

Un lenguaje de programación es un lenguaje formal que proporciona una serie de instrucciones que permiten a un programador escribir secuencias de órdenes y algoritmos a modo de controlar el comportamiento físico y lógico de una computadora con el objetivo de que produzca diversas clases de datos.

Programar en C++

```cpp
#include <iostream>

int main() {
  // Comentario: definicion de las variable
  int a = 0;
  int b = 0;
  std::cout << "Ingrese un valor de a " << std::endl;
  std::cin >> a;
  std::cout << "Ingrese un valor de b " << std::endl;
  std::cin >> b;
  int r = 3*b - a;
  std::cout << "El valor de r es " << r << std::endl;
  return 0;
}
```

- El C++ entra en la categoría de **lenguajes imperativos**.
- Son secuenciales: los comandos se ejecutan linea a linea, de arriba hacia abajo.
- La ejecución de un programa imperativo tiene temporalidad.

Programación imperativa

- **Entidad fundamental:** variables, que corresponden a posiciones de memoria (RAM) que almacenan valores durante la ejecución de un programa.
- **Operación fundamental:** asignación, cambia el valor de una variable.

$$variable = valor$$

1. Una variable no cambia a menos que se cambie explícitamente su valor, a través de una asignación.
2. En los lenguajes tipados (*typed*), las variables tienen un tipo de datos y almacenan valores del conjunto base de su tipo.

- **Entrada y Salida desde Consola:** la consola (teclado y monitor) permite interactuar con el usuario.
  - `cout`: console out. Imprime por pantalla un dato (usamos $<<$)
  - `cin`: console in: Lee un dato ingresado desde el teclado (usamos $>>$ y hay que apretar ENTER)
  - `cout` y `cin` están definidas en la librería `iostream`.

Programación imperativa: Variables

- Para almacenar valores utilizamos variables, que se declaran con un tipo de datos asociado:

```
1  #include <iostream>
2
3  int main() {
4    int a = 11;
5    std::cout << a;
6    return 0;
7  }
```

- A partir de la línea 5, la variable **a** contiene el entero 11.
- En el siguiente comando, se accede a esta variable y se imprime por consola su valor usando `cout`.

Demo #1: Entrada y Salida desde Consola

Una variable también puede recibir la asignación de un valor de su mismo tipo desde la consola, usando la instrucción `cin`.

```
1  #include <iostream>
2
3  int main() {
4    // Primero declaro una variable que recibe el dato
5    int valor = 0;
6    std::cout << "Ingrese un valor entero" << std::endl;
7    std::cin >> valor;
8    std::cout << "El valor ingresado fue " << valor << std::endl;
9    return 0;
10 }
```

Tipos de datos de C++

- Una variable está asociada a un **tipo de datos**:
  1. es un conjunto de valores (llamado el *conjunto base* del tipo),
  2. tiene definidas una serie de operaciones para trabajar con los elementos de ese conjunto.
- En C++ tenemos tipos de datos que implementan (en algunos casos parcialmente) cada uno de los tipos de datos del lenguaje de especificación:
  - El tipo `int` para números enteros ($\mathbb{Z}$)
  - El tipo `float` para números reales ($\mathbb{R}$)
  - El tipo `bool` para valores booleanos (Bool)
  - El tipo `char` para caracteres (Char)
- Atención: Ni `int` ni `float` contienen todos los valores de $\mathbb{Z}$ y $\mathbb{R}$, pero a los fines de AED1, podemos asumir que $\mathbb{Z} = \texttt{int}$ y $\mathbb{R} = \texttt{float}$.

Declaración y asignación de variables

- TODAS LAS VARIABLES se deben declarar antes de su uso.
  1. **Declaración:** Especificación de la existencia de la variable, con su tipo de datos.
  2. **Inicialización:** La primera asignación a una variable. Entre la declaración y la inicialización tiene "basura".
  3. **Asignación:** Asociación de un valor a la variable, que no cambia a menos que sea explícitamente modificado por otra asignación.

```
1 int main() {
2   int a; // Declaracion, aqui a no tiene valor util
3   a = 5; // Inicializacion
4   a = a+2; // Asignacion de un nuevo valor
5   ...
6 }
```

- Una variable puede ser inicializada al declararla: `int a = 5` es válido.

Operadores aritméticos

- Asociados a los tipos de variables, se definen los siguientes operadores aritméticos:
  1. + y -: suma y resta.
  2. *, /: multiplicación y división.
  3. **%**: módulo, devolviendo el resto de la división entre dos números.

```
1  #include <iostream>
2  int main() {
3    // declaro variables
4    int a = 11,c;
5    float b = 3.14,d;
6    // realizo las operaciones
7    c = a + b;
8    d = a + b;
9    // imprimo los resultados
10   std::cout << "Impresion de operacion -> " << a + b << std::endl;
11   std::cout << "Impresion de resultado entero -> " << c << std::endl;
12   std::cout << "Impresion de resultado flotante -> " << d << std::endl;
13   return 0;
14 }
```

Expresiones en C++

- El elemento del lado derecho de una asignación es una expresión.
- Esta expresión también puede incluir llamadas a funciones:
-

```
1  #include <iostream>
2  #include <cmath> // incluye seno, coseno, tangente, etc.
3
4  int main() {
5    float x = 2 + 5;
6    float y = sin(x) + cos(x);
7
8    std::cout << y;
9    return 0;
10 }
```

Operadores de Comparación e Igualdad

- Existen operadores de comparación e igualdad que devuelven un resultado booleano:
  1. **==** y **!=**: igualdad y desigualdad.
  2. $>$, $<$, $>=$, $<=$: mayor, menor, mayor e igual, menor e igual.

- Las comparaciones pueden realizarse sobre constantes numéricas o sobre variables.

- ```
  (5 == 5); // devuelve true
  (6 <= 2); // devuelve false
  (b == d); // depende de las variables b y d
  (c > a);
  ```
  ```

Demo #2: Operadores de Comparación e Igualdad

```cpp
#include <iostream>

int main() {
    // declaro e inicializo las variables
    int a = 6;
    int b = 3;
    // imprimo resultados de comparacion
    std::cout << "(5 == 5) -> " << (5 == 5) << std::endl;
    std::cout << "(a < b) -> " << (a < b) << std::endl;

    return 0;
}
```

Operadores Lógicos

- Los operadores lógicas en C++ son:
  1. `&&`: AND lógico.
  2. `||`: OR lógico.
  3. `!` : NOT lógico.

- Veamos un ejemplo utilizando variables y operaciones lógicas.

Demo #2: Ejemplo de Operaciones Lógicas

- Reemplazar el main.cpp con el siguiente código:
-

```cpp
#include <iostream>

int main() {

    bool a = false;
    bool b = true;
    bool c;

    c = a && b;

    std::cout << "Valor c: " << c << std::endl;
    return 0;
}
```

Demo #2: Ejemplo de Operaciones Lógicas

- Al ejecutarlo, el panel de salida muestra:

```
variable c: 0

Process finished with exit code 0
```

- El valor de la variable c es 0!!!
- Y el valor booleano????
- Podemos imprimir un mensaje más adecuado?.

Demo #2: Ejemplo de Operaciones Lógicas

- Introducimos el Operador Condicional Ternario
- `expresion ? resultado1 : resultado2`
- Reemplazar el código
-
```
1 std::cout << "Valor c: " << (c ? "true" : "false") << std::endl;
```
- Tambien puede realizar otro tipo de operaciones interesantes
-
```
1 std::cout << "El maximo es: " << (a > b ? a : b) << std::endl;
```

Operadores Lógicos

- Los operadores && y || utilizan lógica de cortocircuito: No se evalúa la segunda expresión si no es necesario.
- En otras palabras && implementa el $\wedge_L$ y || implementa el $\vee_L$
-

```
1 // inversoMayor entre n y m
2 bool c = (n != 0 && 1/n > m);
3 std::cout << "El inverso de n " << (c ? "" : " no ");
4 std::cout << " es mayor que m" << std::endl;
```

- Si $n = 0$, entonces el primer término es falso, pero el segundo está indefinido! En C/C++, esta expresión evalúa directamente a falso.
- Solamente se evalúa $1/n > m$ si $n \neq 0$.

Concordancia de tipos

- En C/C++ es obligatorio asignar a cada variable una expresión que coincida con su tipo, o que el compilador sepa cómo convertir en el tipo de la variable.
- Se dice que C++ es un lenguaje débilmente tipado.

```
1 int main() {
2   int a = "Hey, hey!"; // La asignacion NO es un int!!!
3   ...
4 }
```

Demo #3: Expresiones indefinidas en C++

¿Qué pasa al ejecutar este programa?

```
1  #include <iostream>
2
3  int main() {
4    std::cout << "Ingrese un valor por teclado " << std::endl;
5    int x;
6    std::cin >> x;
7    int r = 1 / x;
8    std::cout << "Su inverso es " << r << std::endl;
9    return 0;
10 }
```

- ¿Qué valor retorna si ingresamos 10?
- ¿Qué valor retorna si ingresamos 0?

Expresiones indefinidas en C++

- Lamentablemente, C++ no define qué ocurre cuando evaluamos una operación indefinida
- Algunas cosas que pueden pasar
  - Termina la ejecución con un `exit code` distinto de 0
  - Continua la ejecución con un valor cualquiera (El HORROR, El HORROR, El HORROR)