Guía de Ejercicios 1: Programación básica en C++

[Version: 7 de febrero de 2023]

Objetivos:

- Introducir la sintaxis y las principales características del lenguaje de programación C++.
- Introducir los tipos de datos básicos, los tipos `std::string`, `std::vector`, `std::set`, `std::map`, y la definición de tipos utilizando `struct`.
- Entender los conceptos de pasaje de parámetros por copia y por referencia.

Ejercicio 1. Hola mundo en C++.

(a) Utilizando el operador << se puede imprimir valores por pantalla. Escribir un programa con el siguiente código y compilarlo. Ejecutarlo y observar su salida.

```
#include <iostream>

int main(){
    std::cout << "Hola mundo" << std::endl;
}
```

¿Qué es <iostream> y por qué es necesario incluirlo?

(b) Escribir un programa con el siguiente código y compilarlo. Ejecutarlo y observar su salida.

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

int main(){
    cout << "Hola mundo" << endl;
}
```

¿Cuál es la diferencia entre `cout` y `std::cout`? ¿Qué sucede si este programa no incluye la línea `using namespace std;`?

(c) También podemos imprimir variables y utilizar varios valores intercalando el operador <<. Escribir un programa con el siguiente código, compilarlo, ejecutarlo y observar su salida.

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

int main(){
    string nombre = "Juan";
    string edad = "30";
    cout << nombre << " y tiene " << edad << " años." << endl;
}
```

(d) Se pueden tomar valores de la entrada estándar utilizando `cin` y el operador `>>` para almacenarlos en una variable declarada previamente. Escribir un programa con el siguiente código, compilarlo, ejecutarlo y observar su salida.

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

int main(){
    string nombre;

    cout << "Ingrese su nombre: ";
    cin >> nombre;
    cout << "Su nombre es " << nombre << "." << endl;
}
```

(e) ¿Qué sucede si se ingresa un texto con espacios en el programa anterior? Ingrese por ejemplo su nombre y apellido.

(f) La función `getline` permite leer una línea completa de la entrada estándar. Escribir un programa con el siguiente código y compilarlo. Ejecutarlo ingresando un texto de varias palabras y comparar su salida con la del programa anterior.

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

int main(){
    string nombre_completo;

    cout << "Ingrese su nombre completo: ";
    getline(cin, nombre_completo);
    cout << "Su nombre es " << nombre_completo << "." << endl;
}
```

Ejercicio 2. Tipos básicos en C++.

(a) Escribir un programa que tome dos números enteros por entrada estándar y que imprima si la suma de los números ingresados es positiva, negativa o cero.

(b) ¿Cuál será la salida al ejecutar el siguiente programa? ¿Qué realiza la instrucción de de la línea 6?

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

int main(){
    int n = 5;
    bool condicion = (n == 4 || n == 5);
    if(condicion){
        cout << "verdadero" << endl;
    } else{
        cout << "falso" << endl;
```

```
    }
}
```

(c) Indique cuál será la salida al ejecutar el siguiente programa.

- ¿Por qué el comportamiento cambia para `mi_entero` y para `mi_decimal`?
- ¿Qué valor toma la expresión `(mi_entero / 2 > 4)`? ¿Y `(mi_decimal / 2 > 4)`?

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

int main(){
    int mi_entero = 9;
    float mi_decimal = 9;

    if(mi_entero / 2 > 4){
        cout << "mi_entero dividido 2 es mayor a 4" << endl;
    } else {
        cout << "mi_entero dividido 2 NO es menor a 4" << endl;
    }

    if(mi_decimal / 2 > 4){
        cout << "mi_decimal dividido 2 es mayor a 4" << endl;
    } else {
        cout << "mi_decimal dividido 2 NO es menor a 4" << endl;
    }
}
```

Ejercicio 3. Tipos `std::string` y `std::vector`.

(a) Dada la variable `materia` de tipo `string`, indique el tipo y el valor de las siguientes expresiones en C++.

```cpp
string materia = "Algoritmos y Estructuras de Datos";
```

- `materia[0]`
- `materia[5]`
- `materia.front() == 'E'`
- `materia.back() == 's'`
- `materia.size()`
- `materia.length()`
- `materia.substr(13, 11)`

(b) Dada la variable `mediciones` de tipo `vector<int>`, indique el tipo y el valor de las siguientes expresiones en C++.

```cpp
vector<int> mediciones = {19, 18, 22, 20, 23, 24};
```

- `mediciones[0]`
- `mediciones[3]`
- `mediciones.front()`
- `mediciones.back() == 23`
- `mediciones.size()`

(c) Las siguientes operaciones modifican el contenido de las variables `materia` y `mediciones`. Indique cuál es el contenido de cada variable luego de cada una de las siguientes operaciones.

- `materia.append("{} - UTDT")`
- `materia.push_back('X')`
- `mediciones.pop_back()`
- `mediciones.push_back(20)`

Ejercicio 4. Ciclos.

(a) Completar el código del siguiente programa para que imprima por pantalla todos los elementos del vector `mediciones`.

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

int main(){
    vector<int> mediciones = {19, 18, 22, 20, 23, 24};

    int i = 0;
    while( i < mediciones.size() ){
        // ...
    }
}
```

(b) Se quiere modificar el código del programa del punto anterior para que imprima los valores en orden inverso al que están almacenados en la variable. La salida esperada en ese caso es `24 23 20 22 18 19`.

- Implementar la modificación haciendo que la variable iteradora `i` comience en `mediciones.size() - 1` y recorra el vector desde el final hacia el principio.
- Implementar la modificación haciendo que la variable iteradora `i` comience en `0` y aumente hasta `mediciones.size()`.

(c) Completar el código del siguiente programa para que imprima por pantalla un mensaje en caso de que el texto ingresado contenga la letra `'z'`. En ese caso, se debe indicar también la posición de la letra en el texto ingresado. Por ejemplo, si el texto ingresado es `"adivinanza"`, se debe indicar que la letra `'z'` se encuentra en la posición `8`.

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

int main(){
    string texto;
    getline(cin, texto);

    // ...
}
```

(d) Escriba un programa que tome un texto ingresado por el usuario e imprima por pantalla el mismo texto reemplazando todas las letras vocales por la letra `'a'`. Por ejemplo, si el texto ingresado es `"Hola, ¿que tal?"`, se debe imprimir `"Hala, ¿qaa tal?"`.

Ejercicio 5. Funciones. Completar el código faltante en las siguientes funciones.

(a)
```
string reeplazar_vocales_por_a(string texto){
    // ...
}
```

(b)
```
bool suman_cero(int a, int b){
    // ...
}
```

(c)
```
int maximo(int a, int b){
    // ...
}
```

(d)
```
int cantidad_de_vocales(string texto){
    // ...
}
```

(e) ¿Cómo modificaría el código del ítem anterior si contase con la siguiente función que indica si un caracter es una letra vocal o no?

```
bool es_vocal(char caracter){ /* ... */ }
```

Ejercicio 6. Pasaje de parámetros por referencia.

(a) El símbolo & luego del tipo de un parámetro indica que este se pasará por referencia, en lugar de pasarse por copia.

- ¿Qué se imprimirá por pantalla al ejecutar el siguiente programa? Compile y ejecute el programa.
- ¿En qué se diferencian las dos funciones que están implementadas?
- ¿Qué significa que el tipo de retorno de la función sea **void**?

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

void funcion_A(int n){
    n = 5;
}

void funcion_B(int & n){
    n = 5;
}

int main(){
    int numero = 10;
    cout << "numero: " << numero << endl;
    cout << "Llamada a funcion_A(numero)." << endl;
```

```
16      funcion_A(numero);
17      cout << "numero: " << numero << endl;
18      cout << "Llamada a funcion_B(numero)." << endl;
19      funcion_B(numero);
20      cout << "numero: " << numero << endl;
21  }
```

(b) Implementar una función llamada `encerrar_entre_parentesis` que tome un `string` por parámetro y lo modifique para que su contenido quede encerrado entre paréntesis. La función no debe devolver un `string` sino modificar el que se pasa por parámetro.

(c) Implementar una función que tome un vector de enteros y un entero n y devuelva un entero. La función debe modificar cada aparición de n en el vector por un 0 y además debe devolver la cantidad de apariciones de n en el vector original. Puede tomar el siguiente encabezado para escribir la función.

```
1  int reemplazar_numero(vector<int> & lista, int n)
```

Ejercicio 7. La palabra clave `const` en el tipo de un parámetro indica que no se debe modificar su contenido.

(a) El siguiente programa tiene un defecto en el código, compilarlo e inspeccionar el mensaje de error.

- ¿Cuál es el defecto en el código?
- ¿Qué se debe modificar para resolverlo?

```
1  #include <vector>
2  using namespace std;
3
4  void modificar_primero(const vector<int> & lista){
5      lista[0] = 0;
6  }
7
8  int main(){
9      vector<int> mediciones = {19, 18, 22, 20, 23, 24};
10     modificar_primero(mediciones);
11 }
```

(b) El modificador `const` también se puede utilizar al definir una variable e impide que el contenido de la variable se modifique. Compilar el siguiente programa e inspeccionar el mensaje de error, ¿por qué no se puede realizar el llamado a función de la línea 10?

```
1  #include <vector>
2  using namespace std;
3
4  bool es_vacio(vector<int> & lista){
5      return (lista.size() == 0);
6  }
7
8  int main(){
```

```
    const vector<int> mediciones = {19, 18, 22, 20, 23, 24};
    bool res = es_vacio(mediciones);
}
```

(c) Se tiene una gran cantidad de datos almacenados en una variable de tipo `vector<string>` y se quiere aplicar un procesamiento a partir del cual se obtiene un número. El procesamiento no requiere modificar el vector de strings.

- ¿Cuál de estas opciones de encabezado para la función utilizaría y por qué?

```
int procesar_datos(vector<string> datos);

int procesar_datos(vector<string> &datos);

int procesar_datos(const vector<string> &datos);
```

Ejercicio 8. Ejercicios variados de funciones y ciclos.

(a) Completar la siguiente función:

```
int sumatoria(const vector<int> &numeros){
    // ...
}
```

(b) Escribir una función llamada `farenheit_a_celsius` que convierta una temperatura expresada en grados Fahrenheit a grados Celsius.

(c) Utilizar la función del ítem anterior para imprimir una tabla con la conversión de los valores Fahrenheit desde 30 a 100 en intervalos de 10.

(d) Escribir una función llamada `es_primo` que indique si un número es primo o no.

(e) Utilizar la función del ítem anterior para imprimir por pantalla todos los números primos que se encuentran entre el 1 y el 100.

(f) Escribir una función que, dado un string, indique si es palíndromo (si se lee igual al derecho y al revés).

(g) Escribir una función que toma un vector de strings (es decir, `vector<string>`) y devuelve un único string que resulta de concatenar todos los strings del vector.

Ejercicio 9. Tipos enumerados.

(a) Dado el estado final de las variables del siguiente programa indique el tipo y el valor de las expresiones.

```
  enum class Color {Blanco, Negro, Rojo, Verde, Azul};
  Color un_color = Color::Blanco;
  Color otro_color = Color::Rojo;
```

- `un_color`
- `un_color == otro_color`
- `un_color == Color::Blanco`
- `Color::Blanco == Color::Azul`

(b) Utilizando la definición del tipo `Color` del punto anterior, implemente una función que dado un `vector<Color>` cuente la cantidad de veces que contiene el color rojo.

(c) Escribir un programa que utilice un tipo enumerado para almacenar un día de la semana y escribir una función que tome un día de la semana e indique si es fin de semana o no.

Ejercicio 10. Tipos `std::set` y `std::map`.

(a) Dado el estado final de la variable `mi_conjunto` ...

```
set<int> mi_conjunto = {6, 8, -95, 9};
mi_conjunto.insert(6);
mi_conjunto.insert(7);
mi_conjunto.erase(9);
```

... indique el tipo y el valor de las siguientes expresiones en C++.

- `mi_conjunto.size()`
- `mi_conjunto.contains(6)`
- `mi_conjunto.contains(9)`
- `mi_conjunto.empty()`

(b) Considere la siguiente variable `info_personas` de tipo `map`.

```
map<string, int> info_personas = { {"Ada", 1800}, {"Barbara", 1939}};
```

¿Cómo se modifica `info_personas` con las siguientes instrucciones?

```
info_personas["Rebeca"] = 1926;
info_personas["Ada"] = 1815;
```

Luego de esas instrucciones, ¿cuál es el tipo y el valor de las siguientes expresiones?

- `info_personas["Ada"]`
- `info_personas.contains("Alan")`
- `info_personas.size()`
- `info_personas.empty()`

Ejercicio 11. Registros o structs. La keyworkd **`struct`** nos permite definir un nuevo tipo de datos como un agrupamiento de variables.

(a) Utilizar el siguiente **`struct`** para escribir un programa que tome por entrada los datos de un nuevo usuario, inicializando la cantidad de seguidores en cero y luego imprima todos los datos por pantalla.

```
struct usuario
{
    string nombre;
    string email;
    int cantidad_de_seguidores;
};
```

(b) Dada la siguiente **struct**, escribir una función que determine si dos personas se llevan más de 10 años de diferencia.

```
struct persona
{
    string nombre;
    int edad;
    string dni;
};
```

(c) Escribir la declaración de un tipo `materia` que almacene el nombre de la materia, el código de la materia, la cantidad de alumnos y el nombre de la carrera.

(d) Implementar la función `persona_mas_grande` que toma un vector de elementos del tipo `persona` definido anteriormente y devuelve la persona cuya edad es la más grande.

```
persona persona_mas_grande(const vector<persona> & lista_de_personas){
    // ...
}
```