Informe de Laboratorio 01

Tema: C++ estructuras, clases y objetos

Nota

Estudiante	Escuela	Asignatura
Andre Jimmy Hilacondo Begazo	Escuela Profesional de Ingenieria de Sistemas	Tecnología de Objetos Código: 1703240
Laboratorio	**Tema**	**Duración**
01		02 horas

Semestre académico	Fecha de inicio	Fecha de entrega
2024 - B		

1. Tarea

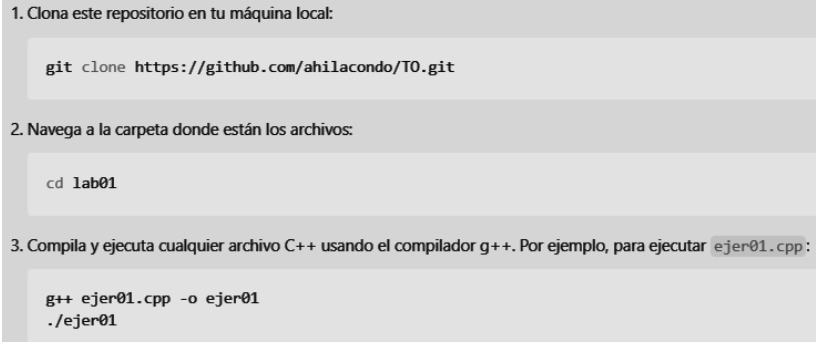
- Implementar el algoritmo en C++ que resuelva los ejercicios propuestos
- Utilizar Git para llevar el control de versiones del proyecto.
- Crear un repositorio privado en GitHub y otorgar permisos de colaborador a los profesores.

2. Equipos, materiales y temas utilizados

- Sistema Operativo: Windows 10 / Ubuntu 22.04
- Editor/IDE: Visual Studio Code
- C++: Compilador g++ o cl.exe
- Cuenta en GitHub con el correo institucional.

3. URL de Repositorio Github

- URL para el laboratorio 01 en el Repositorio GitHub https://github.com/ahilacondo/TO.git
- Los pasos a seguir se en encuentran en https://github.com/ahilacondo/TO/tree/master/lab01



4. Actividades

1. Crear un programa que indique si un número es un cubo perfecto (anstrong) o no, se dice que un número es cubo perfecto si al sumar el cubo de sus dígitos dan el mismo número.

Por ejemplo 153, cubos de sus dígitos:

$1^3 + 5^3 + 3^3 = 153$. -> es cubo perfecto

Resolución:

```
#include <iostream>
#include <cctype>
#include <limits>
#include <string>
#include <sstream>

using namespace std;

int esAnstrong(int n); //Funcion que nos dice si un numero es cubo perfecto
inline int cubo(int n); //Funcion que nos da el cubo de un numero
bool esNumValido(const string &str); //Funcion que nos dice si un string es un
numero valido
void obtenerDigitos(int n, int digitos[], int &tam); //Funcion que nos da los
digitos de un numero

inline int cubo(int n){
    return n * n * n;
}

void obtenerDigitos(int n, int digitos[], int &tam){
    tam = 0;
    while (n > 0){
        digitos[tam++] = n % 10;
        n /= 10;
    }
}

int esAnstrong(int n){
    int tam;
    int digitos[10];
    obtenerDigitos(n, digitos, tam);

    int suma = 0;
    while(tam > 0){
        suma += cubo(digitos[--tam]);
    }
    return suma == n;
}

bool esNumValido(const string &str){
    for(char c: str){
        if(!isdigit(c))
            return false;
```

```
    }
    return true;
}
int main(){
    char seguir = 's';
    string str;

    while (tolower(seguir) == 's'){
        int num = 0;
        cout << "''''''''''''''''''INGRESE UN NUMERO ''''''''''''''''''" << endl;
        getline(cin, str);

        if(esNumValido(str)){
            stringstream ss(str);
            ss >> num;

            if(esAnstrong(num))
                cout << "El numero ingresado es cubo perfecto" << endl;
            else
                cout << "El numero ingresado NO es cubo perfecto" << endl;

        }else{
            cout << "El valor ingresado no es un numero valido" << endl;
        }
        cout << "Desea ingresar otro numero? (s/n)" << endl;
        cin >> seguir;
        cin.ignore(numeric_limits<streamsize>::max(), '\n');
    }

    return 0;
}
```

Ejecución

```
PS C:\UNSA - SISTEMAS\2024-UNSA\2024 - B\TO\lab01> cd
deRunnerFile } ; if ($?) { .\tempCodeRunnerFile }
''''''''''''''''INGRESE UN NUMERO ''''''''''''''''
1234
El numero ingresado NO es cubo perfecto
Desea ingresar otro numero? (s/n)
s
''''''''''''''''INGRESE UN NUMERO ''''''''''''''''
153
El numero ingresado es cubo perfecto
Desea ingresar otro numero? (s/n)
s
''''''''''''''''INGRESE UN NUMERO ''''''''''''''''
asdf
El valor ingresado no es un numero valido
Desea ingresar otro numero? (s/n)
n
```

2. Se ha seleccionado N números de personas para realizar una encuesta, en un proceso repetitivo se ingresa el grado de instrucción (1-Primaria/2-Secundaria/3-Superior) y la edad (entre 15 y 80 años) de cada persona. Obtener la edad y el grado de instrucción usando número aleatorios.

Construya un programa que muestre lo siguiente:
a) El promedio de edades
b) La mayor edad.
c) Cantidad de Personas con instrucción Primaria
d) Cantidad de Personas con instrucción Secundaria.
e) Cantidad de Personas con instrucción Superior.

Nota: Resolver con struct, bucles, arreglos y funciones.

```
#include <iostream>
#include <string>
#include <cstdlib>
#include <ctime>

using namespace std;

const int PRIMARIA = 1;
const int SECUNDARIA = 2;
const int SUPERIOR = 3;

struct persona{
    string grado;
    int edad;
};

int generarAleatorio(int min, int max); //Funcion que genera numeros
aleatorios
void imprimirPersonas(persona personas[], int n);
void imprimirDatos(); //Esta funcion va imprimir los datos que requiere el
ejercicio
int edadMayor(persona persona[], int n);


int generarAleatorio(int min, int max){
    return rand() % (max - min + 1) + min;
}

void imprimirPersonas(persona personas[], int n){
    for(int i = 0; i < n; i++){
        cout << "Persona " << i + 1 << ": Edad = " << personas[i].edad << ",
Grado = " << personas[i].grado << endl;
    }
}

int edadMayor(persona personas[], int n){
```

```
    int mayor = personas[0].edad;
    for(int i = 1; i < n; i++){
        if(mayor < personas[i].edad)
            mayor = personas[i].edad;
    }

    return mayor;
}

void imprimirDatos(persona personas[], int n, int cantPrimaria, int
cantSecundaria, int cantSuperior){
    cout << "Cantidad de personas con grado de primaria: " << cantPrimaria
<< endl;
    cout << "Cantidad de personas con grado de secundaria: " <<
cantSecundaria << endl;
    cout << "Cantidad de personas con grado de superior: " << cantSuperior
<< endl;
    cout << "Edad de la persona mayor: " << edadMayor(personas, n) << endl;
}

int main(){
    int n = 10; //numero de personas
    int edadMin = 15;
    int edadMax = 80;
    int cantPrimaria = 0;
    int cantSecundaria = 0;
    int cantSuperior = 0;
    srand(time(0));

    persona personas[n];

    for(int i = 0; i < n ; i++){
        personas[i].edad = generarAleatorio(edadMin, edadMax);
        int g = generarAleatorio(PRIMARIA, SUPERIOR);

        switch(g){
            case PRIMARIA:
                personas[i].grado = "Primaria";
                cantPrimaria++;
                break;
            case SECUNDARIA:
                personas[i].grado = "Secundaria";
                cantSecundaria++;
                break;
            case SUPERIOR:
                personas[i].grado = "Superior";
                cantSuperior++;
                break;
        }
    }

    imprimirPersonas(personas, n);
    imprimirDatos(personas, n, cantPrimaria, cantSecundaria, cantSuperior);
    return 0;
}
```

Ejecución

```
PS C:\UNSA - SISTEMAS\2024-UNSA\2024 - B\TO\lab01>
deRunnerFile } ; if ($?) { .\tempCodeRunnerFile }
Persona 1: Edad = 30, Grado = Superior
Persona 2: Edad = 39, Grado = Superior
Persona 3: Edad = 77, Grado = Secundaria
Persona 4: Edad = 55, Grado = Secundaria
Persona 5: Edad = 80, Grado = Superior
Persona 6: Edad = 29, Grado = Primaria
Persona 7: Edad = 51, Grado = Superior
Persona 8: Edad = 79, Grado = Primaria
Persona 9: Edad = 46, Grado = Primaria
Persona 10: Edad = 56, Grado = Superior
Cantidad de personas con grado de primaria: 3
Cantidad de personas con grado de secundaria: 2
Cantidad de personas con grado de superior: 5
Edad de la persona mayor: 80
```

3. **Diseñe un programa que imprima los divisores de un número ingresado por teclado y la cantidad de divisores encontrados.**

Nota: Resolver con bucles, arreglos y funciones

```
#include <iostream>
#include <vector>

using namespace std;

vector<int> cantDivisores(int n);
void printDatos(vector<int> v);

vector<int> cantDivisores(int n){
    vector<int> divisores;

    for (int i = 1; i <= n; i++){
        if (n % i == 0){
            divisores.push_back(i);
        }
    }
    return divisores;
}

void printDatos(vector<int> v){
    cout << "La cantidad de divisores es: " << v.size() << endl;
    cout << "Los divisores son: " << endl;
    for (int i = 0; i < v.size(); i++){
        cout << v[i] << endl;
    }
}
```

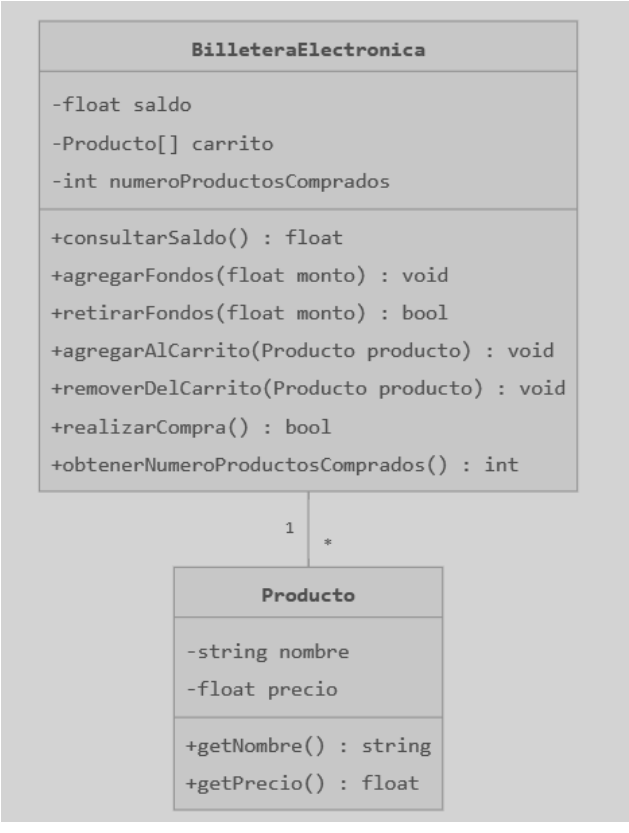
```
int main(){
    int n;
    cout << "Ingrese un numero para sacar su cantidad de divisores:
" << endl;
    cin >> n;
    vector<int> divisores = cantDivisores(n);
    printDatos(divisores);
    return 0;}
```

Ejecución

```
PS C:\UNSA - SISTEMAS\2024-UNSA\2024 - B\TO\lab01> cd "
 { .\ejer03 }
Ingrese un numero para sacar su cantidad de divisores:
36
La cantidad de divisores es: 9
Los divisores son:
1
2
3
4
6
9
12
18
36
```

4. **Proponer en forma gráfica una clase con atributos y métodos, referido a una billetera electrónica (manejar saldo, productos en el carrito, numero de productos comprados).**

Nota: Realizar el diagrama de clases.



Clase BilleteraElectronica:

- Atributos:
  - saldo (float): representa el saldo actual de la billetera.
  - carrito (Producto[]): un array de productos en el carrito.
  - numeroProductosComprados (int): contador de productos comprados.
- Métodos:
  - consultarSaldo(): devuelve el saldo actual.
  - agregarFondos(monto): añade fondos a la billetera.
  - retirarFondos(monto): retira fondos si hay suficiente saldo.
  - agregarAlCarrito(producto): añade un producto al carrito.
  - removerDelCarrito(producto): quita un producto del carrito.
  - realizarCompra(): procesa la compra de los productos en el carrito.
  - obtenerNumeroProductosComprados(): devuelve el número de productos comprados.

Clase Producto:

- Atributos:
  - nombre (string): nombre del producto.
  - precio (float): precio del producto.
- Métodos:
  - getNombre(): devuelve el nombre del producto.
  - getPrecio(): devuelve el precio del producto.

5. Implementar la estructura anterior con una clase.

Nota: Resolver con un Programa Orientado a Objetos que simule el funcionamiento de un carrito y una billetera electrónica.

```
#include <iostream>
#include <string>
#include <vector>

using namespace std;

void imprimirOpciones(); //funcion de opciones
bool buscarProducto(); //funcion de busqueda de producto

//clase producto
class Producto{
    private:
        string nombre;
        float precio;

    public:
        Producto(string name, float price){
            nombre = name;
            precio = price;
        }
```

```
        string getNombre(){
            return nombre;
        }

        float getPrecio(){
            return precio;
        }
};

//clase billetera electronica
class BilleteraElectronica {
    private:
        float saldo;
        vector<Producto> carrito;
        int numProductosComprados;

    public:
        //billetera recien generada
        BilleteraElectronica(float s){
            saldo = s;
            numProductosComprados = 0;
        }

        int consultarSaldo(){
            return saldo;
        }

        void agregarFondos(float f){
            saldo += f;
        }

        bool retirarFondos(float f){
            if(saldo >= f){
                saldo -= f;
                return true;
            }
            return false;
        }

        void agregarAlCarrito(Producto p){
            carrito.push_back(p);
        }

        void removerDelCarrito(Producto p){
            for(int i = 0; i , carrito.size(); i++){
                if(carrito[i].getNombre() == p.getNombre()){
                    carrito.erase(carrito.begin() + i);
                    break;
                }
            }
        }

        bool realizarCompra(){
            float total = 0;
            for(int i = 0; i < carrito.size(); i++){
```

```
                total += carrito[i].getPrecio();
            }
            if(saldo >= total){
                saldo -= total;
                numProductosComprados = carrito.size();
                carrito.clear();
                return true;
            }
            return false;
        }

        int obtenerNumeroProductosComprados(){
            return numProductosComprados;
        }
};

int buscarProducto(vector<Producto> productos, string nombre){
    for(int i = 0; i < productos.size(); i++){
        if(productos[i].getNombre() == nombre){
            return i;
        }
    }
    return -1;
}

void imprimirOpciones(){
    cout << "Seleccione una opcion: " << endl;
    cout << "1. Agregar Fondos" << endl;
    cout << "2. Retirar Fondos" << endl;
    cout << "3. Agregar Producto al Carrito" << endl;
    cout << "4. Remover Producto del Carrito" << endl;
    cout << "5. Realizar Compra" << endl;
    cout << "6. Consultar Saldo" << endl;
    cout << "7. Salir" << endl;
    cout << ">>>>>>>> ";
}

int main(){
    //Menu
    int opcion = 0;
    int i = 0;
    string nombre;
    BilleteraElectronica b(0);

    //Producto ya definidos
    vector <Producto> productos = {
        Producto("Papas", 10),
        Producto("Coca", 5),
        Producto("Chocolatina", 2),
        Producto("Galletas", 3),
        Producto("Chicles", 1)
    };

    cout << "BIENVENIDO AL SISTEMA" << endl;
    while (opcion != 8){
        imprimirOpciones();
```

```
        cin >> opcion;

        switch(opcion){
            case 1:
                float fondos;
                cout << "Ingrese la cantidad de fondos a agregar: ";
                cin >> fondos;
                b.agregarFondos(fondos);
                break;
            case 2:
                float retiro;
                cout << "Ingrese la cantidad de fondos a retirar: ";
                cin >> retiro;
                if(b.retirarFondos(retiro)){
                    cout << "Retiro exitoso" << endl;
                }else{
                    cout << "Fondos insuficientes" << endl;
                }
                break;
            case 3:
                float precio;
                cout << "Ingrese el nombre del producto: ";
                cin >> nombre;
                i = buscarProducto(productos, nombre);
                if ( i != -1){
                    b.agregarAlCarrito(productos[i]);
                }else{
                    cout << "Producto no encontrado" << endl;
                }
                break;
            case 4:
                cout << "Ingrese el nombre del producto a remover: ";
                cin >> nombre;
                i = buscarProducto(productos, nombre);
                if ( i != -1){
                    b.removerDelCarrito(productos[i]);
                }else{
                    cout << "Producto no encontrado" << endl;
                }
                break;
            case 5:
                if(b.realizarCompra()){
                    cout << "Compra realizada con exito" << endl;
                }else{
                    cout << "Fondos insuficientes" << endl;
                }
                break;
            case 6:
                cout << "Saldo: " << b.consultarSaldo() << endl;
                break;
            case 7:
                cout << "Gracias por usar el sistema" << endl;
                break;
            default:
                cout << "Opcion invalida" << endl;
                break;
```

```
        }

    }

    return 0;
}
```

Ejercicio

```
PS C:\UNSA - SISTEMAS\2024-UNSA\2024 - B\TO\lab01> cd "
2024-UNSA\2024 - B\TO\lab01\" ; if ($?) { g++ tempCodeR
mpCodeRunnerFile } ; if ($?) { .\tempCodeRunnerFile }
BIENVENIDO AL SISTEMA
Seleccione una opcion:
1. Agregar Fondos
2. Retirar Fondos
3. Agregar Producto al Carrito
4. Remover Producto del Carrito
5. Realizar Compra
6. Consultar Saldo
7. Salir
>>>>>>>> 6
Saldo: 0
Seleccione una opcion:
1. Agregar Fondos
2. Retirar Fondos
3. Agregar Producto al Carrito
4. Remover Producto del Carrito
5. Realizar Compra
6. Consultar Saldo
7. Salir
>>>>>>>> 1
Ingrese la cantidad de fondos a agregar: 12
Seleccione una opcion:
1. Agregar Fondos
2. Retirar Fondos
3. Agregar Producto al Carrito
4. Remover Producto del Carrito
5. Realizar Compra
6. Consultar Saldo
7. Salir
>>>>>>>> 2
Ingrese la cantidad de fondos a retirar: 13
Fondos insuficientes
```

```
Seleccione una opcion:
1. Agregar Fondos
2. Retirar Fondos
3. Agregar Producto al Carrito
4. Remover Producto del Carrito
5. Realizar Compra
6. Consultar Saldo
7. Salir
>>>>>>>> 3
Ingrese el nombre del producto: Papas
Seleccione una opcion:
1. Agregar Fondos
2. Retirar Fondos
3. Agregar Producto al Carrito
4. Remover Producto del Carrito
5. Realizar Compra
6. Consultar Saldo
7. Salir
>>>>>>>> 1
Ingrese la cantidad de fondos a agregar: 20
Seleccione una opcion:
1. Agregar Fondos
2. Retirar Fondos
3. Agregar Producto al Carrito
4. Remover Producto del Carrito
5. Realizar Compra
6. Consultar Saldo
7. Salir
>>>>>>>> 5
Compra realizada con exito
```

6. Cuestionario:

6.1. ¿Cuáles son los paradigmas de programación que se pueden utilizar con el lenguaje c++ moderno?

Los paradigmas comúnmente asociados con C++ incluyen programación genérica y orientada a objetos, de procedimiento.Porque C++ ofrece excelentes herramientas para la programación de alto nivel, incluso funcional estilo programación es bastante razonable. [3]

6.2. ¿Para qué sirven las directivas de precompilación?

Las directivas de precompilación son instrucciones que se procesan antes de la compilación del código. Se utilizan principalmente para [4]:

- Incluir archivos de encabezado: La directiva #include permite incluir otros archivos (como bibliotecas o archivos de encabezado) en tu programa, facilitando la reutilización de código.

```
#include <iostream>
```

- Definición de macros: #define permite definir macros que son sustituciones de texto en el código. Por ejemplo, puedes definir constantes o funciones simples.

```
#define PI 3.14
```

- Condicionales de compilación: #ifdef, #ifndef, #if, #else, #endif permiten incluir o excluir partes del código según ciertas condiciones. Esto es útil para manejar código que debe compilarse de manera diferente en distintas plataformas.

```
#ifdef DEBUG
cout << "Modo de depuración" << endl;
#endif
```

6.3. ¿Qué son los enum y para qué sirven?

Las enumeraciones ofrecen una manera sencilla de trabajar con conjuntos de constantes relacionadas. Una enumeración, o elemento Enum, es un nombre simbólico para un conjunto de valores. Las enumeraciones se tratan como tipos de datos y se pueden usar a fin de crear conjuntos de constantes para su uso con variables y propiedades.

6.4. ¿Qué son los struct y para qué sirven?

Un struct (estructura) es una forma de agrupar diferentes tipos de datos bajo un mismo nombre. A diferencia de las clases, los miembros de un struct son públicos por defecto. Se utilizan para representar un objeto o entidad que tiene múltiples atributos.

Por ejemplo, si quieres representar un `Producto` con un nombre y un precio, podrías definirlo así:

```
struct Producto {
    string nombre;
    float precio;
};
```

7. Referencias

[1] C++ reference, online: https://en.cppreference.com/w/

[2] Laaksonen, A. (2017). Guide to Competitive Programming. Springer. https://paginas.matem.unam.mx/pderbf/images/mprogintc++.pdf

[3] https://learn.microsoft.com/es-es/archive/msdn-magazine/2012/august/c-functional-style-programming-in-c

[4] https://learn.microsoft.com/es-es/cpp/preprocessor/preprocessor-directives?view=msvc-170