
Laboratorio

Plantillas, sobrecarga de operadores

Nombre: Gabriel Fernando Rodriguez Cutimbo

CUI: 20212157

Grupo: B

Repositorio GitHub:

https://github.com/gaco123/EPCC_CCII.git

1. Ejercicios

Resolver los siguientes ejercicios planteados:

1. Se pide escribir una función utilizando plantillas que tome tres argumentos genéricos y devuelva el menor y el máximo de ellos como valor de retorno. La función debe ser capaz de dar este tipo de resultados.

Código “Max_min.h”:

```
#ifndef MAX_MIN_H
#define MAX_MIN_H

template <class M>
class Max_min{
public:
    Max_min(M x=0, M y=0, M z=0);
    M getMax(){
        M a[3]={x,y,z};
        M temp=a[0];
        for(int i=0; i<3; i++){
            if(temp<a[i]){
                temp=a[i];
            }
        }
        return temp;
    };
    M getMin(){
```

```
        M a[3]={ x,y,z};
        M temp=a[0];
        for(int i=0; i<3; i++){
            if(temp>a[i]){
                temp=a[i];
            }
        }
        return temp;
};

private:
    M x;
    M y;
    M z;
};

template <class M>
Max_min<M>::Max_min(M x, M y, M z){
    this->x = x;
    this->y = y;
    this->z = z;
}

#endif
```

Código “Lab_09.cpp”:

```
#include <iostream>
#include <locale> //Caracteres Español
#include "Max_min.h"
using namespace std;

int main (){

    setlocale(LC_CTYPE, "Spanish");//Caracteres Español

    Max_min<int> MM1(2, 1, 5);
    Max_min<float> MM2(1.400023, 1.400012, 1.400000);
    Max_min<double> MM3(0.4000237, 0.4000238, 0.4000239);
    cout << "Máximo valor de MM1: " << MM1.getMax() << endl;
    cout << "Mínimo valor de MM1: " << MM1.getMin() << endl;
    cout.precision(6);
    cout << "Máximo valor de MM2: " << fixed << MM2.getMax() << endl;
    cout << "Mínimo valor de MM2: " << fixed << MM2.getMin() << endl;
    cout.precision(7);
    cout << "Máximo valor de MM3: " << fixed << MM3.getMax() << endl;
    cout << "Mínimo valor de MM3: " << fixed << MM3.getMin() << endl;

    return 0;
}
```

Funcionamiento:

```
Máximo valor de MM1: 5
Mínimo valor de MM1: 1
Máximo valor de MM2: 1.400023
Mínimo valor de MM2: 1.400000
Máximo valor de MM3: 0.4000239
Mínimo valor de MM3: 0.4000237
```

2. Se pide escribir una función utilizando plantillas que tome dos argumentos genéricos de tipo entero y flotante que devuelva las cuatro operaciones básicas.

Código “Operaciones.h”:

```
#ifndef OPERACIONES_H
#define OPERACIONES_H

template <class O>
class Operaciones{
public:
    Operaciones(O num1=0, O num2=0);
    O Suma(){
        return num1+num2;
    };
    O Resta(){
        return num1-num2;
    };
    O Mult(){
        return num1*num2;
    };
    O Div(){
        return num1/num2;
    };
private:
    O num1;
    O num2;
};

template <class O>
Operaciones<O>::Operaciones(O num1, O num2){
    this->num1 = num1;
    this->num2 = num2;
}

#endif
```

Código “Lab_09.cpp”:

```
#include <iostream>
#include <locale> //Caracteres Español
```

```
#include "Operaciones.h"
using namespace std;

int main (){

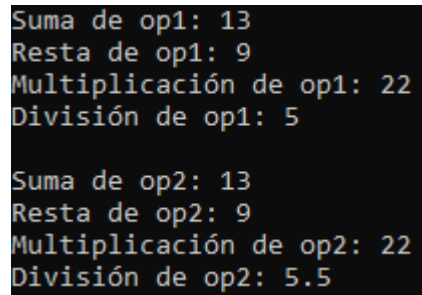
    setlocale(LC_CTYPE, "Spanish");//Caracteres Español

    Operaciones<int> op1(11, 2);
    Operaciones<float> op2(11, 2);

    //op1
    cout << "Suma de op1: " << op1.Suma() << endl;
    cout << "Resta de op1: " << op1.Resta() << endl;
    cout << "Multiplicación de op1: " << op1.Mult() << endl;
    cout << "División de op1: " << op1.Div() << endl;
    cout<<endl;

    //op2
    cout.precision(6);
    cout << "Suma de op2: " << op2.Suma() << endl;
    cout << "Resta de op2: " << op2.Resta() << endl;
    cout << "Multiplicación de op2: " << op2.Mult() << endl;
    cout << "División de op2: " << op2.Div() << endl;

    return 0;
}
```

Funcionamiento:

```
Suma de op1: 13
Resta de op1: 9
Multiplicación de op1: 22
División de op1: 5

Suma de op2: 13
Resta de op2: 9
Multiplicación de op2: 22
División de op2: 5.5
```

3. Se pide escribir una función utilizando plantillas que tome dos valores genéricos de tipo char y string (5 veces); char corresponde a una letra y string corresponde al apellido. El programa debe mostrar por pantalla el siguiente formato de correo electrónico: char/string@unsa.edu.pe.

Código “Correo.h”:

```
#ifndef CORREO_H
#define CORREO_H
```

```
#include <string>

template <class C, class S>
class Correo{
public:
    Correo(C ini='x', S ape="abcdefghi");
    S gen_email(){
        S temp = ini + ape + "@unsa.edu.pe";
        return temp;
    };
private:
    C ini;
    S ape;
};

template <class C, class S>
Correo<C,S>::Correo(C ini, S ape){
    this->ini = ini;
    this->ape = ape;
}

#endif
```

Código "Lab_09.cpp":

```
#include <iostream>
#include <locale> //Caracteres Español
#include <string>
#include "Correo.h"
using namespace std;

int main (){

    setlocale(LC_CTYPE, "Spanish"); //Caracteres Español

    Correo<char,string> pe1('g', "rodriguez");
    Correo<char,string> pe2('f', "cutimbo");
    Correo<char,string> pe3('m', "casas");
    Correo<char,string> pe4('j', "quispe");
    Correo<char,string> pe5('p', "mamani");

    cout<<"Email generado automáticamente de pe1: "<<pe1.gen_email()<<endl;
    cout<<"Email generado automáticamente de pe2: "<<pe2.gen_email()<<endl;
    cout<<"Email generado automáticamente de pe3: "<<pe3.gen_email()<<endl;
    cout<<"Email generado automáticamente de pe4: "<<pe4.gen_email()<<endl;
    cout<<"Email generado automáticamente de pe5: "<<pe5.gen_email()<<endl;

    return 0;
}
```

Funcionamiento:

```
Email generado automáticamente de pe1: grodriguez@unsa.edu.pe
Email generado automáticamente de pe2: fcutimbo@unsa.edu.pe
Email generado automáticamente de pe3: mcasas@unsa.edu.pe
Email generado automáticamente de pe4: jquispe@unsa.edu.pe
Email generado automáticamente de pe5: pmamani@unsa.edu.pe
```

4. Implemente un programa que haga uso de plantillas para determinar el mínimo y máximo valor de un arreglo de elementos dado. Debe de existir dos funciones, la primera que retorne el mayor de los valores y la segunda que retorne el menor de los valores. Asimismo, en la función main, se hace una prueba de estas funciones, con arreglos de enteros y flotantes.

```
int ArrayEntero [5] = {10,7,2, 8, 6};
float ArrayFloat [5] = {12.1, 8.7, 5.6, 8.4, 1.2};
```

Código “Maxmin_array.h”:

```
#ifndef MAXMIN_ARRAY_H
#define MAXMIN_ARRAY_H

template <class A, class T, class N>
class Maxmin_array{
public:
    Maxmin_array(A,N);
    T Max(){
        T temp = arrayx[0];
        for(int i=0; i<size; i++){
            if(temp<arrayx[i]){
                temp=arrayx[i];
            }
        }
        return temp;
    };
    T Min(){
        T temp = arrayx[0];
        for(int i=0; i<size; i++){
            if(temp>arrayx[i]){
                temp=arrayx[i];
            }
        }
        return temp;
    };
private:
    A arrayx;
```

```
        N size;
    };

    template <class A, class T, class N>
    Maxmin_array<A,T,N>::Maxmin_array(A arrayx, N size){
        this->arrayx = arrayx;
        this->size = size;
    }

#endif
```

Código “Lab_09.cpp”:

```
#include <iostream>
#include <locale> //Cáracteres Español
#include "Maxmin_array.h"
using namespace std;

int main (){

    setlocale(LC_CTYPE, "Spanish");//Cáracteres Español
    const int n=5;

    int a[n]={ 10,7,2,8,6};
    Maxmin_array<int*,int,int> b(a,n);

    float c[n]={ 12.1, 8.7, 5.6, 8.4, 1.2};
    Maxmin_array<float*,float,int> d(c,n);

    cout<<"Máximo valor del array a: "<<b.Max()<<endl;
    cout<<"Mínimo valor del array a: "<<b.Min()<<endl;
    cout<<endl;

    cout<<"Máximo valor del array c: "<<d.Max()<<endl;
    cout<<"Mínimo valor del array c: "<<d.Min()<<endl;

    return 0;
}
```

Funcionamiento:

```
Máximo valor del array a: 10
Mínimo valor del array a: 2

Máximo valor del array c: 12.1
Mínimo valor del array c: 1.2
```

5. Realizar la implementación de un programa que haga uso de plantillas, para elaborar una función que permita ordenar ascendentemente y descendientemente los elementos de un arreglo de valores enteros y otro arreglo de valores flotantes. Las funciones deben recibir como parámetros, un puntero al tipo de elemento dado, y dos enteros que indican los índices del primero y último elemento.

int ArrayEntero [5] = {5,7,2,8,6,1,3,4,9};
float ArrayFloat [5] = {10.1, 8.4, 3.6, 4.4, 11.2};

Código “Order.h”:

```
#ifndef ORDER_H
#define ORDER_H

template <class O, class P, class U>
class Order{
public:
    Order(O,P,U);
    void Ascend(){
        auto temp = arrayx[prim];
        int pos = prim;
        int prit = prim;
        int ultt = ult;

        for(int i=prit; i<=ultt; i++){
            if(temp<arrayx[i]){
                temp=arrayx[i];
                pos=i;
            }
            if(i==ultt&&prit!=ultt){
                arrayx[pos] = arrayx[prit];
                arrayx[prit] = temp;
                prit=prit+1;
                i=prit;
                temp=arrayx[prit];
                pos=prit;
            }
        }
    };
    void Descend(){
        auto temp = arrayx[prim];
        int pos = prim;
        int prit = prim;
        int ultt = ult;

        for(int i=prit; i<=ultt; i++){
```



```
        if(temp>arrayx[i]){
            temp=arrayx[i];
            pos=i;
        }
        if(i==ultt&&prit!=ultt){
            arrayx[pos] = arrayx[prit];
            arrayx[prit] = temp;
            prit=prit+1;
            i=prit;
            temp=arrayx[prit];
            pos=prit;
        }
    }
};

private:
    O arrayx;
    P prim;
    U ult;
};

template <class O, class P, class U>
Order<O,P,U>::Order(O arrayx, P prim, U ult){
    this->arrayx = arrayx;
    this->prim = prim;
    this->ult = ult;
}

#endif
```

Código “Lab_09.cpp”:

```
#include <iostream>
#include <locale>//Caracteres Español
#include "Order.h"
using namespace std;

int main (){

    setlocale(LC_CTYPE, "Spanish");//Caracteres Español

    const int n1 = 9;
    int a[n1]={5,7,2,8,6,1,3,4,9};
    cout<<"Array A \n";
    for(int i=0; i<n1; i++){
        cout<<a[i]<<" ";
    }

    Order<int*,int,int> a1(a,0,8);
    cout<<"\nArray A: Ordenado Ascendentemente\n";
    a1.Ascend();
    for(int i=0; i<n1; i++){
```

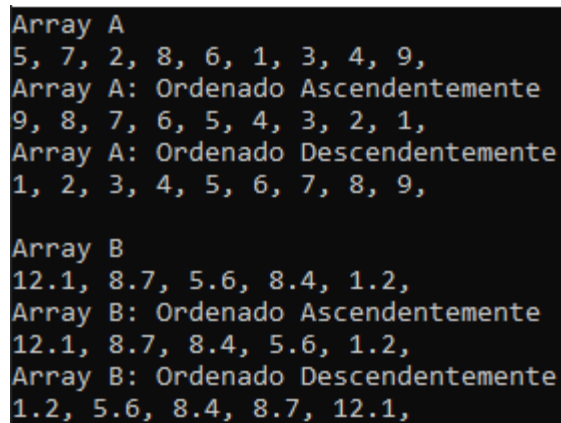
```
        cout<<a[i]<<" ";
    }
    cout<<"\nArray A: Ordenado Descendemente\n";
    a1.Descend();
    for(int i=0; i<n1; i++){
        cout<<a[i]<<" ";
    }
    cout<<endl<<endl;

    const int n2 = 5;
    float b[n2]={ 12.1, 8.7, 5.6, 8.4, 1.2};
    cout<<"Array B \n";
    for(int i=0; i<n2; i++){
        cout<<b[i]<<" ";
    }

    Order<float*,int,int> b1(b,0,4);
    cout<<"\nArray B: Ordenado Ascendemente\n";
    b1.Ascend();
    for(int i=0; i<n2; i++){
        cout<<b[i]<<" ";
    }
    cout<<"\nArray B: Ordenado Descendemente\n";
    b1.Descend();
    for(int i=0; i<n2; i++){
        cout<<b[i]<<" ";
    }

    return 0;
}
```

Funcionamiento:



```
Array A
5, 7, 2, 8, 6, 1, 3, 4, 9,
Array A: Ordenado Ascendemente
9, 8, 7, 6, 5, 4, 3, 2, 1,
Array A: Ordenado Descendemente
1, 2, 3, 4, 5, 6, 7, 8, 9,

Array B
12.1, 8.7, 5.6, 8.4, 1.2,
Array B: Ordenado Ascendemente
12.1, 8.7, 8.4, 5.6, 1.2,
Array B: Ordenado Descendemente
1.2, 5.6, 8.4, 8.7, 12.1,
```

2. Entregables

Al final estudiante deberá:

- 2.1.** Compactar el código elaborado y subirlo al aula virtual de trabajo. Agregue sus datos personales como comentario en cada archivo de código elaborado.
- 2.2.** Elaborar un documento que incluya tanto el código como capturas de pantalla de la ejecución del programa. Este documento debe de estar en formato PDF.
- 2.3.** El nombre del archivo (comprimido como el documento PDF), será su LAB09_GRUPO_A/B/C_CUI_1erNOMBRE_1erAPELLIDO.
(Ejemplo: LAB09_GRUPO_A_2022123_PEDRO_VASQUEZ).
- 2.4.** Debe remitir el documento ejecutable con el siguiente formato:
LAB09_GRUPO_A/B/C_CUI_EJECUTABLE_1erNOMBRE_1erAPELLIDO
(Ejemplo: LAB09_GRUPO_A_EJECUTABLE_2022123_PEDRO_VASQUEZ).

En caso de encontrarse trabajos similares, los alumnos involucrados no tendrán evaluación y serán sujetos a sanción.