UNIVERSIDAD NACIONAL DE SAN AGUSTÍN DE AREQUIPA

FACULTAD DE INGENIERIA DE PRODUCCION Y SERVICIOS

ESCUELA PROFESIONAL DE CIENCIAS DE LA COMPUTACIÓN

LABORATORIO 08 – Templates – Sobrecarga de Operadores.

DOCENTE:

Enzo Edir Velásquez Lobatón

ALUMNO:

Owen Haziel Roque Sosa.

FECHA:

22/05/2022

Arequipa – Perú

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.

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

template <class T>
// pair<dtype,dtype> incluido en STL
//https://www.geeksforgeeks.org/pair-in-cpp-stl/
pair<T,T> findMaxMin(T val1, T val2, T val3){
    pair<T,T> res;
    vector<T> arr = {val1, val2, val3};
    auto max = max_element(arr.begin(),arr.end());
    auto min = min_element(arr.begin(),arr.end());
    // https://en.cppreference.com/w/cpp/algorithm/max_element
    res.first = arr.at(distance(arr.begin(), max));
    res.second = arr.at(distance(arr.begin(), min));
    return res;
}

int main(int argc, char *argv[]) {
    auto res1 = findMaxMin(5,8,-1);
    cout<<"Maximo valor: "<<res1.first<<endl;
    cout<<"Minimo valor: "<<res1.second<<endl;
    cout<<endl;
    auto res2 = findMaxMin(6.2,-1.7,14.3);
    cout<<"Maximo valor: "<<res2.first<<endl;
    cout<<"Minimo valor: "<<res2.second<<endl;
    cout<<endl;
    auto res3 = findMaxMin(3.141519487,5.12385798,-4.21654651);
    cout<<"Maximo valor: "<<res3.first<<endl;
    cout<<"Minimo valor: "<<res3.second<<endl;
    return 0;
}
```

```
Maximo valor: 8
Minimo valor: -1

Maximo valor: 14.3
Minimo valor: -1.7

Maximo valor: 5.12386
Minimo valor: -4.21655
```

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

```cpp
#include <iostream>
using namespace std;
// T entero, S float
template <typename T, typename S>
S suma(T a, S b){
    return a + b;
}
template <typename T, typename S>
S resta(T a, S b){
    return a - b;
}
template <typename T, typename S>
S mult(T a, S b){
    return a * b;
}
template <typename T, typename S>
S divis(T a, S b){
    return a / b;
}
template <typename T, typename S>
void calculadora(T a, S b){
    cout<<"Suma: "<<suma(a,b)<<endl;
    cout<<"Resta: "<<resta(a,b)<<endl;
    cout<<"Multiplicacion: "<<mult(a,b)<<endl;
    cout<<"Division: "<<divis(a,b)<<endl;
}
// El resultado en base al datatype
// de la 2da variable
int main(int argc, char *argv[]) {
    calculadora(5,4.15);
    cout<<endl;
    calculadora(2.1,5.3);
    cout<<endl;
    return 0;
}
```

```
Suma: 9.15
Resta: 0.85
Multiplicacion: 20.75
Division: 1.20482

Suma: 7.4
Resta: -3.2
Multiplicacion: 11.13
Division: 0.396226
```

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.

```cpp
#include <iostream>
using namespace std;
template <class N, class A>
class Email{
    N nomb;
    A ap;
public:
    Email(N _nomb, A _ap){
        nomb = _nomb;
        ap = _ap;
    }
    A output(){
        A res = nomb + ap + "@unsa.edu.pe";
        return res;
    }
};

int main(int argc, char *argv[]) {
    Email<char,string> nombre1('o',"roque");
    Email<char,string> nombre2('y',"ramos");
    cout<<"Resultado nombre1: "<<nombre1.output();
    cout<<endl;
    cout<<"Resultado nombre2: "<<nombre2.output();
    return 0;
}
```

```
Resultado nombre1: oroque@unsa.edu.pe
Resultado nombre2: yramos@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};

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

template <typename S>
S minValue(S arr[5]){
    S min = arr[0];
    for(int i=0;i<5;i++){
        if (arr[i]<min)
            min=arr[i];
    }
    return min;
}
template <typename S>
S maxValue(S arr[5]){
    S max = arr[0];
    for(int i=0;i<5;i++){
        if (arr[i]>max)
            max=arr[i];
    }
    return max;
}

int main(int argc, char *argv[]) {
    int ArrayEntero [5] = {10,7,2, 8, 6};
    float ArrayFloat [5] = {12.1, 8.7, 5.6, 8.4, 1.2};
    cout<<"Max ArrayEntero: "<<maxValue(ArrayEntero)<<endl;
    cout<<"Min ArrayEntero: "<<minValue(ArrayEntero)<<endl;
    cout<<"Max ArrayFloat: "<<maxValue(ArrayFloat)<<endl;
    cout<<"Min ArrayFloat: "<<minValue(ArrayFloat)<<endl;
    return 0;
}
```

```
Max ArrayEntero: 10
Min ArrayEntero: 2
Max ArrayFloat: 12.1
Min ArrayFloat: 1.2
```

5. Realizar la implementación de un programa que haga uso de plantillas, para elaborar una función que permita ordenar ascendente y descendentemente 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};

```
#include <iostream>
using namespace std;
template <typename P>
void printArray(P* arr, int size){
    for (int i=0;i<=size;i++){
        cout<<arr[i]<<" - ";
    }
    cout<<endl;
    delete [] arr;
}
template <typename P>
P* ascendente(P* arr, int first, int last){
    //array copia a ordenar
    P* arr_ord = new P[last+1];
    for (int i=0;i<=last;i++){
        arr_ord[i]=arr[i];
    }
    //metodo de burbuja
    P aux;
    for(int i=0;i<=last;i++){
        for(int j=0;j<=last-1;j++){
            if(arr_ord[j] > arr_ord[j+1]){
                aux = arr_ord[j];
                arr_ord[j] = arr_ord[j+1];
                arr_ord[j+1] = aux;
            }
        }
    }
    return arr_ord;
}
```

```cpp
template <typename P>
P* descendente(P* arr, int first, int last){
    //array copia a ordenar
    P* arr_ord = new P[last+1];
    for (int i=0;i<=last;i++){
        arr_ord[i]=arr[i];
    }
    //metodo de burbuja
    P aux;
    for(int i=0;i<=last;i++){
        for(int j=0;j<=last-1;j++){
            if(arr_ord[j] < arr_ord[j+1]){
                aux = arr_ord[j];
                arr_ord[j] = arr_ord[j+1];
                arr_ord[j+1] = aux;
            }
        }
    }
    return arr_ord;
}
int main(int argc, char *argv[]) {
    int ArrayEntero [9] = {5,7,2,8,6,1,3,4,9};
    float ArrayFloat [5] = {10.1, 8.4, 3.6, 4.4, 11.2};
    int* ar1Ptr = ArrayEntero;
    float* ar2Ptr = ArrayFloat;
    cout<<"ArrayEntero ordenado Ascendente: ";
    printArray(ascendente(ar1Ptr,0,8),8);
    cout<<endl;
    cout<<"ArrayEntero ordenado Descendente: ";
    printArray(descendente(ar1Ptr,0,8),8);
    cout<<endl;
    cout<<"ArrayFloat ordenado Ascendente: ";
    printArray(ascendente(ar2Ptr,0,4),4);
    cout<<endl;
    cout<<"ArrayFloat ordenado Descendente: ";
    printArray(descendente(ar2Ptr,0,4),4);
    cout<<endl;
    return 0;
}
```

```
ArrayEntero ordenado Ascendente: 1 - 2 - 3 - 4 - 5 - 6 - 7 - 8 - 9 -

ArrayEntero ordenado Descendente: 9 - 8 - 7 - 6 - 5 - 4 - 3 - 2 - 1 -

ArrayFloat ordenado Ascendente: 3.6 - 4.4 - 8.4 - 10.1 - 11.2 -

ArrayFloat ordenado Descendente: 11.2 - 10.1 - 8.4 - 4.4 - 3.6 -
```