



**UNSA**  
UNIVERSIDAD NACIONAL DE SAN AGUSTÍN DE AREQUIPA

# **“UNIVERSIDAD NACIONAL DE SAN AGUSTÍN”**

**FACULTAD DE INGENIERÍA, PRODUCCIÓN Y SERVICIOS  
ESCUELA PROFESIONAL DE CIENCIA DE LA  
COMPUTACIÓN**

## **CURSO:**

Ciencias de la Computación - Grupo “B”

## **DOCENTE:**

Enzo Edir Velásquez Lobatón

## **ALUMNO:**

Fabricio Huaquisto Quispe

## **REPOSITORIO:**

<https://github.com/fhuaquisto21/EPCC-CCII>

**Arequipa - Perú**

**2022**

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:

```
#include <iostream>

using namespace std;

// maxAndMin: Devuelve el máximo y mínimo de 3 números;
template <class number>
pair<number, number> maxAndMin(number num1, number num2, number
num3) {
    pair<number, number> minmax;
    minmax.first = num1;
    minmax.second = num1;

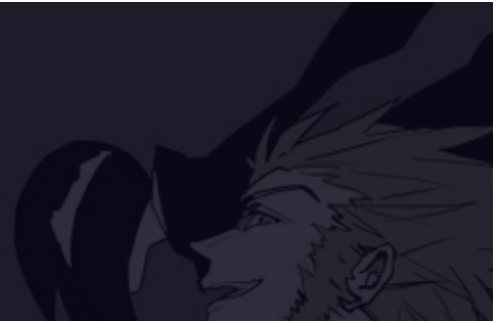
    if (num2 > minmax.second) {
        minmax.second = num2;
    } else if (num2 < minmax.first) {
        minmax.first = num2;
    }

    if (num3 > minmax.second) {
        minmax.second = num3;
    } else if (num3 < minmax.first) {
        minmax.first = num3;
    }

    return minmax;
}

int main() {
    int n1, n2, n3;
    printf("Valor 1: ");
    cin >> n1;
    printf("Valor 2: ");
    cin >> n2;
    printf("Valor 3: ");
    cin >> n3;
    pair<int, int> minmax = maxAndMin(n1, n2, n3);
    printf("El mínimo es: %i\nEl máximo es: %i\n", minmax.first,
minmax.second);
    return 0;
}
```

```
fhuaquisto: 09  
→ ./a.out  
Valor 1: 18  
Valor 2: 12  
Valor 3: 15  
El mínimo es: 12  
El máximo es: 18  
fhuaquisto: 09
```



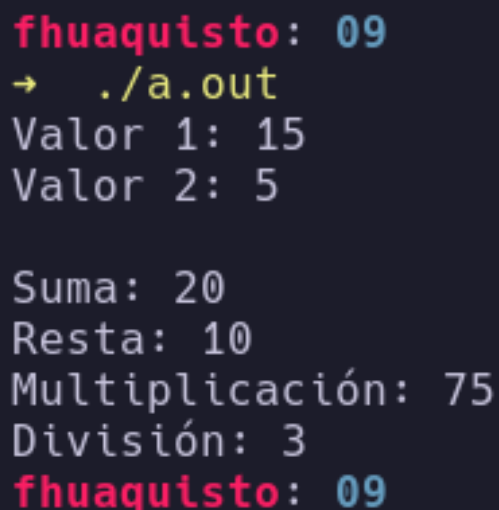
2. Se pide escribir una función utilizando plantillas que tome dos argumentos genéricos de tipo entero y flotante que devuelva las cuatro operaciones básicas:

```
#include <iostream>

using namespace std;

// basicOperation devuelve las 4 operaciones básicas.
template <class number>
void basicOperation(number n1, number n2) {
    cout << "Suma: " << n1 + n2 << endl;
    cout << "Resta: " << n1 - n2 << endl;
    cout << "Multiplicación: " << n1 * n2 << endl;
    cout << "División: " << n1 / n2 << endl;
}

int main() {
    int a, b;
    printf("Valor 1: ");
    cin >> a;
    printf("Valor 2: ");
    cin >> b;
    cout << endl;
    basicOperation(a, b);
    return 0;
}
```



```
fhuaquisto: 09
→ ./a.out
Valor 1: 15
Valor 2: 5

Suma: 20
Resta: 10
Multiplicación: 75
División: 3
fhuaquisto: 09
```

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:

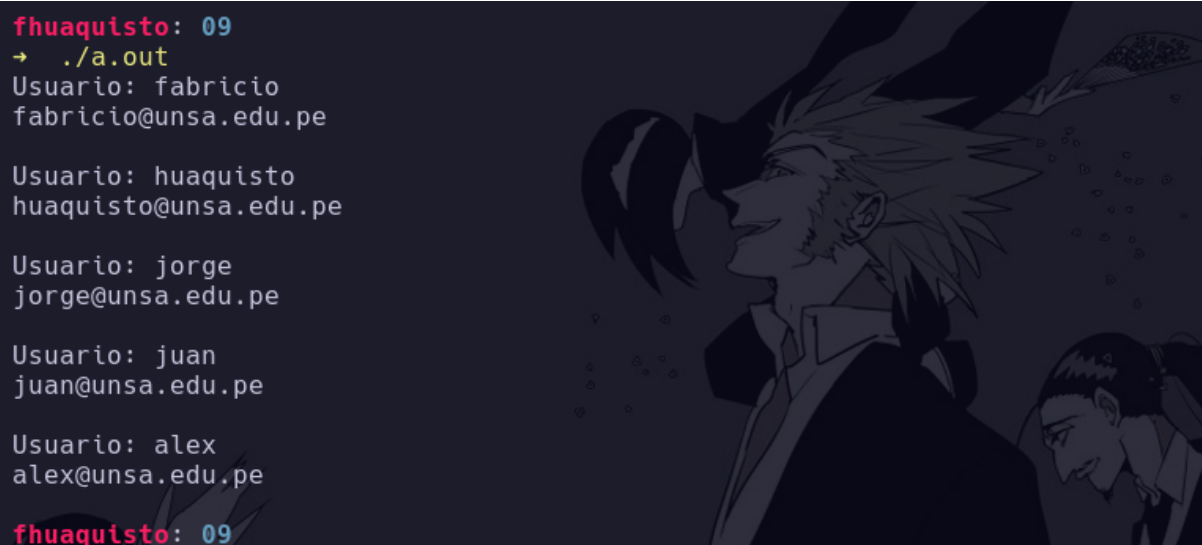
```
#include <iostream>
#include <string>

using namespace std;

template <class str>
void newEmail(str init) {
    cout << init << "@unsa.edu.pe" << endl;
}

int main() {
    string word;
    for (int i = 5; i; --i) {
        printf("Usuario: ");
        cin >> word;
        newEmail(word);
        cout << endl;
    }

    return 0;
}
```

A terminal window with a dark background and a faint anime-style illustration of two characters. The terminal shows the execution of a C++ program. The prompt 'fhuaquisto: 09' is shown in red and blue. The user runs './a.out' in green. The program then prompts 'Usuario:' five times, and the user enters 'fabricio', 'huaquisto', 'jorge', 'juan', and 'alex' respectively. The program outputs each name followed by '@unsa.edu.pe'.

```
fhuaquisto: 09
→ ./a.out
Usuario: fabricio
fabricio@unsa.edu.pe

Usuario: huaquisto
huaquisto@unsa.edu.pe

Usuario: jorge
jorge@unsa.edu.pe

Usuario: juan
juan@unsa.edu.pe

Usuario: alex
alex@unsa.edu.pe
fhuaquisto: 09
```

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:

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

using namespace std;

// minNumber devuelve el menor número en un vector;
template <class number>
number minNumber(vector<number> nums) {
    number min = nums[0];
    for (int i = 0; i < nums.size(); ++i) {
        if (nums[i] < min) {
            min = nums[i];
        }
    }
    return min;
}

// maxNumber devuelve el mayor número en un vector;
template <class number>
number maxNumber(vector<number> nums) {
    number max = nums[0];
    for (int i = 0; i < nums.size(); ++i) {
        if (nums[i] > max) {
            max = nums[i];
        }
    }
    return max;
}

int main() {
    vector<float> arr = { 4, 5, 8, 3, 9 };
    cout << "Mínimo: " << minNumber(arr) << endl;
    cout << "Máximo: " << maxNumber(arr) << endl;
    return 0;
}
```

**fhuaquisto: 09**

→ ./a.out

Mínimo: 3

Máximo: 9

**fhuaquisto: 09**

5. Realizar la implementación de un programa que haga uso de plantillas, para elaborar una función que permita ordenar ascendentemente 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:

```
#include <iostream>

using namespace std;

template <class number>
void ascendSort(number * nums, int length) {
    for (int i = 0; i < length; ++i) {
        for (int y = 0; y < length - 1; ++y) {
            if (nums[y] > nums[y + 1]) {
                number aux = nums[y];
                nums[y] = nums[y + 1];
                nums[y + 1] = aux;
            }
        }
    }
}

template <class number>
void descendSort(number * nums, int length) {
    for (int i = length - 1; i >= 0; --i) {
        for (int y = length - 1; y > 0; --y) {
            if (nums[y] > nums[y - 1]) {
                number aux = nums[y];
                nums[y] = nums[y - 1];
                nums[y - 1] = aux;
            }
        }
    }
}

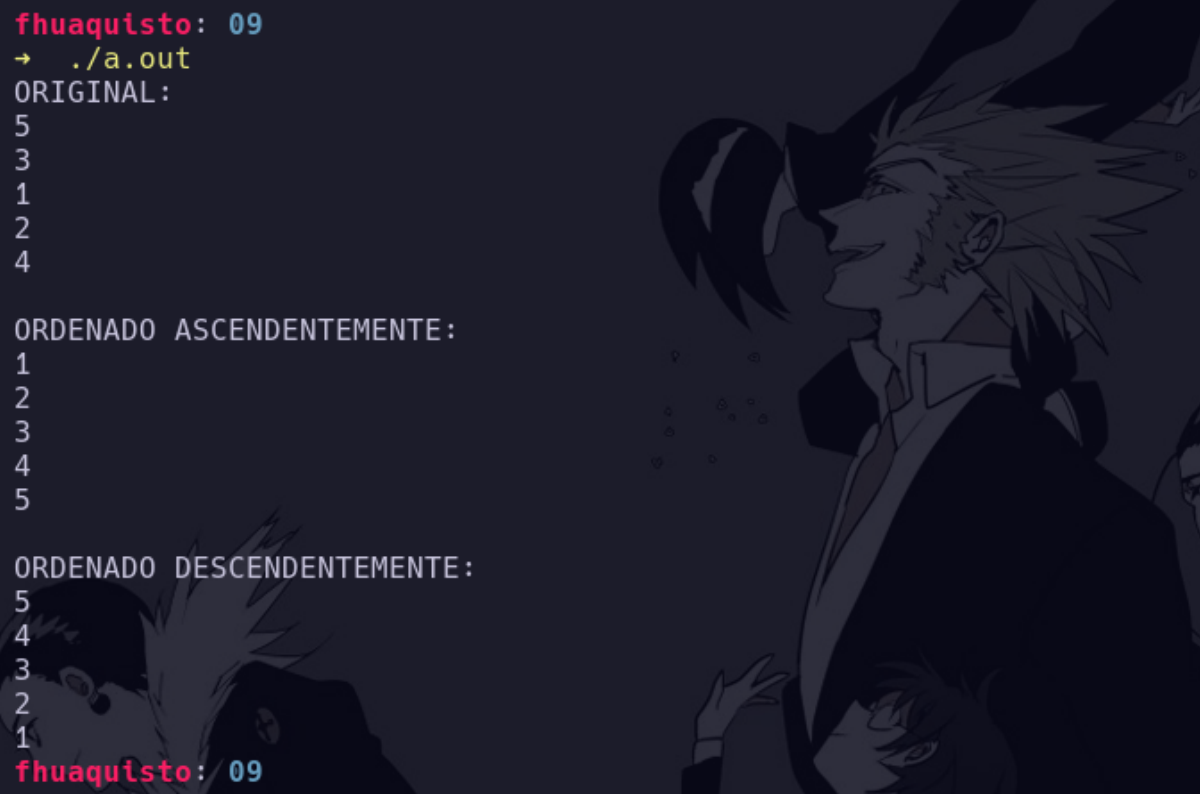
int main() {
    int *arr = new int;
    arr[0] = 5;
    arr[1] = 3;
    arr[2] = 1;
    arr[3] = 2;
    arr[4] = 4;

    cout << "ORIGINAL:" << endl;
    for (int i = 0; i < 5; ++i) {
```



```
    cout << arr[i] << endl;
}

ascendSort(arr, 5);
cout << endl;
cout << "ORDENADO ASCENDENTEMENTE:" << endl;
for (int i = 0; i < 5; ++i) {
    cout << arr[i] << endl;
}
descendSort(arr, 5);
cout << endl;
cout << "ORDENADO DESCENDENTEMENTE:" << endl;
for (int i = 0; i < 5; ++i) {
    cout << arr[i] << endl;
}
return 0;
}
```

A terminal window with a dark background featuring anime-style illustrations of characters. The text in the terminal shows the execution of a C++ program. The prompt is 'fhuaquisto: 09'. The user runs './a.out'. The output shows the original array [5, 3, 1, 2, 4], followed by the ascending sorted array [1, 2, 3, 4, 5], and then the descending sorted array [5, 4, 3, 2, 1]. The prompt 'fhuaquisto: 09' appears again at the bottom.

```
fhuaquisto: 09
→ ./a.out
ORIGINAL:
5
3
1
2
4

ORDENADO ASCENDENTEMENTE:
1
2
3
4
5

ORDENADO DESCENDENTEMENTE:
5
4
3
2
1
fhuaquisto: 09
```