UNIVERSIDAD NACIONAL DE SAN AGUSTÍN DE AREQUIPA

FACULTAD DE INGENIERIA DE PRODUCCION Y SERVICIOS

ESCUELA PROFESIONAL DE CIENCIAS DE LA COMPUTACIÓN

LABORATORIO 17 – Programación Genérica.

DOCENTE:

Enzo Edir Velásquez Lobatón

ALUMNO:

Owen Haziel Roque Sosa.

FECHA:

03/08/2022

Arequipa – Perú

1. Desarrolle un programa simple de calculadora (operaciones básicas) que utilice clases utilizando plantillas.

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

template <class T>
class calc {
public:
    T suma(T a, T b) {
        return a+b;
    }
    T resta(T a, T b){
        return a -b;
    }
    T mult(T a, T b){
        return a*b;
    }
    T div(T a, T b){
        return a/b;
    }
};
```

```cpp
int main(int argc, char *argv[]) {
    calc<int> *iCalc = new calc<int>;
    calc<float> *fCalc = new calc<float>;
    cout << "Suma Enteros: (5, 4) -> " <<  iCalc->suma(5, 4) << endl;
    cout << "Resta Enteros: (5, 4) -> " << iCalc->resta(5, 4) << endl;
    cout << "Multiplicacion Enteros: (5, 4) -> " << iCalc->mult(5, 4) << endl;
    cout << "Divison Enteros: (5, 4) -> " << iCalc->div(5, 4) << endl;
    cout << endl;
    cout << "Suma Flotantes: (7.5, 3.5) -> " <<  fCalc->suma(7.5, 3.5) << endl;
    cout << "Resta Flotantes: (7.5, 3.5) -> " << iCalc->resta(7.5, 3.5) << endl;
    cout << "Multiplicacion Flotantes: (7.5, 3.5) -> "
        << fCalc->mult(7.5, 3.5) << endl;
    cout << "Divison Flotantes: (7.5, 3.5) -> " << fCalc->div(7.5, 3.5) << endl;

    delete iCalc;
    delete fCalc;
    return 0;
}
```

```
Suma Enteros: (5, 4) -> 9
Resta Enteros: (5, 4) -> 1
Multiplicacion Enteros: (5, 4) -> 20
Divison Enteros: (5, 4) -> 1

Suma Flotantes: (7.5, 3.5) -> 11
Resta Flotantes: (7.5, 3.5) -> 4
Multiplicacion Flotantes: (7.5, 3.5) -> 26.25
Divison Flotantes: (7.5, 3.5) -> 2.14286
```

2. Definir una clase utilizando plantillas que permita almacenar datos en un ~~árbol binario~~ arreglo. Por el momento solo se insertarán elementos en la estructura. Simule el proceso de almacenar 100 datos y verifique que la estructura no tenga problemas.

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

template <class T>
class Array {
private:
    T* ptr;
    int size;
public:
    Array(T arr[], int s) {
        ptr = new T[s];
        size = s;
        for (int i = 0; i < size; i++)
            ptr[i] = arr[i];
    }
    void print() {
        for (int i = 0; i < size; i++)
            cout << " - " << *(ptr + i);
        cout << endl;
    };
    void insert(T add) {
        size++;
        T *tmpPtr = new T[size];
        for (int i = 0; i < size; i++)
            tmpPtr[i] = ptr[i];
        delete ptr;
        tmpPtr[size-1] = add;
        ptr = tmpPtr;
    }
};
```

```cpp
int main(int argc, char *argv[]) {
    float arr[5] = { 4.1, 2.56, 3.07, 5.15, 10.12 };
    Array<float> a(arr, 5);
    cout << "Array Inicial: " << endl;
    a.print();
    cout << "Agregando un valor: " << endl;
    a.insert(6.98);
    a.print();
    cout << "Agregando varios valores: " << endl;
    a.insert(7.102);
    a.insert(13.001);
    a.insert(20.99);
    a.insert(0.04);
    a.print();
    system("pause");

    return 0;
}
```

```
Array Inicial:
 - 4.1 - 2.56 - 3.07 - 5.15 - 10.12
Agregando un valor:
 - 4.1 - 2.56 - 3.07 - 5.15 - 10.12 - 6.98
Agregando varios valores:
 - 4.1 - 2.56 - 3.07 - 5.15 - 10.12 - 6.98 - 7.102 - 13.001 - 20.99 - 0.04
Presione una tecla para continuar . . .
```

3. Analice y describa el siguiente comportamiento:

```
#include <iostream>

template <class T>
class Contendor {
    T elemento;
    public :
    Contendor (T arg ) {
        elemento = arg ;
    }
    T add() { return ++elemento; }
};

template <>
class Contendor<char> {
    char elemento;
    public :
    Contendor ( char arg ) {
        elemento = arg ;
    }
    char uppercase() {
    if ((elemento >= 'a') && (elemento <= 'z')) {
        elemento += 'A'-'a'; }
    return elemento ;
    }
};

int main() {
    Contendor<int> cint (5) ;
    Contendor<char> cchar('t');
    std::cout << cint.add() << std::endl;
    std::cout << cchar.uppercase() << std::endl;
    return 0;
}
```

- Se define una clase Contendor con una plantilla T, según dicho valor se inicializará dicha clase y se define una función add(), que incrementará el valor del valor *elemento* en 1.
- A continuación, se define una especialización de plantilla. La definición de la clase anterior con un código diferente para el tipo de dato particular "*char*". Para la clase Contendor de tipo *char* se define una función distinta, uppercase(), la cual cambia la letra de la variable elemento (solo cambia el tipo de dato en el constructor) de minúscula a mayúscula.
- En el main() se prueba la implementación con plantillas de la clase Contendor, con una de tipo *int* y otra de tipo *char*, de modo que cada tipo tiene distintas funciones según su tipo de dato.