

UNIVERSIDAD NACIONAL DE SAN AGUSTÍN DE AREQUIPA  
FACULTAD DE PRODUCCIÓN Y SERVICIOS  
ESCUELA DE CIENCIAS DE LA COMPUTACIÓN



PRÁCTICA DE LABORATORIO 17  
CURSO DE CIENCIAS DE LA COMPUTACIÓN II

ESTUDIANTE:  
RUIZ MAMANI, EDUARDO GERMÁN

EMAIL: [eruizm@unsa.edu.pe](mailto:eruizm@unsa.edu.pe)

CUI: 20193061

TURNO:

C

AREQUIPA- PERÚ

2021

LINK DEL REPOSITORIO: [https://github.com/EGRM23/CCII\\_20193061.git](https://github.com/EGRM23/CCII_20193061.git)

## 1. EJERCICIO

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

- **CÓDIGO**

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

//EDUARDO GERMAN RUIZ MAMANI
//CUI 20193061

template <typename R>
R suma(R n1, R n2) {
    return n1 + n2;
}

template <typename R>
R resta(R n1, R n2) {
    return n1 - n2;
}

template <typename R>
R multiplicacion(R n1, R n2) {
    return n1 * n2;
}

template <typename R>
R division(R n1, R n2) {
    return n1 / n2;
}

int main(int argc, char *argv[]) {
    int n1 = 20, n2 = 15;
    cout << "Para " << n1 << " y " << n2 << endl;
    cout << "Suma: " << suma<int>(n1,n2) << endl;
    cout << "Resta: " << resta<int>(n1,n2) << endl;
    cout << "Multiplicacion: " << multiplicacion<int>(n1,n2)
<< endl;
    cout << "Division: " << division<int>(n1,n2) << endl;
    cout << endl;

    float f1 = 17.6, f2 = 5.3;
    cout << "Para " << f1 << " y " << f2 << endl;
    cout << "Suma: " << suma<float>(f1,f2) << endl;
    cout << "Resta: " << resta<float>(f1,f2) << endl;
    cout << "Multiplicacion: " << multiplicacion<float>(f1,f2)
<< endl;
    cout << "Division: " << division<float>(f1,f2) << endl;
    cout << endl;

    double d1 = 23.22222222, d2 = 8.146646;
    cout << "Para " << d1 << " y " << d2 << endl;
```

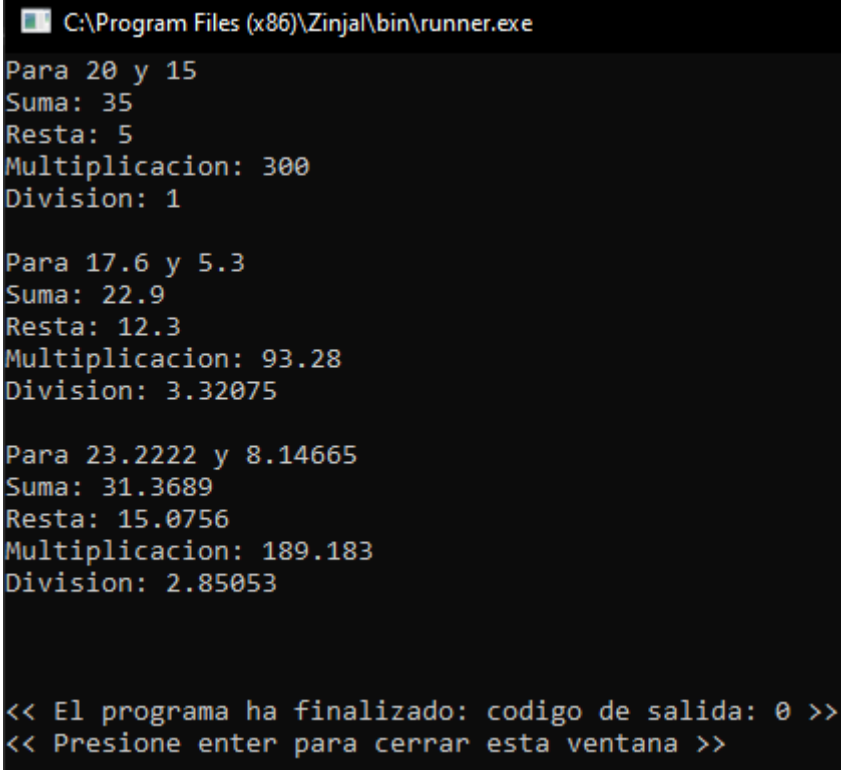
```

        cout << "Suma: " << suma<double>(d1,d2) << endl;
        cout << "Resta: " << resta<double>(d1,d2) << endl;
        cout << "Multiplicacion: " <<
multiplicacion<double>(d1,d2) << endl;
        cout << "Division: " << division<double>(d1,d2) << endl;
        cout << endl;

        return 0;
    }

```

- **CAPTURAS**



```

C:\Program Files (x86)\Zinjal\bin\runner.exe
Para 20 y 15
Suma: 35
Resta: 5
Multiplicacion: 300
Division: 1

Para 17.6 y 5.3
Suma: 22.9
Resta: 12.3
Multiplicacion: 93.28
Division: 3.32075

Para 23.2222 y 8.14665
Suma: 31.3689
Resta: 15.0756
Multiplicacion: 189.183
Division: 2.85053

<< El programa ha finalizado: codigo de salida: 0 >>
<< Presione enter para cerrar esta ventana >>

```

## 2. EJERCICIO

- Definir una clase utilizando plantillas que permita almacenar datos en un árbol binario. 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.

- **CÓDIGO**

- **main.cpp**

```

#include<iostream>
#include <stdlib.h>
#include <time.h>
#include "ArbolBinario.h"
using namespace std;

int main (int argc, char *argv[]) {
    srand(time(NULL));

    int cantnodos;
    cout << "Numero de datos: ";

```

```

    cin >> cantnodos;
    cout << endl;

    ArbolBinario<int>* arbolint = new ArbolBinario<int>();

    int valor;
    for(int i = 0; i < cantnodos; i++) {
        valor = 1 + rand()%(200 - 1);
        arbolint->insertar(valor);
    }

    arbolint->imprimir();

    delete arbolint;
    return 0;
}

```

- **Nodo.h**

```

#ifndef NODO_H
#define NODO_H

template <typename R> class ArbolBinario;
template <typename R> class Nodo {
public:
    Nodo(R val, int ord, Nodo* i = nullptr, Nodo* d = nullptr)
    {
        valor = val;
        izq = i;
        der = d;
        orden = ord;
    }
    ~Nodo() {}
    friend class ArbolBinario<R>;
private:
    Nodo* izq;
    Nodo* der;
    R valor;
    int orden;
};

#endif

```

- **ArbolBinario.h**

```

#ifndef ARBOLBINARIO_H
#define ARBOLBINARIO_H
#include "Nodo.h"

#include<iostream>
using namespace std;

template <typename R> class ArbolBinario {
public:
    ArbolBinario() {
        raiz = nullptr;
        cantnodos = 0;
    }

```

```

    }
    ~ArbolBinario();
    void insertar(R const);
    void imprimir();
private:
    void ubicar(R, Nodo<R>*);
    void infoxnodo(Nodo<R>*);
    void eliminar(Nodo<R>*);
    Nodo<R> *raiz;
    int cantnodos;
};

template <typename R>
ArbolBinario<R> :: ~ArbolBinario() {
    if (raiz != nullptr)
        eliminar(raiz);
}

template <typename R>
void ArbolBinario<R> :: eliminar(Nodo<R>* temp) {
    if (temp->izq != nullptr) { eliminar(temp->izq); }
    if (temp->der != nullptr) { eliminar(temp->der); }
    delete temp;
}

template <typename R>
void ArbolBinario<R> :: insertar(const R v) {
    if (raiz == nullptr) {
        Nodo<R>* nuevo = new Nodo<R>(v, cantnodos);
        raiz = nuevo;
        cantnodos++;
    } else {
        ubicar(v,raiz);
    }
}

template <typename R>
void ArbolBinario<R> :: ubicar(const R v, Nodo<R>* temp) {
    if ((temp->valor) < v) {
        if (temp->der == nullptr) {
            Nodo<R>* nuevo = new Nodo<R>(v, cantnodos);
            temp->der = nuevo;
            cantnodos++;
        } else {
            ubicar(v,temp->der);
        }
    } else {
        if (temp->izq == nullptr) {
            Nodo<R>* nuevo = new Nodo<R>(v, cantnodos);
            temp->izq = nuevo;
            cantnodos++;
        } else {
            ubicar(v,temp->izq);
        }
    }
}

```

```

}

template <typename R>
void ArbolBinario<R> :: imprimir() {
    if (raiz != nullptr) {
        cout << "Para visualizar, pegar la estructura
generada en este link:\n";
        cout <<
"https://dreampuf.github.io/GraphvizOnline/";
        cout << endl << endl;

        cout << "digraph ArbolBinario_" << cantnodos << "
{\n";
        cout << "\tn0_" << raiz->valor << ";\n";
        infoxnodo(raiz);
        cout << "}\n";
    }
}

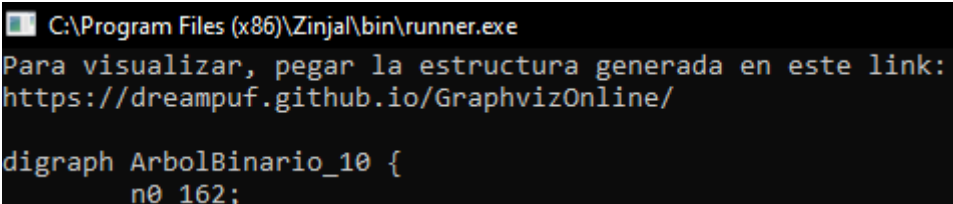
template <typename R>
void ArbolBinario<R> :: infoxnodo(Nodo<R>* temp) {
    if (temp->izq != nullptr) {
        cout << "\tn" << temp->orden << "_" << temp->valor
<< " -> ";
        cout << "n" << temp->izq->orden << "_" << temp-
>izq->valor << ";\n";
        infoxnodo(temp->izq);
    } else {
        cout << "\tn" << temp->orden << "_" << temp->valor;
        cout << " -> n" << temp->orden << "_nullizq;\n";
    }
    if (temp->der != nullptr) {
        cout << "\tn" << temp->orden << "_" << temp->valor
<< " -> ";
        cout << "n" << temp->der->orden << "_" << temp-
>der->valor << ";\n";
        infoxnodo(temp->der);
    } else {
        cout << "\tn" << temp->orden << "_" << temp->valor;
        cout << " -> n" << temp->orden << "_nulllder;\n";
    }
}
}
#endif

```

- **CAPTURAS**

Para la ejecución se tomó 2 ejemplos, uno con 10 datos y otro con 100

#### EJEMPLO DE 10 DATOS



```

C:\Program Files (x86)\Zinjal\bin\runner.exe
Para visualizar, pegar la estructura generada en este link:
https://dreampuf.github.io/GraphvizOnline/

digraph ArbolBinario_10 {
    n0_162;

```

```

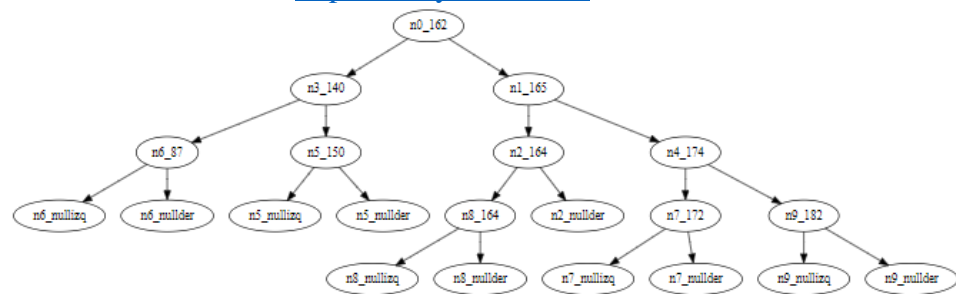
n0_162 -> n3_140;
n3_140 -> n6_87;
n6_87 -> n6_nullizq;
n6_87 -> n6_nulllder;
n3_140 -> n5_150;
n5_150 -> n5_nullizq;
n5_150 -> n5_nulllder;
n0_162 -> n1_165;
n1_165 -> n2_164;
n2_164 -> n8_164;
n8_164 -> n8_nullizq;
n8_164 -> n8_nulllder;
n2_164 -> n2_nulllder;
n1_165 -> n4_174;
n4_174 -> n7_172;
n7_172 -> n7_nullizq;
n7_172 -> n7_nulllder;
n4_174 -> n9_182;
n9_182 -> n9_nullizq;
n9_182 -> n9_nulllder;
}

<< El programa ha finalizado: codigo de salida: 0 >>
<< Presione enter para cerrar esta ventana >>

```

VISUALIZACIÓN (EN PNG fuera del PDF)

También en este link: <https://bit.ly/3PWZ6lR>



EJEMPLO DE 100 DATOS

```

C:\Program Files (x86)\Zinjal\bin\runner.exe
Numero de datos: 100

Para visualizar, pegar la estructura generada en este link:
https://dreampuf.github.io/GraphvizOnline/

digraph ArbolBinario_100 {
    n0_45;
    n0_45 -> n3_8;
    n3_8 -> n10_2;
    n10_2 -> n61_2;
    n61_2 -> n61_nullizq;
    n61_2 -> n61_nulllder;
    n10_2 -> n30_3;
    n30_3 -> n30_nullizq;
    n30_3 -> n38_7;
    n38_7 -> n87_6;

```

```
n87_6 -> n87_nullizq;  
n87_6 -> n87_nullder;  
n38_7 -> n38_nullder;  
n3_8 -> n6_28;  
n6_28 -> n13_23;  
n13_23 -> n14_9;  
n14_9 -> n14_nullizq;  
n14_9 -> n29_23;  
n29_23 -> n32_20;  
n32_20 -> n66_12;  
n66_12 -> n66_nullizq;  
n66_12 -> n68_20;  
n68_20 -> n98_20;  
n98_20 -> n98_nullizq;  
n98_20 -> n98_nullder;  
n68_20 -> n68_nullder;  
n32_20 -> n32_nullder;  
n29_23 -> n29_nullder;  
n13_23 -> n18_28;  
n18_28 -> n41_26;  
n41_26 -> n65_24;  
n65_24 -> n72_24;  
n72_24 -> n72_nullizq;  
n72_24 -> n72_nullder;  
n65_24 -> n65_nullder;  
n41_26 -> n41_nullder;  
n18_28 -> n18_nullder;  
n6_28 -> n12_41;  
n12_41 -> n22_29;  
n22_29 -> n22_nullizq;  
n22_29 -> n54_31;  
n54_31 -> n58_30;  
n58_30 -> n58_nullizq;  
n58_30 -> n97_31;  
n97_31 -> n97_nullizq;  
n97_31 -> n97_nullder;  
n54_31 -> n74_34;  
n74_34 -> n74_nullizq;  
n74_34 -> n74_nullder;  
n12_41 -> n49_43;  
n49_43 -> n86_42;  
n86_42 -> n86_nullizq;  
n86_42 -> n86_nullder;  
n49_43 -> n89_44;  
n89_44 -> n89_nullizq;  
n89_44 -> n89_nullder;  
n0_45 -> n1_79;  
n1_79 -> n11_73;  
n11_73 -> n20_59;  
n20_59 -> n33_58;  
n33_58 -> n40_49;  
n40_49 -> n40_nullizq;  
n40_49 -> n53_55;  
n53_55 -> n57_50;  
n57_50 -> n57_nullizq;  
n57_50 -> n64_51;  
n64_51 -> n64_nullizq;  
n64_51 -> n88_53;
```



```
n88_53 -> n95_53;
n95_53 -> n95_nullizq;
n95_53 -> n95_nulllder;
n88_53 -> n88_nulllder;
n53_55 -> n53_nulllder;
n33_58 -> n81_59;
n81_59 -> n81_nullizq;
n81_59 -> n81_nulllder;
n20_59 -> n28_64;
n28_64 -> n46_64;
n46_64 -> n55_62;
n55_62 -> n71_61;
n71_61 -> n71_nullizq;
n71_61 -> n71_nulllder;
n55_62 -> n55_nulllder;
n46_64 -> n46_nulllder;
n28_64 -> n44_67;
n44_67 -> n79_65;
n79_65 -> n79_nullizq;
n79_65 -> n79_nulllder;
n44_67 -> n69_69;
n69_69 -> n70_68;
n70_68 -> n70_nullizq;
n70_68 -> n70_nulllder;
n69_69 -> n69_nulllder;
n11_73 -> n16_76;
n16_76 -> n25_75;
n25_75 -> n25_nullizq;
n25_75 -> n25_nulllder;
n16_76 -> n16_nulllder;
n1_79 -> n2_182;
n2_182 -> n5_151;
n5_151 -> n7_134;
n7_134 -> n8_106;
n8_106 -> n45_85;
n45_85 -> n52_82;
n52_82 -> n52_nullizq;
n52_82 -> n83_84;
n83_84 -> n83_nullizq;
n83_84 -> n83_nulllder;
n45_85 -> n48_104;
n48_104 -> n50_101;
n50_101 -> n51_87;
n51_87 -> n51_nullizq;
n51_87 -> n78_99;
n78_99 -> n90_90;
n90_90 -> n90_nullizq;
n90_90 -> n90_nulllder;
n78_99 -> n78_nulllder;
n50_101 -> n73_104;
n73_104 -> n73_nullizq;
n73_104 -> n73_nulllder;
n48_104 -> n48_nulllder;
n8_106 -> n31_125;
n31_125 -> n34_113;
n34_113 -> n63_111;
n63_111 -> n85_110;
n85_110 -> n96_108;
```

```
n96_108 -> n96_nullizq;  
n96_108 -> n96_nullder;  
n85_110 -> n85_nullder;  
n63_111 -> n63_nullder;  
n34_113 -> n36_125;  
n36_125 -> n42_114;  
n42_114 -> n43_114;  
n43_114 -> n43_nullizq;  
n43_114 -> n43_nullder;  
n42_114 -> n42_nullder;  
n36_125 -> n36_nullder;  
n31_125 -> n37_130;  
n37_130 -> n37_nullizq;  
n37_130 -> n37_nullder;  
n7_134 -> n27_145;  
n27_145 -> n59_137;  
n59_137 -> n60_135;  
n60_135 -> n82_135;  
n82_135 -> n82_nullizq;  
n82_135 -> n82_nullder;  
n60_135 -> n60_nullder;  
n59_137 -> n77_138;  
n77_138 -> n77_nullizq;  
n77_138 -> n92_139;  
n92_139 -> n99_139;  
n99_139 -> n99_nullizq;  
n99_139 -> n99_nullder;  
n92_139 -> n92_nullder;  
n27_145 -> n27_nullder;  
n5_151 -> n9_162;  
n9_162 -> n15_156;  
n15_156 -> n26_155;  
n26_155 -> n62_154;  
n62_154 -> n84_154;  
n84_154 -> n84_nullizq;  
n84_154 -> n84_nullder;  
n62_154 -> n62_nullder;  
n26_155 -> n26_nullder;
```

```
n15_156 -> n39_161;  
n39_161 -> n56_161;  
n56_161 -> n56_nullizq;  
n56_161 -> n56_nullder;  
n39_161 -> n39_nullder;  
n9_162 -> n24_163;  
n24_163 -> n75_163;  
n75_163 -> n93_163;  
n93_163 -> n94_163;  
n94_163 -> n94_nullizq;  
n94_163 -> n94_nullder;  
n93_163 -> n93_nullder;  
n75_163 -> n75_nullder;  
n24_163 -> n35_171;  
n35_171 -> n67_167;  
n67_167 -> n67_nullizq;  
n67_167 -> n67_nullder;  
n35_171 -> n47_179;  
n47_179 -> n47_nullizq;  
n47_179 -> n47_nullder;
```

```

n2_182 -> n4_190;
n4_190 -> n17_188;
n17_188 -> n23_187;
n23_187 -> n76_184;
n76_184 -> n91_183;
n91_183 -> n91_nullizq;
n91_183 -> n91_nulllder;
n76_184 -> n76_nulllder;
n23_187 -> n23_nulllder;
n17_188 -> n80_189;
n80_189 -> n80_nullizq;
n80_189 -> n80_nulllder;
n4_190 -> n19_197;
n19_197 -> n19_nullizq;
n19_197 -> n21_198;
n21_198 -> n21_nullizq;
n21_198 -> n21_nulllder;
}

<< El programa ha finalizado: codigo de salida: 0 >>
<< Presione enter para cerrar esta ventana >>

```

VISUALIZACIÓN (MUY GRANDE) (EN PNG fuera del PDF)

Link demasiado grande



- **ADICIONAL**

- Los valores son generados aleatoriamente y van entre 0-200
- Al no encontrar alguna estrategia decente y no muy complicada de presentar el árbol gráficamente se optó por imprimirlo de manera que un software de visualización online (Graphviz) pueda mostrarlo gráficamente
- Para la impresión también se incluyó el poder mostrar los punteros null, porque de esta manera se podrá ver cuando un nodo está a la izquierda o la derecha.
- Por último, se incluyó en el nodo una variable que guarde el orden porque podía haber casos donde varios nodos podían tener el mismo valor y el software de visualización tomaba a todos los nodos como el mismo, generando errores.

### 3. EJERCICIO

3. Analice y describa el siguiente comportamiento del siguiente código:

- **CÓDIGO:**

```

#include <iostream>

template <class T>
class Contenedor {
    T elemento;

```

```

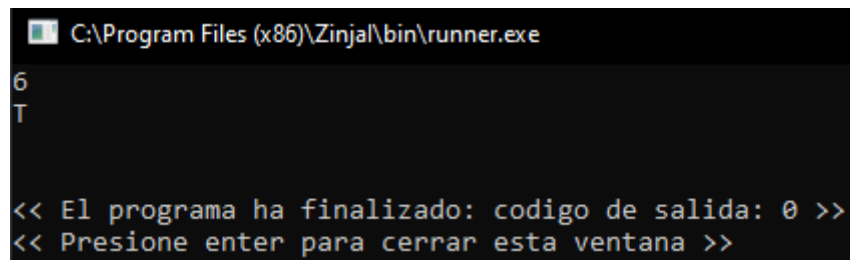
        public :
        Contenedor (T arg ) {
            elemento = arg;
        }
        T add() { return ++elemento; }
};
//La clase contenedor recibe un valor de tipo T y lo guarda en
su
//atributo elemento, por otro lado el método add aumenta el
valor
//de elemento en 1 unidad

template <>
class Contenedor<char> {
    char elemento;
    public :
    Contenedor ( char arg ) {
        elemento = arg ;
    }
    char uppercase() {
        if ((elemento >= 'a') && (elemento <= 'z')) {
            elemento += 'A'-'a'; }
        return elemento ;
    }
};
//Esta parte implementa la clase en caso de que la clase
Contenedor
//se cree con tipo char, ahí también recibe el valor y lo guarda
en
//elemento, la diferencia es que ahora no existe la función add,
//existe una función que vuelve a mayúscula la variable elemento

int main() {
    Contenedor<int> cint (5);
    Contenedor<char> cchar('t');
    std::cout << cint.add() << std::endl;
    std::cout << cchar.uppercase() << std::endl;
    //El código si funciona
    return 0;
}

```

- **CAPTURAS**



```

C:\Program Files (x86)\Zinjal\bin\runner.exe
6
T
<< El programa ha finalizado: codigo de salida: 0 >>
<< Presione enter para cerrar esta ventana >>

```