

Tipos de Datos Primitivo y Estructurado

Tarea 01

Aldo Alexandro Vargas Meza 213495653

24/01/2017



“Reporte de actividad, que consiste en un menú cíclico con 2 opciones y una opción de salir, con tipo de programación orientado a objetos.”

Problema:

Diseñe y codifique un programa que muestre un menú con las opciones:

- a. Tamaño y rangos de los Tipos de Dato Primitivos
- b. Ejemplo de uso de Tipo de dato Estructurado
- c. Salir

Requerimientos:

- a. El estilo de programación debe ser Orientado a Objetos
- b. Los datos para la tabla mostrada en la opción a) deben surgir de los propios recursos del lenguaje de programación, por lo que una simple transcripción de tamaños y rangos no será válida.
- c. Los valores aleatorios para la opción b) deben ser diferentes en cada ejecución.

Resolución de problema.

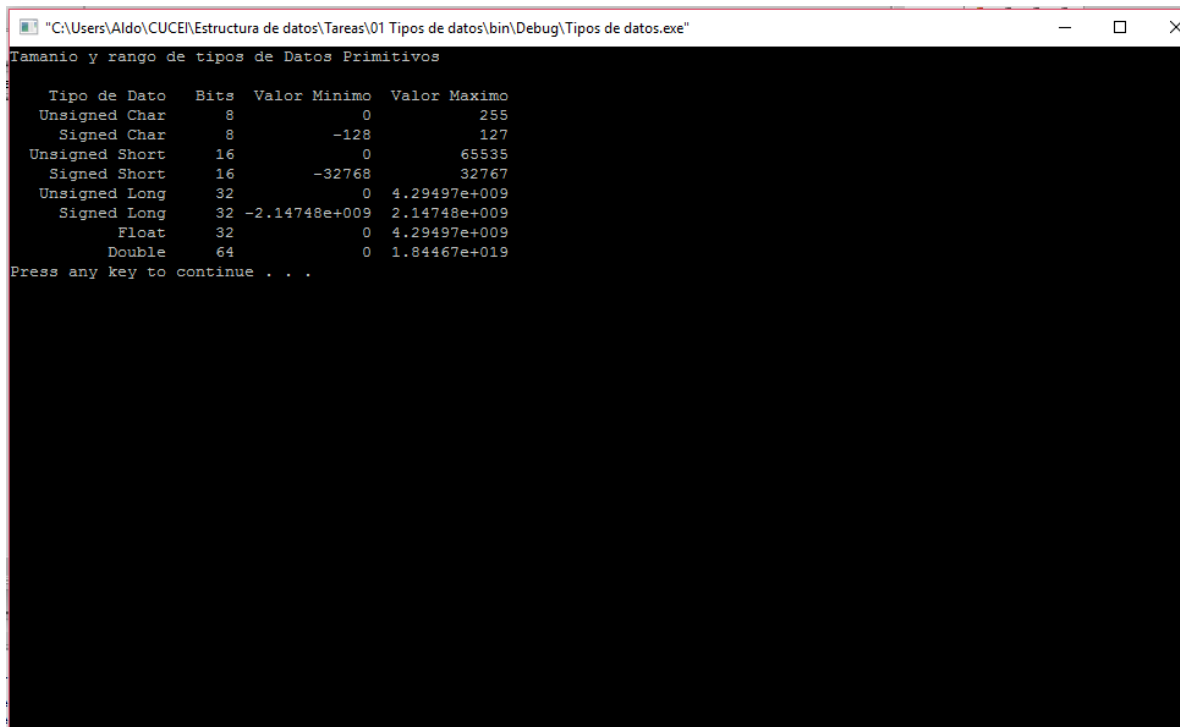
Para la **opción a)** el programa requiere la impresión de las propiedades que manejan los distintos tipos de datos primitivos, y especifica una lista de atributos que es necesario imprimir al usuario, con la condición de no usar una transcripción de texto, y generar los datos por medio de las capacidades del lenguaje de programación.

Para la resolución de este problema, se creó una clase con los atributos nombre, valor mínimo, valor máximo y tamaño en bits. De esta clase derivan dos métodos, los cuales generan la tabla y sus valores y los imprimen en un formato legible.

Para generar los valores, iniciamos obteniendo el tamaño en bits que representa el tipo de dato generado, esto lo hacemos con la función `sizeof()` que devuelve los bytes de cada tipo de dato. Con este valor, multiplicándolo por 8 obtenemos el tamaño en bits.

Con los bits obtenidos, podemos generar los rangos de valores de cada tipo de dato, con valores mínimos de cero y máximos de $2^{(\text{bits})} - 1$ para los datos unsigned, y mínimos de $-2^{(\text{bits})}/2$ y máximos de $2^{(\text{bits})}/2 - 1$.

Una vez los rangos definidos, el segundo método de la clase imprime los valores en una tabla, con un diseño legible con ayuda de la función setw(n) para reservar un ancho de espacio para la impresión.



```
"C:\Users\Aldo\CUCEI\Estructura de datos\Tareas\01 Tipos de datos\bin\Debug\Tipos de datos.exe"
Tamaño y rango de tipos de Datos Primitivos

Tipo de Dato  Bits  Valor Minimo  Valor Maximo
Unsigned Char    8         0         255
Signed Char     8       -128        127
Unsigned Short  16         0       65535
Signed Short    16      -32768      32767
Unsigned Long   32         0  4.29497e+009
Signed Long     32  -2.14748e+009  2.14748e+009
Float           32         0  4.29497e+009
Double          64         0  1.84467e+019
Press any key to continue . . .
```

La **opción b)** solicitará un valor entero n para determinar el tamaño de matriz a utilizar, con un mínimo de 3 y un máximo de 10, luego rellenará dos matrices de tamaño n x n con valores aleatorios de tipo real comprendidos en el rango entre -100.00 y 100.00, mostrará el contenido de las matrices en un formato legible y comprensible con dos decimales, además del resultado de la multiplicación de dichas matrices, así como el resultado de la suma de las mismas.

También definida en una clase, la matriz tendrá un atributo el cual será proporcionado por el usuario y definirá el tamaño de la matriz. Los métodos que la integran son 3, un getter y setter para el valor n, y una función que crea las matrices.

```
"C:\Users\Aldo\CUCE\Estructura de datos\Tareas\01 Tipos de datos\bin\Debug\Tipos de datos.exe"
Ingresa tamaño de Matriz [3 - 10]:
7
```

Dentro de la función **setter**, establecemos el límite con una condicional, en donde solo se aceptará un valor dentro del rango 3-10, y todo otro error no sea tomado como válido.

```
"C:\Users\Aldo\CUCE\Estructura de datos\Tareas\01 Tipos de datos\bin\Debug\Tipos de datos.exe"

Matriz generada:
|-30.14 || -94.21 || 86.87 || -64.06 || 35.09 || 28.5 || -14.38 |
|-27.84 || -70.48 || -65.11 || -98.86 || -88.92 || 87.89 || 39.39 |
|-66.54 || -77.7 || 92.57 || 21.69 || 63.59 || -97.28 || 32.99 |
|-27.12 || 39.89 || 42.49 || -67.03 || -61.12 || 72.65 || -33.63 |
| 51.56 || 42.54 || -6.94 || 81.62 || -35.74 || 65.36 || 32.92 |
| 63.99 || 30.32 || 70.22 || 81.39 || -87.38 || -47.17 || -53.48 |
|-25.27 || -94.55 || 37.39 || 58.62 || 75.71 || 86.33 || 56.18 |

Matriz Sumatoria:
| -60.28 || -188.42 || 173.74 || -128.12 || 70.18 || 57 || -28.76 |
|-55.68 || -140.96 || -130.22 || -197.72 || -177.84 || 175.78 || 78.78 |
|-133.08 || -155.4 || 185.14 || 43.38 || 127.18 || -194.56 || 65.98 |
|-54.24 || 79.78 || 84.98 || -134.06 || -122.24 || 145.3 || -67.26 |
| 103.12 || 85.08 || -13.88 || 163.24 || -71.48 || 130.72 || 65.84 |
| 127.98 || 60.64 || 140.44 || 162.78 || -174.76 || -94.34 || -106.96 |
|-50.54 || -189.1 || 74.78 || 117.24 || 151.42 || 172.66 || 112.36 |

Matriz Multiplicacion:
| 3484.54 || 3890.74 || 10055.5 || 21763.2 || 11925.8 || -22536.1 || 565.775 |
| 9858.74 || 3863.6 || 204.23 || 16170.2 || 5672.54 || -14393.2 || -6613.86 |
|-6359.06 || 2053.94 || 2730.71 || 11704.4 || 17860.3 || -4561.91 || 9369.99 |
| 1044.78 || -3449.35 || 400.273 || 2161.39 || -4409.7 || -16595.1 || -2169.51 |
|-2982.34 || -6711.88 || 10603.3 || -8797.63 || -9344.78 || 9236.1 || -4862.18 |
|-15824.8 || -10465.7 || 8837.73 || -25135.3 || 2235.79 || -4532.55 || -3504.86 |
| 7324.4 || 9004.13 || 17550.1 || 24346.8 || 319.435 || -2682.51 || -3067.2 |

Press any key to continue . . .
```

En la **función creadora de matrices**, definimos las matrices y elementos de apoyo para el método. La generación de matrices es con ciclos for, el cual ingresa un valor aleatorio que esta previamente

tratado para poder generar número negativos, y limitar a dos decimales el número flotante, después es ingresado por índices en el lugar correspondiente. Las demás matrices son generadas con algoritmos matemáticos, para poder hacer la multiplicación correctamente.

```
"C:\Users\Aldo\CUCE\Estructura de datos\Tareas\01 Tipos de datos\bin\Debug\Tipos de datos.exe"

Matriz generada:

|-29.84 || 89.83 || -77.61 |
| -61 || -20.57 || -11.99 |
| 70.56 || -95.85 || 41.19 |

Matriz Sumatoria:

| -59.68 || 179.66 || -155.22 |
| -122 || -41.14 || -23.98 |
| 141.12 || -191.7 || 82.38 |

Matriz Multiplicacion:

| -10065.4 || 2910.59 || -1957.94 |
| 2229 || -3907.26 || 4486.98 |
| 6647.71 || 4361.98 || -2630.3 |

Press any key to continue . . .
```

```
"C:\Users\Aldo\CUCE\Estructura de datos\Tareas\01 Tipos de datos\bin\Debug\Tipos de datos.exe"

Matriz generada:

|-29.3 || 61.11 || 66.3 || -95.49 || 79.21 || 35.11 || -16.51 || 21.75 || -37.77 || -41.16 |
| 63.21 || -77.63 || -36.85 || 81 || 22.36 || -33.12 || -32.7 || -55.04 || -67.11 || -66.78 |
|-40.66 || -58.62 || 88.9 || 40.67 || 76.4 || -26.23 || -64.95 || -10.13 || 13.23 || 68.21 |
|-37.66 || 99.98 || -64.55 || 58.39 || -22 || -75.51 || -42.69 || -22.62 || 55.21 || 26.67 |
|-59.77 || 87.17 || -30.63 || 96.18 || 68.1 || -22.47 || 26.69 || 87.78 || -43.98 || 65.06 |
| 21.91 || -36.28 || -51.75 || -75.95 || -79.18 || -79.29 || -37.92 || -19.91 || 4.9 || 83.12 |
|-66.49 || -2.09 || 29.83 || -56.69 || 21.18 || 94.65 || -33.11 || -69.58 || 24.81 || -90.39 |
| 82.38 || -10.89 || -25.8 || 62.05 || 92.47 || -50.82 || -54.71 || -71.93 || -43.12 || -32.55 |
| 59.49 || 67.83 || -77.89 || 23.25 || -77.72 || -65.47 || -80.27 || -51.39 || 57.63 || -3.83 |
|-74.09 || -25.56 || -5.61 || -98.52 || -11.36 || 74.01 || -39.21 || 21.3 || -71.26 || 85.04 |

Matriz Sumatoria:

| -58.6 || 122.22 || 132.6 || -190.98 || 158.42 || 70.22 || -39.02 || 49.5 || -75.54 || -82.32 |
| 126.42 || -155.26 || -73.7 || 162 || 44.72 || -66.24 || -65.4 || -110.08 || -134.22 || -133.56 |
|-81.32 || -117.24 || 177.8 || 81.34 || 152.8 || -52.46 || -129.9 || -20.26 || 26.46 || 136.42 |
|-75.32 || 199.96 || -129.1 || 116.78 || -44 || -151.02 || -85.38 || -45.24 || 110.42 || 53.34 |
|-119.54 || 174.34 || -61.26 || 192.36 || 136.2 || -44.94 || 53.38 || 175.56 || -87.96 || 130.12 |
| 183.82 || -72.56 || -103.5 || -151.9 || -158.36 || -158.58 || -75.84 || -39.82 || 8.6 || 166.24 |
|-102.98 || -4.18 || 59.66 || -119.38 || 42.36 || 189.3 || -66.22 || -139.16 || 49.62 || -180.78 |
| 164.76 || -21.78 || -51.6 || 124.1 || 184.94 || -101.64 || -109.42 || -143.86 || -86.24 || -65.1 |
| 118.98 || 135.66 || -155.78 || 46.5 || -155.44 || -130.94 || -160.54 || -102.78 || 115.26 || -7.66 |
|-148.18 || -51.12 || -11.22 || -197.04 || -22.72 || 148.02 || -78.42 || 42.6 || -142.52 || 170.08 |

Matriz Multiplicacion:

| 7806.35 || -16049.4 || 5739.48 || 15282.8 || 13890.5 || -5386.46 || 3040.86 || 4390.19 || -11313.1 || 4601.2 |
|-14096.3 || 21121 || 8622.47 || -969.615 || 3011.15 || 817.712 || 14392.4 || 15071.5 || 8134.21 || 256.312 |
|-15420.2 || 7932.22 || 670.374 || 11106 || 4530.07 || -5976.5 || -2920.11 || 13771.1 || 145.589 || 27477.6 |
| 4507.11 || 3779.43 || -16247 || 15770.3 || -10072.8 || -5399.75 || -1498 || -5088.39 || -1077.17 || -9028.39 |
|-3230.97 || 2085.74 || -15437.4 || 21896.3 || 10536.8 || -5433.59 || -6017.39 || -6063.15 || -12072.9 || 109.046 |
|-7598.18 || -1689.14 || 12842.4 || -27022.8 || 1149.53 || 22879.9 || 5906.26 || 6486.43 || -8530.75 || -7530.01 |
| 14813.9 || -8084.94 || -4184.79 || 5972.94 || -15876 || -14659.6 || 5080.98 || 3732.12 || 9441.7 || 10120.2 |
|-17029.2 || 22305 || 3126.74 || 7421.34 || 9177.66 || 87.3162 || 12902.2 || 20881 || -2002.8 || 3707.12 |
| 8279.68 || 5594.98 || -6747.83 || -1420.37 || -11229.9 || -1975.31 || 2267.97 || -1456.61 || 409.325 || -13787.5 |
| 5796.7 || -22897.1 || 1762.59 || -14180.4 || -5623.67 || 6384.16 || 6038.11 || 6279.52 || -12239.9 || 17514.5 |

Press any key to continue . . .
```

Al **final** en el main, diseñamos el menú cíclico el cual obtiene una variable del tipo carácter, para llevar a cabo las opciones. Esta variable debe estar entre a, b y c (minúsculas y mayúsculas). El menú cicla indefinidamente ya que está dentro de un ciclo while que define la condicional como la opción designada para salir del programa.

```
"C:\Users\Aldo\CUCE\Estructura de datos\Tareas\01 Tipos de datos\bin\Debug\Tipos de datos.exe"
Tipos de datos PRIMITIVO | ESTRUCTURADO
a) Tamano y rango de tipos de Datos Primitivos
b) Ejemplo de uso de tipo de Dato Estructurado
c) Salir
Seleccione una opcion:
```

Códigos

```
/* *****Main***** */
/* Program Principal datos primitivos y estructurados
   Aldo Vargas
   213495653
   file: main.cpp */

#include <iostream>
#include "dataTypes.h"
#include "matrix.h"
using namespace std;

int main(){
    char menuOpt = 's';
    DataTypes dt;
    Matrix mt;

    while(menuOpt){

        cout<< "Tipos de datos PRIMITIVO | ESTRUCTURADO " <<endl<<endl;
        cout<< "a) Tamano y rango de tipos de Datos Primitivos" <<endl;
        cout<< "b) Ejemplo de uso de tipo de Dato Estructurado" <<endl;
        cout<< "c) Salir" <<endl<<endl;
        cout<< "Seleccione una opcion: "; cin >> menuOpt;
        system("cls");

        if(menuOpt == 'a' || menuOpt == 'A'){
            cout<< "Tamano y rango de tipos de Datos Primitivos" <<endl<<endl;
            for(int i=0; i<8; i++){
                dt.setValues(i);
                dt.getValues(i);
            }
        }
    }
}
```

```

    }
    system("pause");
    system("cls");

} else if(menuOpt == 'b' || menuOpt == 'B'){
    cout<< "Ejemplo de uso de tipo de Dato Estructurado" <<endl<<endl;
    mtx.setN();
    mtx.createMatrix(mtx.getN());
    system("pause");
    system("cls");

} else if(menuOpt == 'c' || menuOpt == 'C'){
    cout << "Saliendo del programa..." <<endl;
    return 0;
}
else{
    cout << "Opcion incorrecta" <<endl;
    system("pause");
    system("cls");
}

}

return 0;
}

/*****Matrix.h*****/
/* Header Clase Matriz
Aldo Vargas
213495653
file: matrix.h*/

#ifndef MATRIX_H_INCLUDED
#define MATRIX_H_INCLUDED

class Matrix{
private:
    int n;

public:
    /*Imprime las matrices*/
    void createMatrix(const int&);
    /*Establece n*/
    void setN();
    int getN();
};

#endif // MATRIX_H_INCLUDED
/*****Matrix.cpp*****/
/* Clase Matriz
Aldo Vargas
213495653
file: matrix.cpp*/

#include "matrix.h"

```



```

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

void Matrix::setN(){
    system("cls");
    int x = 0;

    while(x == 0){
        cout<<"Ingresa tamaño de Matriz [3 - 10]: "<<endl;
        cin >> x;

        if(x>=3 && x<=10){
            n = x;
        } else{
            cout<<"Error de límites"<<endl;
            x = 0;
            system("pause");
            system("cls");
        }
    }
}

int Matrix::getN(){
    return n;
}

void Matrix::createMatrix(const int& n){
    /*Declaración de matrices*/
    float origMatrix[n][n];
    float sumaMatrix[n][n];
    float multMatrix[n][n];
    /*Declaración para aleatorio flotante de -100.00 a 100.00*/
    float random;
    int _2decimals;
    srand(time(NULL));

    system("cls");

    /*Generación de matrices*/
    for(int x=0; x<n; x++){
        for(int y=0; y<n; y++){
            /*Generación aleatorio entre -100.00 hasta 100.00*/
            random = ((static_cast<float>(rand())) / (static_cast<float>(RAND_MAX/(200.00))))-100.00;
            _2decimals = (random*100);
            random = (_2decimals*1.0)/100;

            origMatrix[x][y] = random;
            sumaMatrix[x][y] = (origMatrix[x][y] + origMatrix[x][y]);
        }
    }
}

```

```

for (int i = 0; i < n; i++){

    for (int j = 0; j < n; j++){
        multMatrix[i][j] = 0;

        for (int k = 0; k < n; k++){
            multMatrix[i][j]=multMatrix[i][j]+(origMatrix[i][k]*origMatrix[k][j]);

        }
    }
}

/*Impresion de matrices*/
cout<<endl<<" Matriz generada: "<<endl<<endl;
for(int x=0; x<n; x++){
    for(int y=0; y<n; y++){
        cout<<"| "<<setw(6)<<origMatrix[x][y]<<" |";

        if(y==n-1){
            cout<<endl;
        }
    }
}

cout<<endl<<" Matriz Sumatoria: "<<endl<<endl;
for(int x=0; x<n; x++){
    for(int y=0; y<n; y++){
        cout<<"| "<<setw(8)<<sumaMatrix[x][y]<<" |";

        if(y==n-1){
            cout<<" "<<endl;
        }
    }
}

cout<<endl<<" Matriz Multiplicacion: "<<endl<<endl;
for(int x=0; x<n; x++){
    for(int y=0; y<n; y++){
        cout<<"| "<<setw(10)<<multMatrix[x][y]<<" |";

        if(y==n-1){
            cout<<endl;
        }
    }
}
cout<<endl;

}
/*****dataTypes.h*****/
/* Header Tipos de Datos
Aldo Vargas
213495653
file: dataType.h*/
#ifndef DATATYPES_H_INCLUDED
#define DATATYPES_H_INCLUDED

```

```

#include <string>

class DataTypes{

private:
    std::string dataType;
    float minValue, maxValue;
    int bits;

public:
    /*Imprime la tabla*/
    void getValues(const int&);
    /*Calcula la tabla*/
    void setValues(const int&);

};

#endif // DATATYPES_H_INCLUDED

/*****dataTypes.cpp*****/
/* Clase Tipos de Datos
   Aldo Vargas
   213495653
   file: DataTypes.cpp*/
#include "dataTypes.h"
#include <iostream>
#include <iomanip>
#include <math.h>
using namespace std;

void DataTypes::getValues(const int& s){
    /*Imprime la tabla de tipos de datos*/
    if(s == 0){
        cout<<setw(16)<<"Tipo de Dato"<<setw(7)<<"Bits"<<setw(14)<<"Valor Minimo"<<setw(14)<<"Valor
Maximo"<<endl;

        cout<<setw(16)<<dataType<<setw(7)<<bits<<setw(14)<<minValue<<setw(14)<<maxValue<<endl;
    }
    else{

        cout<<setw(16)<<dataType<<setw(7)<<bits<<setw(14)<<minValue<<setw(14)<<maxValue<<endl;
    }
}

void DataTypes::setValues(const int& s){
    /*Characters*/
    if(s == 0){
        dataType = "Unsigned Char";           //Asigna el tipo de dato
        bits = sizeof(unsigned char) * 8;      //Obtiene su tamaño en bits
        maxValue = -1 + (pow(2,bits));          //Calculo de valor maximo
        minValue = 0;                           //Dado que es unsigned el min es cero

    }else if(s == 1){
        dataType = "Signed Char";
        bits = sizeof(signed char) * 8;
        maxValue = -1 + (pow(2,bits)/2);        //El valor maximo es la mitad del rango positivo
        minValue = 0 - (pow(2,bits)/2);         //y la mitad negativa será el minimo
    }
}

```

```

}
/*Integers*/
else if(s == 2){
    dataType = "Unsigned Short";
    bits = sizeof(unsigned short int) * 8;
    maxValue = -1 + (pow(2,bits));
    minValue = 0;

}
else if(s == 3){
    dataType = "Signed Short";
    bits = sizeof(signed short int) * 8;
    maxValue = -1 + (pow(2,bits)/2);
    minValue = 0 - (pow(2,bits)/2);
}
/*Longs*/
else if(s == 4){
    dataType = "Unsigned Long";
    bits = sizeof(unsigned long) * 8;
    maxValue = -1 + (pow(2,bits));
    minValue = 0;

}
else if(s == 5){
    dataType = "Signed Long";
    bits = sizeof(signed long) * 8;
    maxValue = -1 + (pow(2,bits)/2);
    minValue = 0 - (pow(2,bits)/2);

}
/*Floats*/
else if(s == 6){
    dataType = "Float";
    bits = sizeof(float) * 8;
    maxValue = -1 + (pow(2,bits));
    minValue = 0;

}
else if(s == 7){
    dataType = "Double";
    bits = sizeof(double) * 8;
    maxValue = -1 + (pow(2,bits));
    minValue = 0;

}

}

```