Informe de Laboratorio 01

Tema: Introducción

Nota

Estudiante	Escuela	Asignatura
Reyser Julio Zapata Butrón rzapata@unsa.edu.pe	Escuela Profesional de Ingeniería de Sistemas	Análisis Y Diseño de Algoritmos Semestre: IV Código: 1702231

Laboratorio	Tema	Duración
01	Introducción	02 horas

Semestre académico	Fecha de inicio	Fecha de entrega
2024 - B	17 septiembre 2024	17 septiembre 2024

1. Ejercicios

Se eligieron los ejercicios 2, 4, 7, 9 y 10 para el informe actual, sin embargo se realizaron los 10 ejercicios y se encuentran en el repositorio de Github.

En la mayoría de los ejercicios, se utilizaron argumentos en línea de comandos para proporcionar datos de entrada al programa de manera eficiente. Además, se emplearon arrays dinámicos para manejar datos de tamaño variable y optimizar el uso de memoria.

1.1. Ejercicio 2

- Desarrollar un programa que tenga como entrada un array de números y muestre en la salida los números con la respectiva posición que ocupa en el array.

```cpp
#include <iostream>
#include <string>
#include <cstdlib>
#include <filesystem>

int main(int argc, char *argv[])
{
  std::filesystem::path filePath(argv[0]);
  std::string fileName = filePath.filename().string();

  if (argc < 2)
  {
    std::cout << "\n[+] Uso: ./" << fileName << " <numero1> <numero2> ... <numeroN>\n\n";
```

```cpp
    return 1;
  }

  int *numbers = new int[argc - 1];

  for (int i = 1; i < argc; ++i)
  {
    numbers[i - 1] = std::atoi(argv[i]);
  }

  std::cout << "\n[+] Numeros ingresados:\n\n";

  for (int i = 0; i < argc - 1; ++i)
  {
    std::cout << i << " -> " << numbers[i] << std::endl;
  }

  delete[] numbers;
  printf("\n");
  return 0;
}
```

Listing 1: ejercicio2.cpp

Ejecución del ejercicio

```
> .\ejercicio2.exe

[+] Uso: ./ejercicio2.exe <numero1> <numero2> ... <numeroN>

> .\ejercicio2.exe 1 2 5 49 10 6

[+] Numeros ingresados:

0 -> 1
1 -> 2
2 -> 5
3 -> 49
4 -> 10
5 -> 6
```

1.2. Ejercicio 4

- Desarrollar un programa que lea la entrada estándar un array de enteros y determine el menor elemento del array.

```cpp
#include <iostream>
#include <cstdlib>
#include <string>
#include <filesystem>

int main(int argc, char *argv[])
{
  std::filesystem::path filePath(argv[0]);
  std::string fileName = filePath.filename().string();

  if (argc < 2)
  {
    std::cout << "\n[+] Uso: ./" << fileName << " <numero1> <numero2> ... <numeroN>\n\n";
```

```cpp
    return 1;
  }

  int *numbers = new int[argc - 1];
  int min = std::atoi(argv[1]);

  for (int i = 1; i < argc; ++i)
  {
    numbers[i - 1] = std::atoi(argv[i]);
  }

  for (int i = 1; i < argc - 1; ++i)
  {
    if (numbers[i] < min)
    {
      min = numbers[i];
    }
  }

  std::cout << "\n[+] Numero menor del array ingresado: " << min << "\n";

  delete[] numbers;
  printf("\n\n");
  return 0;
}
```

Listing 2: ejercicio4.cpp

Ejecución del ejercicio

```
> .\ejercicio4.exe

[+] Uso: ./ejercicio4.exe <numero1> <numero2> ... <numeroN>

> .\ejercicio4.exe 1 2 3 4 5 -5 -8 -10 0

[+] Numero menor del array ingresado: -10
```

1.3. Ejercicio 7

- Crear un programa que lea 10 elementos en un array, los copie a otro array multiplicado por un número que se ingrese por teclado. Mostrar el contenido del nuevo array.

```cpp
#include <iostream>
#include <cstdlib>
#include <string>
#include <filesystem>

void printArray(const int array[], int size, const std::string &name)
{
  std::cout << name << " = [";
  for (int i = 0; i < size; ++i)
  {
    std::cout << array[i];
    if (i < size - 1)
      std::cout << ", ";
  }
  std::cout << "]" << std::endl;
}

```

```cpp
int main(int argc, char *argv[])
{
  std::filesystem::path filePath(argv[0]);
  std::string fileName = filePath.filename().string();

  if (argc != 11)
  {
    std::cout << "\n[>] Uso: ./" << fileName << " <numero1> <numero2> ... <numero10>\n\n";
    return 1;
  }

  const int size = 10;
  int *numbers = new int[size];
  int *products = new int[size];
  int numberToMultiply;

  for (int i = 0; i < size; ++i)
  {
    numbers[i] = std::atoi(argv[i + 1]);
  }

  std::cout << "\n[+] Ingrese el numero a multiplicar: ";
  std::cin >> numberToMultiply;

  printArray(numbers, size, "\n[#] A1");

  for (int i = 0; i < size; ++i)
  {
    products[i] = numbers[i] * numberToMultiply;
  }

  printArray(products, size, "\n[+] A2");

  delete[] numbers;
  delete[] products;
  printf("\n\n");
  return 0;
}
```

Listing 3: ejercicio7.cpp

Ejecución del ejercicio

```
> .\ejercicio7.exe 1 2 3 4 5 6

[>] Uso: ./ejercicio7.exe <numero1> <numero2> ... <numero10>

> .\ejercicio7.exe 1 2 3 4 5 6 7 8 9 10 11

[>] Uso: ./ejercicio7.exe <numero1> <numero2> ... <numero10>

> .\ejercicio7.exe 1 -5 3 2 4 6 -7 -9 8 10

[+] Ingrese el numero a multiplicar: 3

[#] A1 = [1, -5, 3, 2, 4, 6, -7, -9, 8, 10]

[+] A2 = [3, -15, 9, 6, 12, 18, -21, -27, 24, 30]
```

1.4. Ejercicio 9

- Dada un array de enteros, mueva todos los ceros presentes al final del array. La solución debe mantener el orden relativo de los elementos en el array y no debe usar un espacio constante.

```cpp
#include <iostream>
#include <cstdlib>
#include <filesystem>

void printArray(const int array[], int size, const std::string &name)
{
  std::cout << name << " = [";
  for (int i = 0; i < size; ++i)
  {
    std::cout << array[i];
    if (i < size - 1)
      std::cout << ", ";
  }
  std::cout << "]" << std::endl;
}

void moveZerosToEnd(int arr[], int n)
{
  int lastNonZeroIndex = 0;

  for (int i = 0; i < n; ++i)
  {
    if (arr[i] != 0)
    {
      arr[lastNonZeroIndex] = arr[i];
      lastNonZeroIndex++;
    }
  }

  for (int i = lastNonZeroIndex; i < n; ++i)
  {
    arr[i] = 0;
  }
}

int main(int argc, char *argv[])
{

  std::filesystem::path filePath(argv[0]);
  std::string fileName = filePath.filename().string();

  if (argc < 2)
  {
    std::cout << "\n[>] Uso: ./" << fileName << " <numero1> <numero2> ... <numeroN>\n\n";
    return 1;
  }

  const int size = argc - 1;
  int *numbers = new int[size];

  for (int i = 1; i < argc; ++i)
  {
    numbers[i - 1] = std::atoi(argv[i]);
  }

  printArray(numbers, size, "\nA1");

  moveZerosToEnd(numbers, size);

  printArray(numbers, size, "\nA2");
```

```cpp

  delete[] numbers;
  printf("\n\n");
  return 0;
}
```

Listing 4: ejercicio9.cpp

Ejecución del ejercicio

```
> .\ejercicio9.exe
[>] Uso: ./ejercicio9.exe <numero1> <numero2> ... <numeroN>
> .\ejercicio9.exe 6 0 -8 -2 0 3 0 4 0 1 -1
A1 = [6, 0, -8, -2, 0, 3, 0, 4, 0, 1, -1]
A2 = [6, -8, -2, 3, 4, 1, -1, 0, 0, 0, 0]
```

1.5. Ejercicio 10

- Desarrollar un programa que permita ingresar varios números. Almacene los números pares en un array llamado numPar[] y los números impares en un array llamado numImpar[]. Finalmente deberá mostrar los números pares ingresados y luego los números impares.

```cpp
#include <iostream>
#include <cstdlib>
#include <filesystem>

void printArray(const int array[], int size, const std::string &name)
{
  std::cout << name << " = [";
  for (int i = 0; i < size; ++i)
  {
    std::cout << array[i];
    if (i < size - 1)
      std::cout << ", ";
  }
  std::cout << "]" << std::endl;
}

int main(int argc, char *argv[])
{
  std::filesystem::path filePath(argv[0]);
  std::string fileName = filePath.filename().string();

  if (argc < 2)
  {
    std::cout << "\n[>] Uso: ./" << fileName << " <numero1> <numero2> ... <numeroN>\n\n";
    return 1;
  }

  int size = argc - 1;
  int *numbers = new int[size];

  int *numPar = new int[size];
  int *numImpar = new int[size];
  int sizePar = 0;
```

```cpp
  int sizeImpar = 0;

  for (int i = 1; i < argc; ++i)
  {
    numbers[i - 1] = std::atoi(argv[i]);
  }

  for (int i = 0; i < size; ++i)
  {
    if (numbers[i] % 2 == 0)
    {
      numPar[sizePar++] = numbers[i];
    }
    else
    {
      numImpar[sizeImpar++] = numbers[i];
    }
  }

  printArray(numbers, size, "\n[#] A1: ");
  printArray(numPar, sizePar, "\n[+] numPar");
  printArray(numImpar, sizeImpar, "\n[+] numImpar");

  delete[] numbers;
  delete[] numPar;
  delete[] numImpar;
  printf("\n\n");
  return 0;
}
```

Listing 5: ejercicio10.cpp

Ejecución del ejercicio

```
> .\ejercicio10.exe
[>] Uso: ./ejercicio10.exe <numero1> <numero2> ... <numeroN>
> .\ejercicio10.exe 1 -5 8 -8 2 0 1 5 9 87 4 32 165
[#] A1:  = [1, -5, 8, -8, 2, 0, 1, 5, 9, 87, 4, 32, 165]
[+] numPar = [8, -8, 2, 0, 4, 32]
[+] numImpar = [1, -5, 1, 5, 9, 87, 165]
```

2. Repositorio de Github

- Repositorio de Github donde se encuentra el actual laboratorio
  `https://github.com/ReyserLynnn/ada-lab-b-24b/tree/main/laboratorio01/src`
- Repositorio de Github donde se encuentran los laboratorios del curso
  `https://github.com/ReyserLynnn/ada-lab-b-24b.git`

3. Cuestionario

1. **¿Cómo se declaran los arrays en C++?**

   En C++, tenemos 2 tipos de arrays: normales y dinámicos. Los arrays normales tienen un tamaño fijo, definido en tiempo de compilación, y se declaran como:

```cpp
int array[10];
```

   Los arrays dinámicos permiten un tamaño variable en tiempo de ejecución usando punteros y `new`, y se declaran como:

```cpp
int* array = new int[size];
```

2. **Para el siguiente caso: Se desea calcular el total a pagar, en una venta normal en una papelería, proporcionando el precio unitario de un producto, así como el número total de productos a comprar, además de aplicar el 18 % de IGV. ¿Qué entradas se requiere?¿Cuál es la salida deseada?¿Qué métodos produce la salida deseada?**

   - **Entradas requeridas:**
     - Precio unitario del producto.
     - Número total de productos a comprar.
   - **Salida deseada:**
     - Total a pagar después de aplicar el 18 % de IGV.
   - **Métodos para producir la salida deseada:**
     - Calcular el subtotal: `subtotal = precio_unitario * número_productos`.
     - Calcular el IGV (18 %): `IGV = subtotal * 0.18`.
     - Calcular el total a pagar: `total = subtotal + IGV`.