Chris Janes 23/02/25

Data Structures, Algorithms and Advanced Programming

Data Structures and You

Today’s topics

- What are Data Structures anyway?
- Advanced Data Structures
- Arrays
- Multidimensional Arrays
- Array Operations
- Traversal
- Insertion
- Deletion
- Searching
- Sorting

What are Data Structures anyway?

Data structures are a way for us to represent data in our applications.

- Two types:
  - Primitive
  - Non-Primitive

The primitive types in C++ are:

```
bool b = true;
int x = 5;
float y = 3.14f;
double z = 3.14159;
char c = 'x';
```

"Bad programmers worry about the code. Good programmers worry about data structures and their relationships."

– Linus Torvalds

Advanced Data Structures

These come in a few different forms which influence how they work and how they are used.

- Linear
- Static
- Homogeneous

vs.

- Non-linear
- Dynamic
- Heterogeneous

Arrays

Static, Linear, Homogenous

A simple data structure that stores elements in contiguous memory:


```
int* numbers = new int[6];
delete[] numbers;
```

```
int first = numbers[0];
numbers[5] = -5;
```

```
int first = numbers[-1];
numbers[6] = -5;
```

Multidimensional Arrays

Arrays all the way down.

```
int matrix[3][2]
```

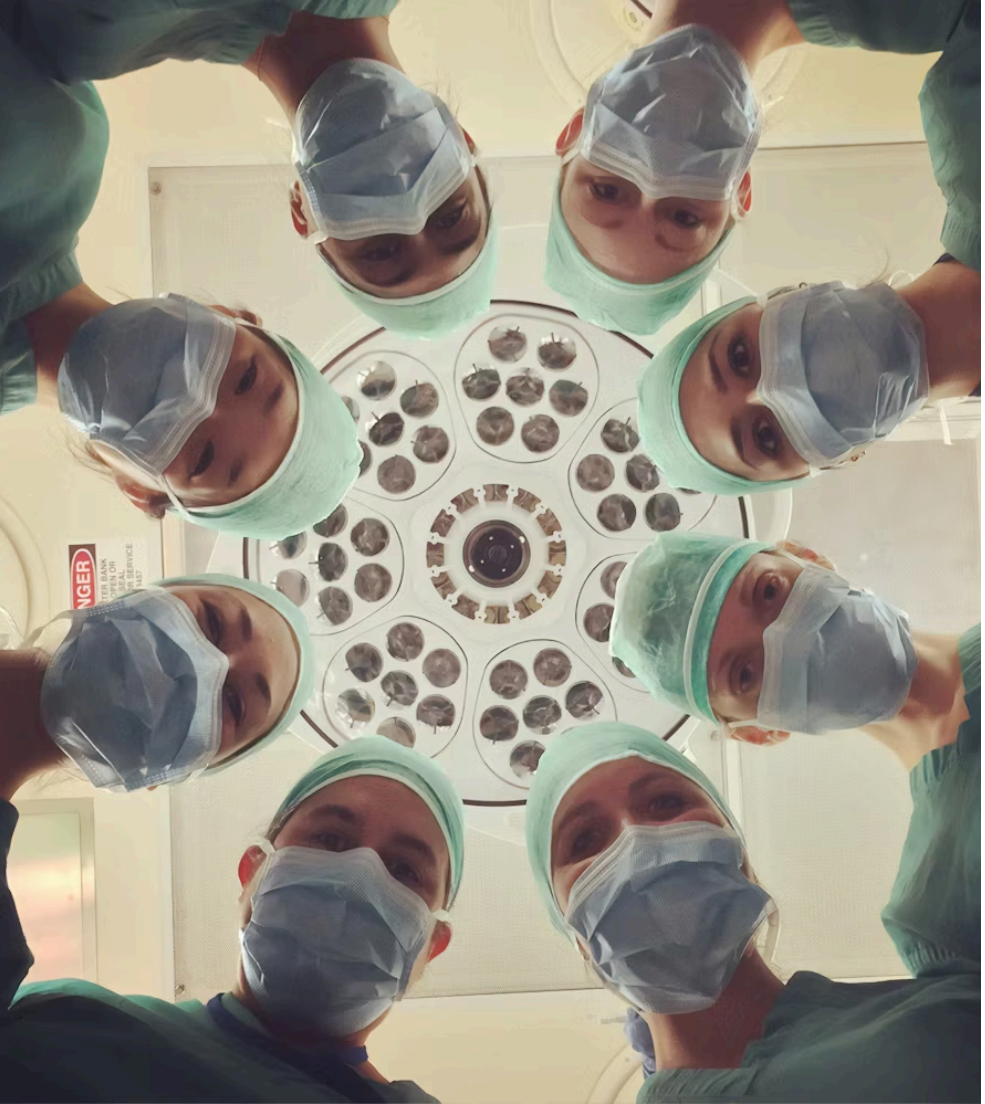
```
for (int i = 0; i < 3; ++i)
{
    for (int j = 0; j < 2; ++j)
    {
        matrix[i][j] = i * 3 + j * 2 + 1;
    }
}
```

```
matrix[0][3] = 1;
std::cout << matrix[1][1] // prints 1!
```

$$Address[i][j] = BaseAddress(BA) + \omega(n(i-1) + (j-1))$$

$$Address[0][3] = \omega(3(0-1) + (3-1))$$

$$Address[1][1] = \omega(3(1-1) + (1-1))$$



Array Operations

The most common operations are:

- Traversal
- Insertion
- Deletion
- Searching
- Sorting

Traversal

The process of visiting each element of an array

```
std::string words[] = {"hello", "world", "example", "string"};

for (int i = 0; i < 4; ++i)
{
  if (!words[i].empty()) {
    words[i][0] = std::toupper(words[i][0]);
  }
}

for (int i = 4; i >= 0; i--)
{
  if (!words[i].empty()) {
    words[i][0] = std::toupper(words[i][0]);
  }
}
```

Insertion

At the end

```
START
IF N = SIZE
  END
SET ARR[N] = NEW_ITEM
END
```

```
bool InsertIntoArrayAtEnd(int* numbersArray, int arrMax,
    int arrSize, int valToInsert)
{
    if (arrSize >= arrMax) return false;
    numbersArray[arrSize] = valToInsert;

    return true;
}
```

Deletion

At the end

```
START
SET N = N-1
END
```

```
int DeleteFromArrayAtEnd(int arrSize)
{
    return --arrSize;
}
```

Searching

Linear search iterates over an array from start to finish

```
int SearchArrayLinearly(int valToFind, int arr[], int arrSize)
{
  for (int i = 0; i < arrSize; i++)
  {
    if(arr[i] == valToFind)
      return i;
  }

  return -1;
}
```

Sorting