C++ Dynamic Memory Allocation

01 Uses of DMA

New Operator 02

03 Delete Operator

Dangling Pointer 04

05 Memory Leak

By – Mohammad Imran

Memory Management

Introduction

Memory management is the process of controlling and coordinating a computer's main memory. It ensures that blocks of memory space are properly managed and allocated so the operating system (OS), applications and other running processes have the memory they need to carry out their operations.

By – Mohammad Imran

Memory Management

Using C++

When we run a C++ program on our machine, it requires some space to store its instructions (statements), local variables, global variables, and various other functions in C++. This space required to run a C++ Program is known as **memory** in computers.

There are **two types of memory** in our system, **Static Memory and Dynamic Memory**.

By – Mohammad Imran

Types of Memory

Static Memory

It is a **constant space** allocated by the operating system during the **compile time** of a C++ program and it internally uses **stack** data structure to manage the static memory allocation. We can't reallocate the space consumed by the program until its execution is over.

Types of Memory

Dynamic Memory

It is the memory that can be allocated or de-allocated by the operating system during the **run-time** of a C++ program. It is more efficient than static memory because we can **de-allocate and reuse** our memory during the run-time of our program.

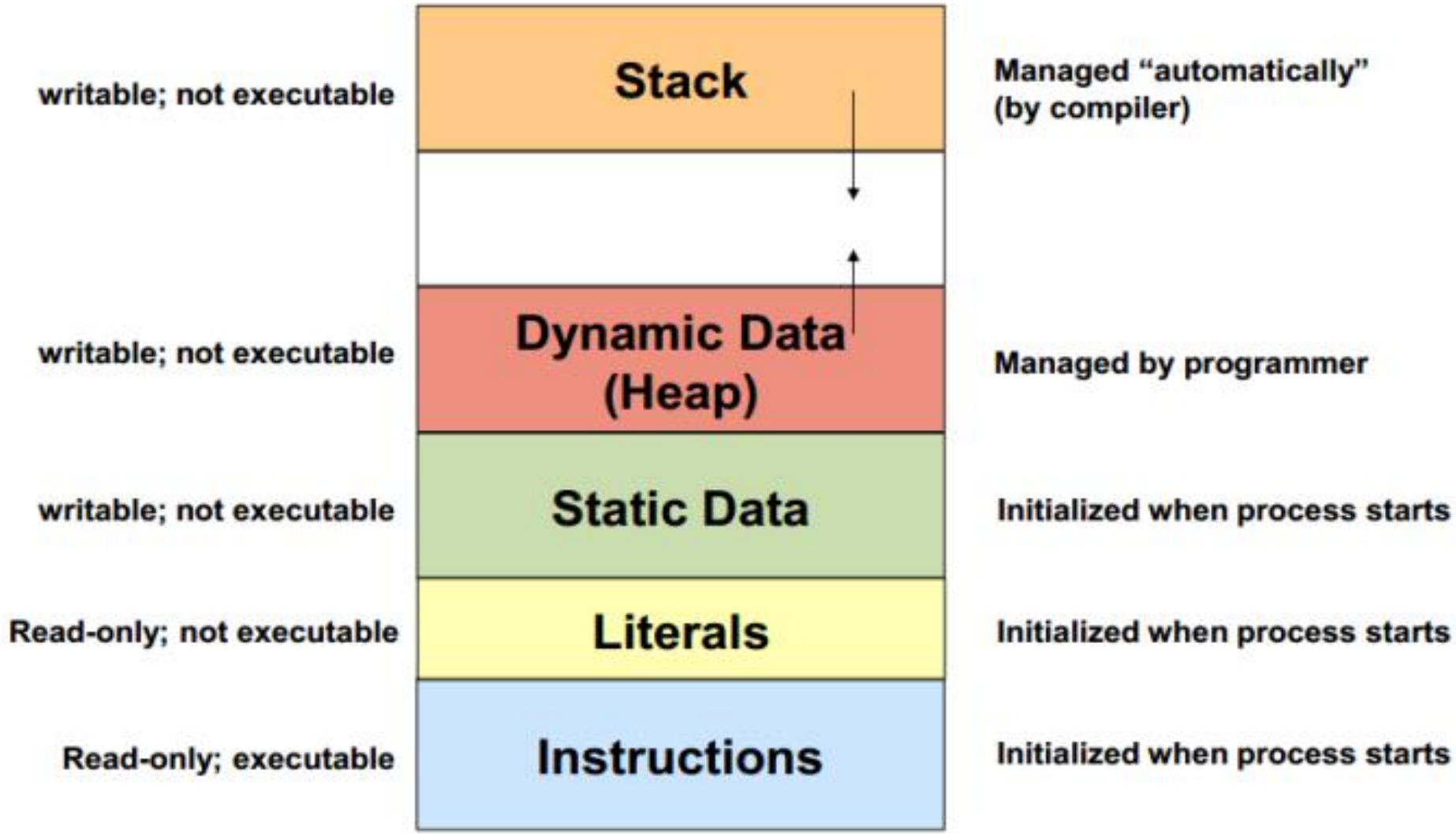
Types of Memory

Dynamic Memory

The memory used by a C++ program can be divided further into four parts:

- ✓ Run-time Stack (Static Memory)
- ✓ Static Data Memory (for Global and Static Variables)
- ✓ Instructions / Statements (Static Memory)
- ✓ Heap (Dynamic Memory)

Types of Memory



By – Mohammad Imran

Types of Memory

Stack Memory

- ✓ Our operating system **allocates a constant space** during compile-time of a C++ program, this space is known as **Stack memory**.
- ✓ Stack memory is used to **hold functions**, different variables, and local statements that exist in the function definition.
- ✓ Stack is a part of the static memory in our system and it constitutes the majority of our system's static memory.

By – Mohammad Imran

Types of Memory

Heap Memory

- ✓ **Heap memory** is also known as the **dynamic memory** in our system. It can be thought of as a large block of memory that is **expandable and shrinkable** during the execution of a program.
- ✓ **Allocation and De-allocation of memory blocks** during the execution of a program can be done using **new** and **delete** operators in C++.
- ✓ Heap memory can be expanded as long as we do not exhaust the machine memory itself. It is not good from a programming perspective to completely use the machine memory to avoid errors, thus we must use the heap memory carefully.

By – Mohammad Imran

Types of Memory Allocation

Static Memory Allocation in C++ (Compile-time Memory Allocation)

- ✓ When memory is allocated at compile-time, it is referred to as **Static Memory Allocation**.
- ✓ A fixed space is allocated for the local variables, function calls, and local statements, that **can not** be changed during the execution of the program.
- ✓ We **can not allocate or de-allocate a memory block** once the execution of the program starts.
- ✓ We **can't re-use** the static memory while the program is running. As a result, it is less effective.

By – Mohammad Imran

Types of Memory Allocation

Dynamic Memory Allocation (Runtime Memory Allocation)

- ✓ When memory is allocated or de-allocated during run-time, it is referred to as **Dynamic Memory Allocation** in C++.
- ✓ A variable space is allocated that **can** be changed during the execution of the program.
- ✓ We use dynamic/heap memory to allocate and de-allocate a block of memory during the execution of the program using **new** and **delete** operators.
- ✓ We **can** re-use our heap memory during the run-time of our program. As a result, it is highly effective.

By – Mohammad Imran

Dynamic Memory Allocation

Why Need

There were some drawbacks of **stack memory or static memory allocation**, like the space allocated for the stack **can not be expanded** during the execution of a C++ program or we can't keep variables in the program till the time we want. So, to overcome these drawbacks, we use the **Dynamic Memory Allocation concepts**.

By – Mohammad Imran

Dynamic Memory Allocation

Criteria

- ✓ **Creating the Dynamic Space in Memory.**

  During the dynamic memory allocation in C++, first, we have to create a dynamic space (in the heap memory). We use the **new** operator to create a dynamic space.

- ✓ **Storing its Address in a Pointer**

  Once a **dynamic space** a created, we have to store the address of the **allocated space in a pointer variable** to access and modify the contents of the **memory block**.

- ✓ **Deleting the Allocated Space**

  Once the user does not require the memory block, we delete the allocated space using the **delete** operator to free up the heap memory.

By – Mohammad Imran

Dynamic Memory Allocation

new Operator

The **new** operator in C++ is used to **dynamically allocate a block of memory and store its address** in a pointer variable during the execution of a C++ program if enough memory is available in the system.

Syntax for "new" operator:

```
data_type* ptr_var = new data_type;
```

By – Mohammad Imran

Dynamic Memory Allocation

Program - 1

```
#include <iostream>
using namespace std;
int main()
{
   int* ptr = new int;
   *ptr = 12;
   cout << "Value at ptr = " << *ptr;
   return 0;
}
```

OUTPUT

```
Value at ptr = 12
```

By – Mohammad Imran

Dynamic Memory Allocation

Program - 2

```
#include <iostream>
using namespace std;
int main()
{
   int* ptr = new int(12);
   cout << "Value at ptr = " << *ptr;
   return 0;
}
```

OUTPUT

```
Value at ptr = 12
```

By – Mohammad Imran

Dynamic Memory De-allocation

delete Operator

The **delete** operator is used to de-allocate the block of memory, which is dynamically allocated using the **new** operator. Since, a programmer must de-allocate a block of memory, once it is not required in the program. So, we have to use the **delete** operator to avoid memory leaks and program crash errors, which occur due to the exhaustion of the system's memory.

Syntax for "new" operator:

```
delete ptr_var;
```

By – Mohammad Imran

Dynamic Memory De-allocation

Program - 3

```
#include <iostream>
using namespace std;
int main()
{
   int* stars = new int;
   *stars = 5000;
   cout << "Stars in the sky: " << *stars;
   delete stars;
   cout << "\nGarbage value: " << *stars;
   stars = NULL;
   return 0;
}
```

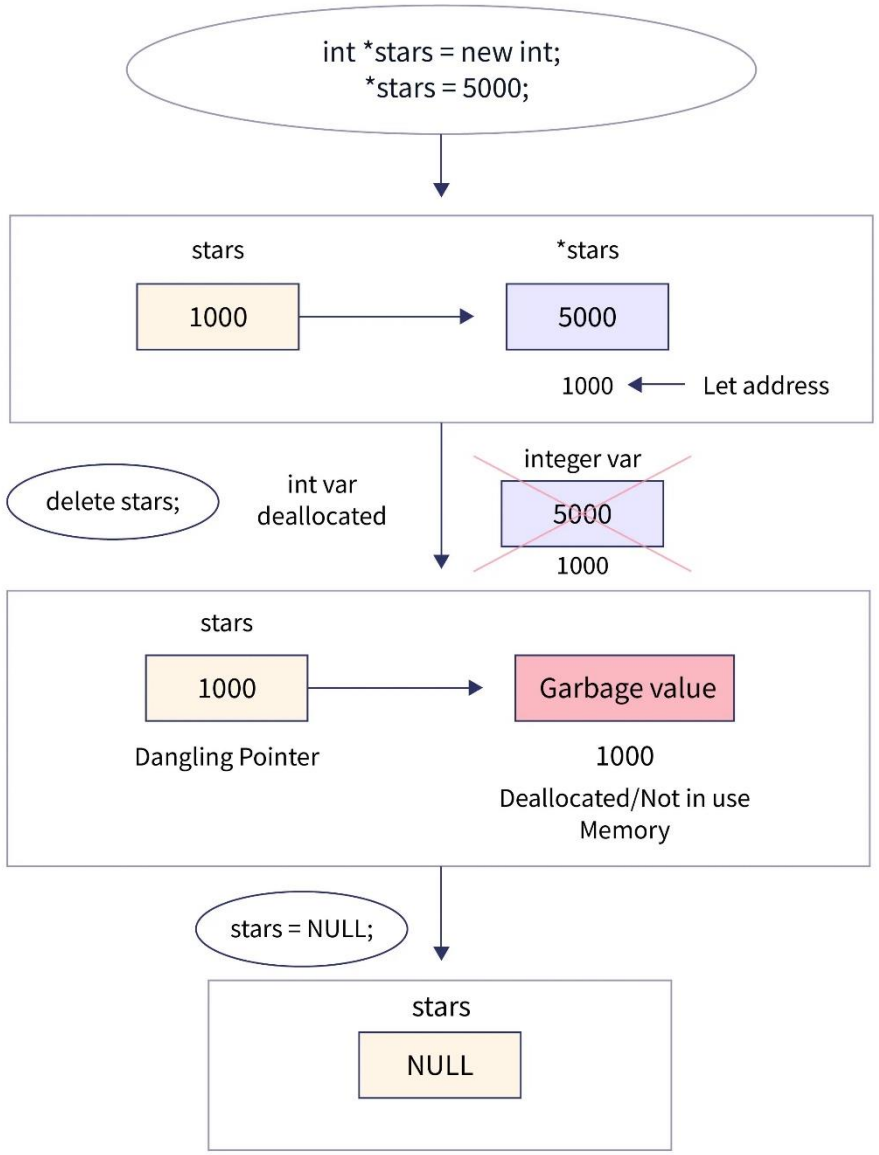
OUTPUT

```
Stars in the sky = 5000
Garbage value = 1234688
```

By – Mohammad Imran

Dynamic Memory De-allocation

Program - 3



By – Mohammad Imran

Dynamic Memory Allocation

Array

The **new** operator can also be used to allocate a block of memory (an array) of any **data_type**.

Syntax

```
data_type* ptr_var = new data_type[size_of_the_array];
```

Example

```
char* name = new char[10];
```

Dynamic Memory Allocation

Array

Program - 4

```
#include <iostream>
using namespace std;
int main()
{
   int* arr = new int[5];
   for(int i = 0; i < 5; i++)
   {
      cout << "Enter array element: ";
      cin >> arr[i];
   }
   cout << "\nArray elements: ";
   for(int i = 0; i < 5; i++)
   {
      cout << arr[i] << " ";
   }
```

By – Mohammad Imran

Dynamic Memory Allocation

Array

Program - 4

```
   delete [] arr;
   cout << "\nGarbage memory:";
   cout << endl;;
   for(int i = 0; i < 5; i++)
   {
      cout << arr[i] << endl;
   }
   return 0;
}
```

OUTPUT

```
Enter array element: 1
Enter array element: 2
Enter array element: 3
Enter array element: 4
Enter array element: 5
Array elements: 1 2 3 4 5

Garbage memory:
11277680
0
11272528
0
5
```

By – Mohammad Imran

What happened if enough memory is not available at runtime

If enough memory is not available in the heap to allocate, the new request indicates failure by throwing an exception of type **std::bad_alloc**, unless **"nothrow"** is used with the new operator, in which case it returns a NULL pointer. Therefore, it may be a good idea to check for the pointer variable produced by the new before using its program.

Dynamic Memory Allocation

nothrow | **Program - 5**

```
#include <iostream>
using namespace std;
int main ()
{
   int* p = NULL;
   p = new(nothrow) int;
   if (!p)
      cout << "allocation of memory failed\n";
   else
   {
      *p = 29;
      cout << "Value of p: " << *p << endl;
   }
   delete p;
   return 0;
}
```

OUTPUT

```
Value of p = 29
```

By – Mohammad Imran

Dynamic Memory Management

Pointer and Classes

A pointer to a C++ class is done exactly the same way as a pointer to a structure and to access members of a pointer to a class you use the member access operator **->** operator, just as you do with pointers to structures. Also as with all pointers, you must initialize the pointer before using it.

By – Mohammad Imran

Pointer and Classes

We can define pointer of class type, which can be used to point to class objects.

Syntax:

```
className *objectName;
```

Pointer and Classes

Program - 6

```
#include <iostream>
using namespace std;
class Pointer
{
   int num;
  public:
   void set_number(int value)
   {
      num = value;
   }
   void show_number();
};
void Pointer::show_number()
{
      cout << num << "\n";
}
```

```
int main()
{
   Pointer object, *p;
   object.set_number(1);
   object.show_number();
   p = &object;
   p -> show_number();
   return 0;
}
```

OUTPUT

```
1
1
```

By – Mohammad Imran

```
#include <iostream>
using namespace std;
class Data
{
  public:
   int a;
   void print()
   {
      cout << "a is "<< a
      << endl;
   }
};
```

```
int main()
{
   Data d, *dp;
   dp = &d;
   int Data :: *ptr = &Data :: a;
   d.*ptr = 10;
   d.print();
   dp -> *ptr = 20;
   dp -> print();
   return 0;
}
```

OUTPUT

```
a is 10
a is 20
```

By – Mohammad Imran

Dynamic Memory Allocation

this Keyword

In C++ programming, **this** is a keyword that refers to the current instance of the class. There can be 3 main usage of this keyword in C++.

- ✓ It can be used **to refer current class instance variable.**
- ✓ It can be used **to pass current object as a parameter to another method.**
- ✓ It can be used **to declare indexers.**

By – Mohammad Imran

this Keyword

Example - 8

```
#include <iostream>
using namespace std;
class This
{
   int val;
   public:
   This(int val)
   {
      this -> val = val;
   }
   void display()
   {
      cout << "Val = " << val << endl;
   }
};
```

OUTPUT

```
Val = 100
```

```
int main()
{
   This th(100);
   th.display();
   return 0;
}
```

By – Mohammad Imran

Memory Leak

Memory leakage occurs in C++ when programmers allocates memory by using new keyword and forgets to de-allocate the memory by using delete() function or delete[] operator. One of the most memory leakage occurs in C++ by using wrong delete operator.

The delete operator should be used to free a single allocated memory space, whereas the delete [] operator should be used to free an array of data values.

By – Mohammad Imran

Memory Leak

Disadvantages

If a program has memory leaks, then its memory usage is satirically increasing since all systems have limited amount of memory and memory is costly. Hence it will create problems.

By – Mohammad Imran

Memory Leak

Program - 6

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

void func_to_show_mem_leak()
{
   int* ptr = new int(5);
   cout << *ptr;
   return;
}
int main()
{
   func_to_show_mem_leak();
   return 0;
}
```

OUTPUT

```
5
```

By – Mohammad Imran

Memory Leak

How to Avoid

- ✓ Instead of managing memory manually, try to use smart pointers where applicable.
- ✓ use std::string instead of char *. The std::string class handles all memory management internally, and it’s fast and well-optimized.
- ✓ Never use a raw pointer unless it’s to interface with an older lib.

Memory Leak

How to Avoid

- ✓ The best way to avoid memory leaks in C++ is to have as few new/delete calls at the program level as possible – ideally NONE. Anything that requires dynamic memory should be buried inside an RAII object that releases the memory when it goes out of scope. RAII allocate memory in constructor and release it in destructor, so that memory is guaranteed to be de-allocated when the variable leave the current scope.
- ✓ Allocate memory by new keyword and de-allocate memory by delete keyword and write all code between them.

By – Mohammad Imran

Memory Leak

Program - 7

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

void func_to_show_mem_leak()
{
   int* ptr = new int(5);
   cout << *ptr;
   delete ptr;
}
int main()
{
   func_to_show_mem_leak();
   return 0;
}
```

OUTPUT

5

By – Mohammad Imran

Coding Questions

By – Mohammad Imran

Question - 1

Write a C++ program to dynamically allocate an integer, a character and a string and assign a value to them.

Click here to see code

By – Mohammad Imran

Question - 2

Write a C++ program to dynamically allocate an array of integers and strings and initialize its elements.

Click here to see code

By – Mohammad Imran

Question - 3

Write a C++ program to dynamically allocate two two-dimensional arrays of floating values. Initialize its elements and display it.

Click here to see code

By – Mohammad Imran

Question - 4

Write a C++ program to dynamically allocate two two-dimensional arrays of string values. Initialize its elements and display it.

Click here to see code

By – Mohammad Imran

Question - 5

Write a C++ program to dynamically allocate memory for a character. Input a character from the user.

Click here to see code

By – Mohammad Imran

Question - 6

Write a C++ program to dynamically allocate memory to a structure of two member integer and string. Initialize value and display it.

Click here to see code

By – Mohammad Imran

Question - 7

Write a C++ program to create an array and display the element of an array using pointer.

Click here to see code

By – Mohammad Imran

QUIZ

By – Mohammad Imran

Quiz - 1

```
#include<iostream>
using namespace std;
int main()
{
    int *p;
    p = new int(20);
    cout << sizeof(p);
    return 0;
}
```

OUTPUT

8

Click here to see code

By – Mohammad Imran

Quiz - 2

```
#include<iostream>
using namespace std;
int main()
{
    int arr[] = {4, 5, 6, 7};
    int *p = (arr + 1);
    cout << arr;
    return 0;
}
```

OUTPUT

Address

Click here to see code

By – Mohammad Imran