Dynamic Memory Allocation in C++

Lesson Plan

Date: [Insert Date]

Instructor: [Insert Instructor Name]

Objectives

- - Understand and use dynamic memory allocation in C++.
- - Dynamically allocate and deallocate single variables and arrays.
- - Manage memory manually in classes.

Introduction to Dynamic Memory Allocation

- - Static vs. dynamic memory allocation.
- - Heap and stack.

- Static Memory Allocation:
- - Memory size determined at compile time.

- Dynamic Memory Allocation:
- - Memory allocated at runtime.

Single Variables Example

- int* p = new int;
- *p = 10;
- std::cout << *p; // Outputs 10
- delete p;

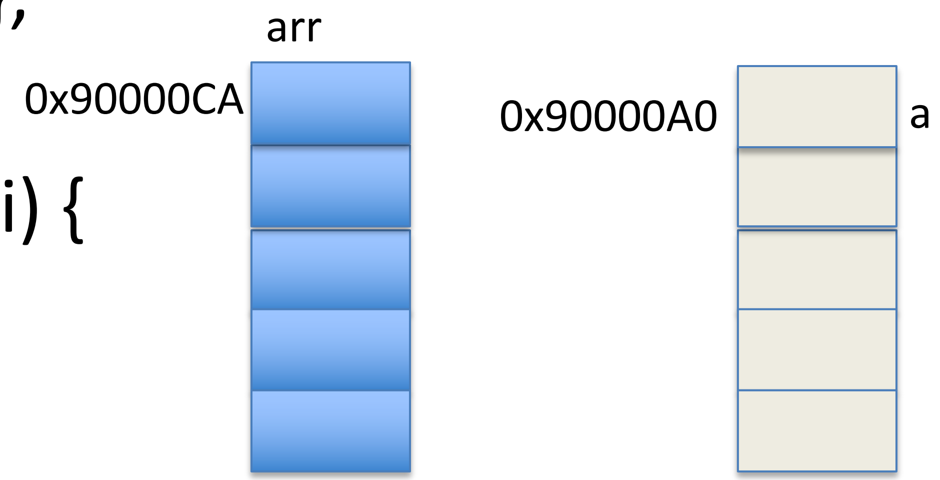
Allocating and Deallocating Single Variables

- - Use 'new' and 'delete' for single variables.
- - Importance of 'delete' to avoid memory leaks.

- Example:
- int* p = new int(5);
- std::cout << *p; // Outputs 5
- delete p;

Arrays Example

- int a[5] = {1,2, 3, 4,5};
- int* arr = new int[5];
- for (int i = 0; i < 5; ++i) {
-     arr[i] = i * 2;
- }
- for (int i = 0; i < 5; ++i) {
-     std::cout << arr[i] << ' ';
- }
- delete[] arr;



Dynamic Memory Allocation for Arrays

- - Allocating and deallocating arrays using 'new[]' and 'delete[]'.
- - Accessing elements in dynamically allocated arrays.

- Key Points:
- - Use 'new[]' to allocate an array.
- - Use 'delete[]' to deallocate an array.
- - Avoid using 'delete' for arrays and 'delete[]' for single variables.

Managing Dynamic Memory in Classes

- - Using dynamic memory within classes.
- - Implement constructors, destructors, copy constructors, and assignment operators to manage dynamic memory.

- Example:

```
class DynamicArray {
private:
    int* data;
    int size;
public:
    DynamicArray(int s) : size(s) {
        data = new int[size];
    }
    ~DynamicArray() {
        delete[] data;
    }
};
```

Class Example

- int main() {
- DynamicArray arr(10);
- // Use arr
- return 0; // destructor will be called automatically
- }

Hands-on Exercise

- Task:
- - Dynamically allocate a single variable and modify it.
- - Dynamically allocate an array, read values, and print them.
- - Implement a class that dynamically allocates an array and manages memory.

Code Example for Exercise

```
class DynamicArray {
private:
    int* data;
    int size;
public:
    DynamicArray(int s) : size(s) {
        data = new int[size];
        for (int i = 0; i < size; ++i) {
            data[i] = i;
        }
    }
    ~DynamicArray() {
        delete[] data;
    }
};

int main() {
    int* p = new int(5);
    std::cout << 'Single variable: ' << *p << std::endl;
    delete p;

    int* arr = new int[5];
    for (int i = 0; i < 5; ++i) {
        arr[i] = i * 2;
    }
    std::cout << 'Dynamic array: ';
    for (int i = 0; i < 5; ++i) {
        std::cout << arr[i] << ' ';
    }
    std::cout << std::endl;
    delete[] arr;

    DynamicArray dArr(10);
    std::cout << 'Class-managed dynamic array: ';
    dArr.print();

    return 0;
}
```

Review and Q&A

- - Recap key points:
- - Dynamic memory allocation for single variables and arrays.
- - Managing memory in classes.
- - Q&A session.

References

- - 'Programming: Principles and Practice Using C++' by Bjarne Stroustrup
- - 'C++ Primer' by Stanley B. Lippman