22 June 2016

Interview Preparation

Lecture: 4- Dynamic Allocation & Additional Problems

Ankush Singla

Question from last class

Doubts from Arrays, Maps & Time Space Complexity

Assignment Status?

Map Class?

Reference Variable

Pass by Reference in C++

Pointer and Reference as return value from function!

Address Typecasting

Dynamic Memory Allocation

Allocating Memory

There are two ways that memory gets allocated for data storage:

- Compile Time (or static) Allocation
  - Memory for named variables is allocated by the compiler
  - Exact size and type of storage must be known at compile time
  - For standard array declarations, this is why the size has to be constant
- Dynamic Memory Allocation
  - Memory allocated "on the fly" during run time
  - dynamically allocated space usually placed in a program segment known as the heap or the free store
  - Exact amount of space or number of items does not have to be known by the compiler in advance.
  - For dynamic memory allocation, pointers are crucial

Dynamic Memory Allocation

- We can dynamically allocate space while the program is running but we cannot create new variable names “on the fly”
- For this reason, dynamic allocation requires two steps
  1. Creating the dynamic space
  2. Storing its address in a pointer
- To dynamically allocate memory in C++, we use new operator
- De-allocation:
  - De-allocation is the “clean-up” of space being used by variable

De-allocation

- De-allocation is the “clean up” of space being used by variables or other data storage
- Compile time variable are automatically deallocated based on their know scope
- It is the programmer’s job to deallocate dynamically created memory
- To de-allocate dynamic memory we use delete operator

new operator

- To allocate space dynamically, use the unary operator new, followed by the type being allocated.
  - new int; // dynamically allocates an int
  - new double; // dynamically allocates a double
- If creating an array dynamically, use the same form, but put brackets with a size after the type:
  - new int[40]; /allocates an array of 40 ints
  - new double[size]; // allocates an array of size double
    // doubles
- These statements above are not very useful by themselves, because allocation space have no names.

new operator contd..

```
int * p;        // declare a pointer p
 p = new int;    // dynamically allocate an int and load address into p

 double * d;     // declare a pointer d
 d = new double; // dynamically allocate a double and load address into d

 // we can also do these in single line statements
 int x = 40;
 int * list = new int[x];
 float * numbers = new float[x+10];
```

delete operator

- To de-allocate memory that was created with new, we use the unary operator delete. The one operand should be a pointer that stores the address of the space to be deallocated:

```
int * ptr = new int;       // dynamically created int
// ...
delete ptr;                // deletes the space that ptr points to
```

Note that the pointer ptr still exists in this example. That's a named variable subject to scope and extent determined at compile time. It can be reused:

- To deallocate a dynamic array, use this form:

```
int * list = new int[40];   // dynamic array

delete [] list;             // deallocates the array
list = 0;                                  // reset list to null pointer
```

After deallocating space, it's always a good idea to reset the pointer to null unless you are pointing it at another valid target right away.

Lets look at some input output problems

#define

Inline Functions?

Default Value of Arguments?

Compute Time Complexity

```
while(n){
        j=n;
        while(j>1){
                j-=1;
        }
        n/=2;
}
```

Compute Time Complexity

```
while(n){
        j=n;
        while(j>1){
                j-=n/j;
        }
        n/=2;
}
```

Doubts from Recursion

Backtracking?

Rat in a Maze

N queens

Homework

Sodoku

Thank you

Ankush Singla
ankush@codingninjas.in