Lecture 6 – Dynamic Arrays

CST238 – Intro to Data Structures
YoungJoon Byun
ITCD

1

Lecture Objectives

- After completion of this lecture, you will be able to

2

Chapter 3: Data Structures and Abstract Data Types

- 3.1 Data Structures, Abstract Data Types and Implementations
- 3.2 Static Arrays
- 3.3 Multidimensional Arrays
- **3.4 Dynamic Arrays**
- 3.5 C-Style Structs
- 3.6 Procedural Programming

3

A Problem of a Static Array

- Array size can't be changed during the program execution.

- Common errors
  1. cin >> n;
     double a[n]; // error! *n* must be a constant.
  2. int n =10;
     double a[n]; // Again, it's an error.
     // It should be "const int n =10;"

4

Dynamic Array

- Dynamic allocation of memory for an array with **`new`** and **`delete`**.
  - Acquire memory as needed.
  - Release memory when no longer needed.

5

Dynamic Allocation with **new**

```
type * ptr = new type [capacity];
```

// It requests a block of memory dynamically.
// Then, the starting address is assigned to the pointer

- Example

```
int n;
int * arrayPtr;
cin >> n;   // e.g., n has 6.
arrayPtr = new int[n];
```

6

Dynamic Array Access

- How can we access an element of the dynamically allocated array?
- Example

```
int * arrayPtr = new int[5];
arrayPtr[0] = 10;
arrayPtr[1] = 20;
…
arrayPtr[4] = 40;
```

7

Release the dynamic memory with **`delete`**

- Counterpart to the **`new`** operation
  - The allocated memory is returned to the heap.
  - Then, the area can be reused later.
- Syntax

```
  delete [ ] arrayPointerVariable;
// Example: delete [ ] ptr;
```

8

`delete` Operation

- Example

(1) int * ptrArray = new int [size];
    delete [ ] ptrArray;

(2) int * ptrVar = new int;
    delete ptrVar;

9

Memory Leaks

- A sample code: anything wrong?

```
int * intPtr = new int [10000];
…
intPtr = new int [10];
```

10

Accessing an element of an array

- Array index
  - You can access an array element through the square bracket [ ].
  - e.g., int records[100];
    records[3] = 75;
    records[4] = records[3];
- Access through a pointer (or an address)
  - e.g., int my_score = *(records + 3);

11

Array access using a pointer

- An array name is the “base address” of the whole array.
  - In other words, an array name has the address of the first element of array.
- Example

int a[20]; // Here, **a** is the same as &a[0]
int * p = a; // same as p= &a[0]
What happens for p = p + 1; // or p++ ?

12

Pointer Arithmetic: **p++**

13

Pointer Arithmetic Operation

- Pointer to an array can do + and – an integer operations such as
  - **p = p + i;** and **p = p - i;**
  - where **p** is a pointer to an array and **i** is an integer.
- Example

```
int a[20]
int * p = a; // *(p+i) is the same as a[i] or p[i].
++p; p+=2;
p = & a[10];
p = p – 5;
```

14

Sample Program

```
1. int a[100], i, * p, sum = 0;
2. //suppose array a is initialized here.

3. for (i = 0; i < 100; i++) {
4.     sum += *(a + i);
5. }

6. p = a;
7. for (i = 0; i < 100; i++) {
8.     sum += p[i];
9. }

10.p = a;
11.for (i = 0; i < 100; i++) {
12.    sum += *(p + i);
13.}
```

15

Pointer vs. Array Name

- One difference
  - Array name is the address of the first element.
    - So, it is fixed and constant.
  - A pointer is a variable.
    - So, it can hold different addresses.
- Example

16

Example – An array name is a constant.

17

Summary

- Dynamic arrays (chap. 3.4)
  - new and delete operators for arrays
  - A pointer variable and an array name
- Next Lecture
  - Dynamic arrays (contd.)
  - C-Style structs (chap. 3.5)
  - Classes (chap. 4)

18

References

- Larry Nyhoff, *ADTs, Data Structures, and Problem Solving with C++*, 2nd Edition, Prentice-Hall, 2005
- Walter Savitch, *Problem Solving with C++*, 6th Edition, Addison-Wesley, 2006
- Dr. Meng Su's Lecture Notes
  http://cs.bd.psu.edu/~mus11/122Fa06/cse122Fa06.htm

19