Lecture 3 – Pointers

CST238 – Intro to Data Structures
YoungJoon Byun
ITCD

Lecture Objectives

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

2

Chapter 2: Introduction to Abstract Data Types

- 2.1 A first look at ADTs and Implementations
- 2.2 C++'s Simple Data Types
- 2.3 Programmer-Defined Data Types
- **2.4 Pointers**

3

Address of a Variable

- When a variable is declared
  - Some memory location is allocated to store a value of the specified type.
  - The variable name is actually associated with that memory location.
  - The memory location is initialized with a value, if it is provided.

4

Example

5

A Pointer Variable

- A variable that can hold a memory address.
  - Often it is just called a pointer.
- Forms

  type * pointerVariable;

  type * pointerVariable = address;
- Example

```
int * intPtr;
```

6

Memory Layout with a Pointer

7

Exercise

- Declare a pointer variable called **`ptr1`** that can hold the address of a double variable.
- Declare a pointer variable called **`ptr2`** that can hold the address of a character variable.

8

Address Operator (**&**)

- **&** operator can determine the address of a variable
- Example

```
 int intVar1 = 100;
int * p;
p = & intVar1;
```

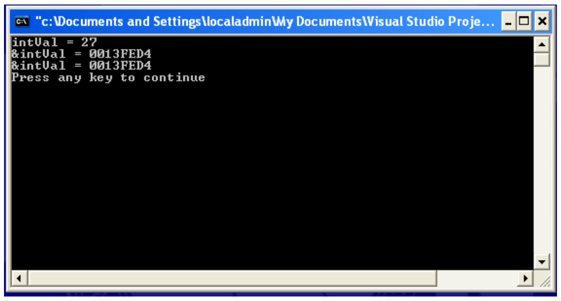
9

Sample Program

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

int main()
{
    int intVal = 27;
    int * intPtr;
    intPtr = &intVal;
    cout << "intVal = " << intVal << endl;
    cout << "&intVal = " << intPtr << endl;
    cout << "&intVal = " << &intVal << endl;
}
```

Execution Result


"c:\Documents and Settings\localadmin\My Documents\Visual Studio Proje...

```
intVal = 27
&intVal = 0013FED4
&intVal = 0013FED4
Press any key to continue
```

Anything wrong in this program?

```
int main()
{
    int intVar = 27;
    double doubleVar = 100.5;

    int * intPtr;
    double * doublePtr;

    intPtr = &intVar;
    doublePtr = &intVar;

    cout << "&intVar = " << intPtr << endl;
    cout << "&doubleVar = " << doublePtr << endl;
}
```

Declaration of Multiple Pointers

- To declare multiple pointers in a statement, use the asterisk before each pointer variable
- Example

```
int * p1, * p2, v1, v2;
```

- Very common mistake

```
int * p1, p2;
```

13

Dereferencing Operator (*)

- A pointer variable stores address of a memory location (= variable)
  - Accessing contents of that location requires dereferencing operator *
- Example

```
int intVar = 100;
int * iPtr;
iPtr = &intVar;
int anotherInt = *iPtr;
```

14

Exercise: Result of Execution

```
1. int v1;
2. int * p1;
3. v1 = 0;
4. p1 = &v1;
5. *p1 = 42;
6. cout << v1 << endl;
7. cout << *p1 << endl;
```

15

Pointer Assignment

- Suppose another pointer p2 of the previous exercise.

  int * p2;

- What happens after the pointer assignment?

  p2 = p1;

  – The assignment operator = is used to assign the value of one pointer to another pointer.

16

Exercise: Draw memory layout and determine the execution result.

1. int v1 = 84;
2. int v2 = 99;
3. int * p1, * p2;
4. p1 = &v1;
5. p2 = &v2;
6. p1 = p2;
7. cout << *p1 << “ ” << v1 << endl;

17

Exercise: Determine execution result.

1. int v1 = 84;
2. int v2 = 99;
3. int * p1, * p2;
4. p1 = &v1;
5. p2 = &v2;
6. *p1 = *p2;
7. cout << *p1 << " " << v1 << endl;

18

Summary

- Pointers (Chap. 2.4)
  – Address of a memory location (= variable)
  – Pointer operations (* and &)
- Next Lecture
  – Dynamic memory allocation (Chap. 2.4)

19

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

20