Object Oriented Programming

Lecture 2

C++ Pointers
Spring 2024

Lecture Contents

▪ Introduction to Pointers ▪

Pointer variable declaration ▪

Pointer initialization ▪

Referencing

- Dereferencing

- Arithmetic on Pointers

Regular Variables vs Pointers

- Variables are used to keep track of data in the computer's memory.
- Declaring a regular variable = Allocating a location in memory to store a value

- **Example:**

int myInt = 0; //Allocates enough space in memory

 //to store an int and puts 0 there

▪Every location in memory has an address

▪ Use *address-of* operator **'&'** to get the address ▪ **Example:**

▪ cout << "Address of myInt = " << &myInt << endl;

Regular Variables vs. Pointer Variables

▪ **A pointer variable is used to hold an address** •

Use operator * to declare a pointer variable

- Example:

```
int *pMyInt; //Declares a pointer
```

- An address must be assigned to the pointer variable before it can be used

```
int myInt = 0; //Allocates memory, stores 0 int

*pMyInt; //Declares an empty pointer pMyInt =

&myInt; //Puts an address in the pointer
```

Pointer Variable

- The main difference between regular variables and pointers is that, variables contain a value, while pointers contain a memory address.

- **Memory address:**
  - We know each memory address contain some value.
  - Thus if pointers contain memory addresses, we can get its value (indirectly).

3. Content at the memory address 174 is 7

2. myInt's memory address is 174

myInt

... 7 3 4 ...

PmyInt

Memory

1. myInt is a variable

172 173 174 175 176 177 178 179 180 181

PmyInt is a Pointer that holds the address of myInt

... 174 3 4 ... 832 833 834 835 836 837 838 839 840 841

Dereferencing a Pointer

- Accessing the object addressed by the pointer
- Dereference Operator **'*'** used with the name of the pointer, ***after*** it is declared and initialized

- Examples

```
int myInt = 0; //Allocates memory, stores 0
int *pMyInt; //Declares an empty pointer
pMyInt = &myInt; //Puts address in the pointer
cout << *pMyInt << endl; //Prints 0
*pMyInt = 5; //puts 5 into myInt cout <<
myInt << endl; //Prints 5
cout << *pMyInt << endl; //Also prints 5
cout << pMyInt << endl; //What prints? address of the pMyInt
```

Where are the errors?

```
int main(){

 int m;
 int *pm;
 *pm = 5;

 int n;
 int *pn = &n;
pn = 5;

}
```

Where are the errors?

```
int main(){

 int m;
 int *pm;
 *pm = 5;

 int n;
```

```
 int *pn = &n;
pn = 5;

}
```

ERROR! No address in pm

```
//Correction
```

```
pm = &m;                 *pm = 5;
```

Where are the errors?

```
int main(){
                    int *pm;
 int m;             *pm = 5;
 int n;
 int *pn = &n;
pn = 5;

}
```

ERROR! No address in pm

```
//Correction
pm = &m;
*pm = 5;
```

ERROR! Missing operator*

```
//Correction
*pn = 5;
```

Dereferencing—Another Example

```
int C;
int *P; /* Declare P as a pointer to int */
```

```
C = 7;
P = &C;
cout<<*p<<endl; //What is the output?
```

C

... 7 3 4 ...

172 173 174 175 176 177 178 179 180 181

P

... 174 3 4 ...

832 833 834 835 836 837 838 839 840 841

Dereferencing

cout << *P; /* Prints out ‘7’ */

```c
*P = 177;
P = 177; /* This is unadvisable!! */
```

```
                C
       ... 7 3 4 ...
                177
   172 173 174 175 176 177 178 179 180 181

                                 P
                         ... 174 3 4 ... 177
                    832 833 834 835 836 837 838 839 840 841
```

Pointers & Allocation (1/2)

- After declaring a pointer:

int *ptr1;

ptr1 doesn't actually point to anything yet. So its address is NULL.

. We can either:

- make it point to something that already exists, • ptr1 = &C
- or
- allocate room in memory for something new that it will point to… (dynamic memory will discuss later)

ptr1 = NULLMemory

Pointers & Allocation(2/2)

- Pointing to something that already exists:
  - int *ptr, var1, var2;
  - var1 = 5;
  - ptr = &var1;

- `var2 = *ptr; //Dereferencing using the * operator`
- **`var1`** and **`var2`** have room implicitly allocated for them.

```
                ?
ptr  var1           var2  ?  5  ?5
```

Arithmetic on Pointers (1/4)

- A pointer may be incremented or decremented.
- This means only address that the pointer holds is incremented or decremented.

▪ An integer may be added to or subtracted from a pointer. ▪ Pointer variables may be subtracted from one another. ▪ Pointer variables can be used in comparisons, but usually only in a comparison to pointer variables or NULL.

Arithmetic on Pointers (2/4)

▪ When an integer **(n)** is added to or subtracted from a pointer **(ptr)** ▪ The new pointer value **(ptr)** is changed by the **ptr** address **plus (+) n** **multiple (*)** the (bytes of **ptr** data type).

▪ **ptr** + **n** * (bytes of **ptr** data type)

▪ Example 1

```
#include <iostream>int main(){
int x;
int* ptr = &x ; // assume 4 byte ints
cout << ptr << '\n';
++ptr; // ptr = ptr + 1
cout << ptr << '\n’;
--ptr; // ptr = ptr - 1
cout << ptr << '\n’;
return 0;
}
```

Output:

```
00AFFD70 00AFFD74 00AFFD70
```

Arithmetic on Pointers (3/4)

Example

```cpp
/* Test pointer declaration and initialization (TestPointerInit.cpp) */
#include <iostream>
using namespace std;

int main() {
   int number = 88;    // Declare an int variable and assign an initial value
   int * pNumber;      // Declare a pointer variable pointing to an int (or int pointer)
   pNumber = &number;  // assign the address of the variable number to pointer pNumber

   cout << pNumber << endl;  // Print content of pNumber (0x22ccf0)
   cout << &number << endl;  // Print address of number (0x22ccf0)
   cout << *pNumber << endl; // Print value pointed to by pNumber (88)
   cout << number << endl;   // Print value of number (88)

   *pNumber = 99;            // Re-assign value pointed to by pNumber
   cout << pNumber << endl;  // Print content of pNumber (0x22ccf0)
   cout << &number << endl;  // Print address of number (0x22ccf0)
   cout << *pNumber << endl; // Print value pointed to by pNumber (99)
   cout << number << endl;   // Print value of number (99)
                             // The value of number changes via pointer

   cout << &pNumber << endl; // Print the address of pointer variable pNumber (0x22ccec)
```

Arithmetic on Pointers ▪ {
(4/4)

▪Example

▪ void main (void)

▪ int *pointer1, *pointer2;

▪ int num1 = 93;

▪ pointer1 = &num1; //address of num1 ▪ pointer2 = pointer1; // pointer1 address is assigned to pointer2

▪ cout<<"Address stored

Address data 1000

in pointer1 "<<pointer1; num1 = 93

▪ cout<<"Address stored in pointer2 "<<pointer2;

pointer1 = 1004 pointer2 =

1004

▪ }

1004 1008 1012 1016

Logical operators on Pointers (<,>,== etc.)

- Example

```
 int *pointer1, *pointer2;
 int num1 = 93;
 If ( pointer1 == NULL ) // pointer compared to
NULL  pointer1 = &num1;
pointer2 = &num1;
 If ( pointer1 == pointer2 ) // pointer compared to
pointer
   cout<<“Both pointers are equal\n”;
```