Lecture 2.3

Pointers

Introduction to pointers

- Pointers are **variables that hold addresses** in C and C++.
- Provide much power and utility for the programmer to access and manipulate data.
- Useful for passing parameters into functions, allowing a function to modify and return values to a calling routine.
- Used in dynamic memory allocation

Variables storage

- How values for variables are stored in computer memory?
- The computer memory could be represented like this:

Address	
0x241FF5C	
0x241FF58	
0x241FF54	

- Addresses are always represented in hexadecimal format.

Hexadecimal numbers

- Everyday we use base 10 to represent numbers.

  0 1 2 3 4 5 6 7 8 9
- Computers use base 2 (binary) to store information

  0 1
- For memory addresses base 16 is used for the convenience of representing bigger numbers with less digits.

  0 1 2 3 4 5 6 7 8 9 A B C D E F

Hexadecimal numbers (cont.)

- F = 15 in decimal and is represented as 1111 in binary (1*8+1*4+1*2+1*1)
- Therefore, one digit is used to represent 4 bits.
- This makes it easy to understand which bits are 1 and which ones are 0 for a number like 0FF00 (What will this number be in decimal?)

Hexadecimal numbers (cont.)

- With two digits in decimal, the maximum number you can represent is 99
- In hexadecimal this would be FF=255 (F*16+F*1)
- With four digits 9999 or FFFF=65535 (F*4096+F*256+F*16+F*1)

How are values for variables stored in memory

- As a program is executing, all variables are stored in memory.
- Each at its own unique address or location.
- Typically a **variable** and its associated memory address contain **data values**.

Values in memory

- For example, when you declare

  int count = 5;

  The value "5" is stored in memory and can be accessed by using the variable "count" Address

0x241FF5C	5	count
0x241FF58		
0x241FF54		

Manipulating memory address

- To manipulate the memory address of a variable, we use the unary operator **&**.
- So, if you do:

  *cout << **&**count << endl;*

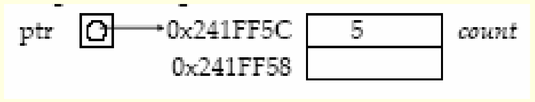
  This should print the address of the variable "count", which for this example is 0x241FF5C.

Pointers and memory address

- A pointer is a special type of **variable** that **contains a memory address** rather than a data value.
- Data is modified when a normal variable is used.
- The value of the address stored in a pointer is modified as a pointer variable is manipulated.

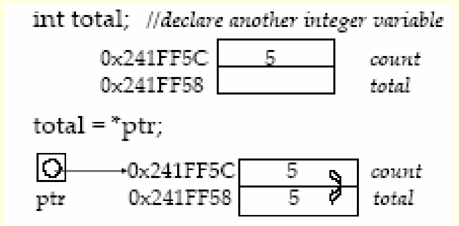
Pointer representation

- Usually, the address stored in the pointer is the address of some other variable.
- int *ptr; //declares a pointer to an integer //ptr
- ptr = &count; //stores the address of count in ptr
- The pointer "points to" count.



Pointers dereferencing

- To get the **value** that is stored at the memory location in the pointer.
- Dereferencing is done with the unary operator "*".



```
int total; //declare another integer variable
```

```
total = *ptr;
```

Declaration and Initialization examples

- Declaring and initializing a pointer is really easy:

```
int j=1;
int k=2;
int *pt1; //pointer to integer
int *pt2; //pointer to integer
pt1 = &j; // pt1 points to j
pt2 = &k; //pt2 contains the address of k
```

Declaration and initialization examples

```
float values[100];
float resluts[100];
float *pt3; //declares pointer to float
float *pt4; //declares a float pointer
pt3 = values;      //pt3 contains the address
                   //of the first element of values
pt3 = &values[0]   //This is the equivalent of the
                   //above statement.
                   //pt3 points to values
```