



UNIVERSITY OF MASSACHUSETTS
DARTMOUTH

ECE160: Foundations of Computer Engineering I

Lecture #26 –**Pointers**

Instructor: Dr. Liudong Xing
SENG-213C, lxing@umassd.edu
ECE Dept.



Administrative Issues

- Lab 11 due **5pm, Wednesday, April 12**
- **No Classes on Monday, April 17 (Patriot's Day Holiday); No Lab in the week of April 17**
- Exam #3 on **Friday, April 21**
 - Review session on Wednesday, April 19

Review of Lecture #25

- In C, a string is a variable-length array that is DELIMITED BY THE NULL CHARACTER (\0).
- Four ways to initialize a string

`char month[10] = "March";`

M	a	r	c	h	\0	0	0	0	0
---	---	---	---	---	----	---	---	---	---

`char month[] = "March";`

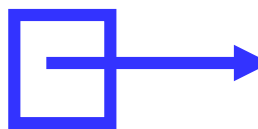
M	a	r	c	h	\0
---	---	---	---	---	----

`char month[6] = {'M', 'a', 'r', 'c', 'h', '\0'};`

M	a	r	c	h	\0
---	---	---	---	---	----

`char *pstr="March";`

--	--	--	--	--	--	--	--	--	--



M	a	r	c	h	\0
---	---	---	---	---	----

Topics

- How to obtain a pointer
- How to use pointers
- Pointers to pointers
- Pointers and functions

Pointers

- A pointer variable can be declared using * in the declaration statement
- How to obtain a pointer to a variable (or the address of a variable)?

Using & operator

```
#include "stdio.h"
void main(void)
{
    int x = 3;
    int *p = &x;
    printf("%d\n", x);
    printf("%d\n", *p);
    printf("%d\n", p);
}
```

What is the output?

Example Explanation

```
#include "stdio.h"
void main(void)
{
    int x=3;
    int *p= &x;
    printf("%d\n",x);
    printf("%d\n",*p);
    printf("%d\n",p);
}
```

& means the address of , a copy of the address of variable x is put into memory cell reserved for p

In a C statement, * is an unary operator (when it is not meant to mean multiplication), meaning to go to the address stored in the variable following * and get the value at that address or put a value at that address

Topics Next

- **How to use pointers?**
 - Ways to increment a number
 - Test for equality using pointers
 - Multiple pointers for one variable
 - Number addition using pointers

Exercises (1)

Ways to increment a number

What is the output for the following program?

```
#include "stdio.h"
void main(void)
{
    int x=3;
    int *p=&x;
    ++x;
    *p=*p+1;
    (*p)++;
    printf("x is: %d\n",x);
}
```


Ways to increment a number (cont'd)

- Assume

```
int a=0;  
int *p=&a;
```

we need to add 1 to a:

```
a++;
```

```
++a;
```

```
a=a+1;
```

```
*p=*p+1;
```

```
(*p)++;
```

Note: postfix increment ++ has a higher priority than indirection operator *; () are needed to force the dereference to occur before the addition so that we add to the data variable, not to the pointer!

Exercise (2)

Test for equality using pointers

```
#include "stdio.h"
void main(void)
{
    int x = 10;
    int y = 10;
    int *px = &x;
    int *py = &y;
    /*Fill out the if equal statement using pointers */
    if (?????)
        printf("The numbers are equal\n");
}
```

Exercises (3)

Multiple pointers for one variable

What is the output for the following program?

```
#include "stdio.h"
void main(void)
{
    int x = 7;
    int *p= &x;
    int *q = &x;
    printf("*p is %d\n", *p);
    printf("*q is %d\n", *q);
    printf("p is %d\n", p);
    printf("q is %d\n", q);
}
```

Example: Number addition using pointers

```
#include "stdio.h"
void main(void)
{
    int x;
    int y;
    int result;
    int *px = &x;
    int *py = &y ;
    int *pr = &result;
    printf("Enter the first number x:\n");
    scanf("%d", px);          /* same as scanf("%d",&x); */
    printf("Enter the second number y:\n");
    scanf("%d", py);        /* same as scanf("%d",&y); */
    printf("Add two numbers:\n");
    *pr = *px + *py;        /* same as result = x+y; */
    printf("The result is: %d\n", result);
}
```

Topics

- How to obtain a pointer
- How to use pointers
- **Pointers to pointers**
- Pointers and functions

Pointers to pointers

- So far all the pointers point directly to data
- It is possible to use pointers that point to other pointers

Example

```
#include "stdio.h"

void main(void)
{
    int x = 10;
    int *p;      /*p is a pointer to an integer*/
    p = &x;
    int **q;    /*q is a pointer to an integer pointer*/
    q = &p;
    printf("%d\n", x);
    printf("%d\n", *p);
    printf("%d\n", **q);
}
```

Output?

To refer to x using q, you have to dereference it twice to get to the integer x because there are two levels of indirection / pointers involved!

Topics

- How to obtain a pointer
- How to use pointers
- Pointers to pointers
- **Pointers and functions**

Pointers and Functions

- Pointers can be arguments to a function (pass by reference)
- Pointers can be returned from a function

Example

```
#include "stdio.h"
void swap(int x, int y);
void main(void)
{
    int a=3;
    int b=7;
    swap(a,b);
    printf("%d %d\n", a, b);
}
void swap(int x, int y)
{
    int temp;
    temp=x;
    x=y;
    y=temp;
}
```

```
#include "stdio.h"
void swap(int *x, int *y);
void main(void)
{
    int a=3;
    int b=7;
    swap(&a,&b);
    printf("%d %d\n", a, b);
}
void swap(int *x, int *y)
{
    int temp;
    temp=*x;
    *x=*y;
    *y=temp;
}
```

Output

An unworkable exchange using passing by values

Output:

3 7

Successful exchange using passing by reference

Output:

7 3

Pointers as function arguments (Note!)

- Every time we want a called function (**swap**) to have access to a variable in the calling function (**main**), we send the address of that variable (using **&** in the function call) to the called function and use the indirection operator (*****) to access it – **passing by reference/address**

```
#include "stdafx.h"
void swap(int *x, int *y);
void main(void)
{
    int a=3;
    int b=7;
    swap(&a,&b);
    printf(" %d %d\n ", a, b);
}
void swap(int *x, int *y)
{
    int temp;
    temp=*x;
    *x=*y;
    *y=temp;
}
```

Functions returning pointers

- Pointers can be returned from a function
- When you return a pointer, it must point to data in the calling function
- It's an error to return a pointer to a local variable in the called function because when the function terminates, its memory can be used by other parts of the program!

An Example

To determine the larger of two numbers

```
#include "stdio.h"
int *max(int *pa, int *pb);
void main(void)
{
int a;
int b;
int *pmax = NULL;

printf("Enter first number:\n");
scanf_s("%d", &a);
printf("Enter second number:\n");
scanf_s("%d", &b);

pmax = max(&a, &b);
printf("The maximum is %d\n", *pmax);
}
```

```
int *max(int *pa, int *pb)
{
int larger;

if (*pa > *pb)
    larger = *pa;
else
    larger = *pb;
printf("The larger one is %d\n", larger);
return &larger;
}
```

Are there any
errors/dangers
in this program?

An Example (Correct Program)

To determine the larger of two numbers

```
#include "stdio.h"
int *max(int *pa, int *pb);
void main(void)
{
int a;
int b;
int *pmax = NULL;

printf("Enter first number:\n");
scanf_s("%d", &a);
printf("Enter second number:\n");
scanf_s("%d", &b);

pmax = max(&a, &b);
printf("The maximum is %d\n", *pmax);
}
```

```
int *max(int* pa, int* pb)
{
    if (*pa > *pb)
        return pa;
    else
        return pb;
}
```

Review Questions I

1. Which of the following statement defines and initializes a pointer to the address of an integer variable `x`?

- a) `int *ptr=*x;`
- b) `int &ptr = *x;`
- c) `int *ptr=^x;`
- d) `int *ptr = &x;`
- e) `int &ptr=^x;`

2. Assume `p` is a pointer that points to the variable `a`, which of the following statements will NOT add 1 to the variable `a`?

- a) `a++;`
- b) `a+=1;`
- c) `a=a+1;`
- d) `p=p+1;`
- e) `*p=*p+1;`
- f) `*p++;`

Review Questions II

3. Given the following declarations:

```
int a=5;  
int b=7;  
int *p=&a;  
int *q=&b;  
int *r=&a;
```

what is the value of each of the following expressions?

a) ++a;

b) ++(*p);

c) --(*q);

d) --b;

e) a++;

f) b--;

g) (*r)++;

h) (*q)--;

Summary of Lecture #26

- Using pointers
 - to increment a number
 - to test for equality using pointers
 - to add two numbers
- Use multiple pointers for one variable
- Use pointers that point to other pointers
- Pointers and functions
 - Pointers can be arguments to a function (pass by reference)
 - Pointers can be returned from a function

Things To Do

- Review lecture notes and run & test programs in the exercise

Next Topic

- Pointers and Arrays