C - Pointers

COEN 10

C - Lecture 8

Pointers

★Definition

◎Pointers are variables whose value is a memory address

Pointers

★Declaration

<type> *<name>;

Pointers

★Example

```
int   *pi;
float *pf, *pf2;
char  *pc;
```

◎Variables pi, pf, and pc have the capacity to hold an address

◎Need one * per pointer

Pointers

★NULL

- ◎Constant defined as a null pointer, equivalent to zero
- ◎This is an invalid address

Pointers

★Example

```
int    *pi = NULL;
```

- ◎Variable pi points to address zero, which is invalid

Pointers

★Operators

& -- address operator

* -- indirection operator

★Example

```
int   x, y = 5;
int   *pi;
pi = &x;
*pi = y;
```

Using Pointers to Communicate between Functions

★Function A calls function B

- ◎Passes the address of a variable as argument

★Function B

- ◎Receives the address in a pointer
- ◎Can use the pointer to access the variable in function A

Using Pointers to Communicate between Functions

```
void a (void)
{
    int x = 0;
    b (&x);
    printf ("%d\n", x);
    return;
}

void b (int *p)
{
    *p = 100;
   return;
}
```

Pointers in Assignments

★Without pointers

variable = value of expression

★With pointers

pointer = address of variable

pointed value = value of expression

→ Pointed value can be obtained with a pointer expression

Pointers in Assignments

★Example

```
int    x, y;
int    *pi;
y = 5;       // variable = value of expression
pi = &x;     // pointer = address of variable
*pi = y;     // pointed value = value of expression
```

Pointers and Arithmetic Operations

★Addition and subtraction

◎Move the pointer according to its type

◎Example

```
int    x[3] = {0, 1, 2};
int    *ptr = &x[0];
ptr++;    // points to the next int in memory
Ptr--;    // points to the previous int in memory
```

Pointers and Relational Operations

★Pointers can be compared with pointers and/or with addresses

★All the relational operators can be used

◎==, !=, >, >=, <, <=

Pointers and Arrays

★The name of an array

◎Represents the address of the array

◎Is a constant pointer

★Arrays and pointers can be used interchangeably

Pointers and Arrays

★Example

```
int  x[5] = {0, 1, 2, 3, 4};
int  *p = x;
*p = 10;
p++;
*(p + 1) = 9;
p[0] = 8;
```

Pointers and Arrays

★Pointers can be used to traverse arrays

★Example

```
int  x[5] = {0, 1, 2, 3, 4};
int  *p = x;
for (i = 0; i < 5; i++)
  printf ("%d\n", *p++);
```

Pointers and Strings

★Pointers can be used to traverse strings

★Example

```
char  x[5] = "abc";
char  *p = x;
while (*p++ != '\0')
  counter++;
```

Pointers and Strings

★Example

```
char    str[5] = "abc";
char    *p;
p = str;
*p = 'A';
p++;
*(p + 1) = 'C';
p[0] = 'B';
```

Pointers Operators

★Precedence

1.	++,-- postfix ( )	→ L-R
2.	++,-- prefix +,- (type) ! sizeof * &	→ R-L
3.	* / %	→ L-R
4.	+ -	→ L-R
5.	< > <= >=	→ L-R
6.	== !=	→ L-R
7.	&&	→ L-R
8.	||	→ L-R
9.	:?	→ R-L
10.	= += -= *= /= %=	→ R-L
11.	,	

Pointers

★Important points

◎Need to initialize before using

◎Need to be careful not to write to or read from an illegal address

◎Need to be careful with the range, when traversing arrays and strings