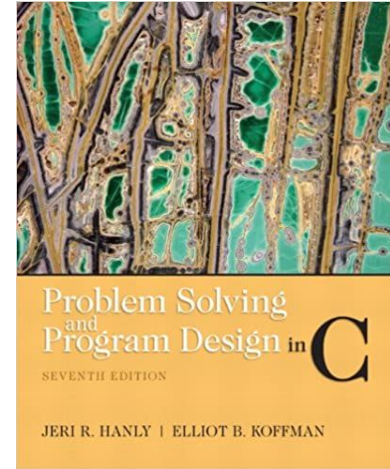


Faculty of Engineering and Technology Department of Computer Science

Introduction to Computers and
Programming (Comp 133)



References :

Book : Problem Solving and Program Design in C (7th Edition) 7th Edition

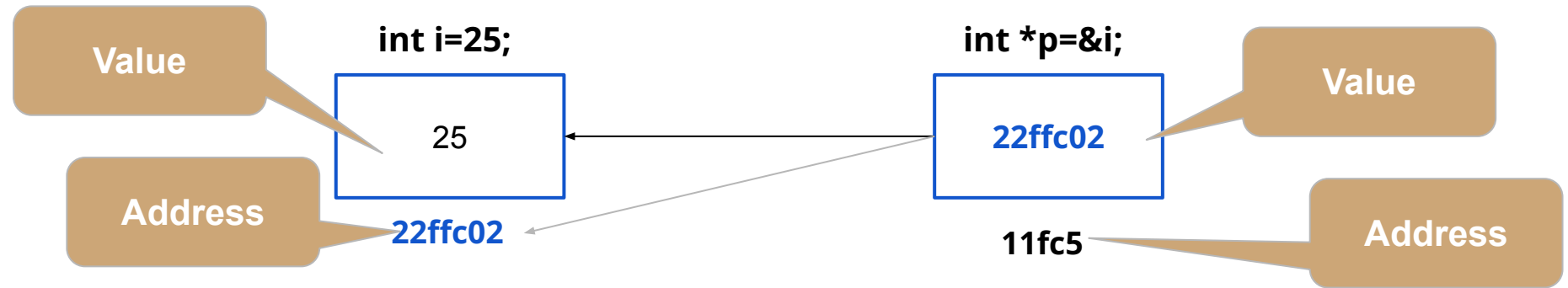
Slides : Dr. Radi Jarrar , Dr. Abdallah Karakra , Dr. Majdi Mafarja.

Pointers and Modular Programming

Chapter 6

Pointer

- Pointer or pointer variable: A memory cell that stores the address of a data item.
- The declaration:
 - **float *p;**
 - Identifies **p** as a pointer variable of type “**pointer to float** .”
 - We can store the memory address of a type float variable in **p**.





Chapter 6

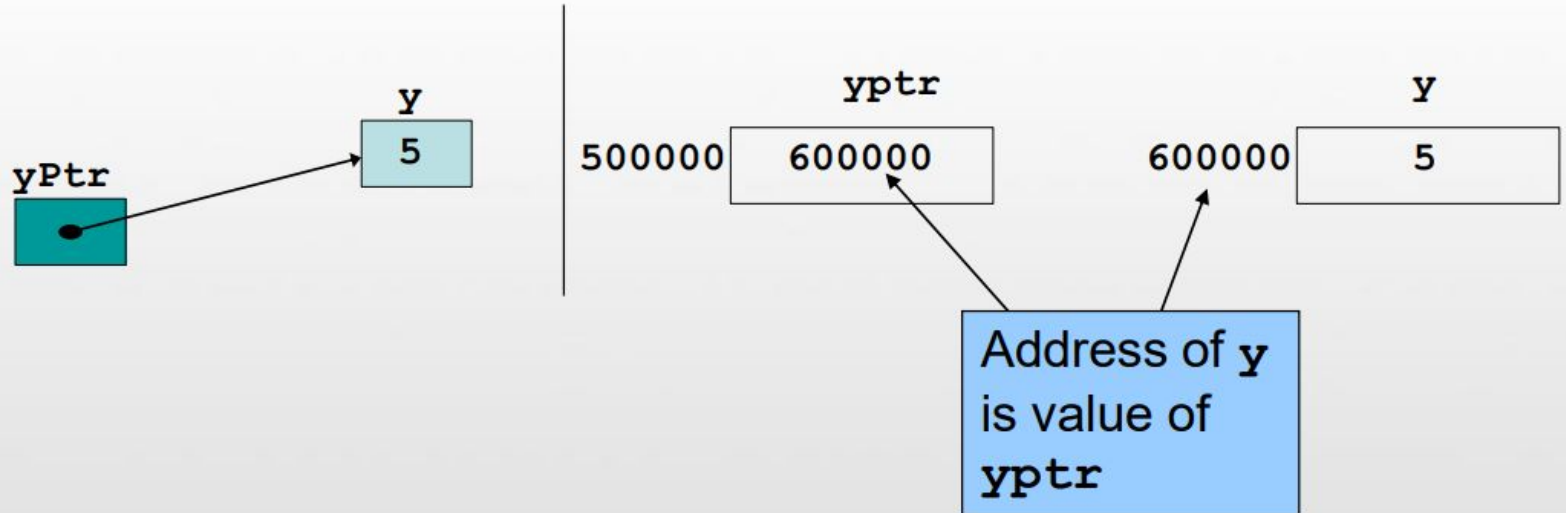
- Pointer variable

Pointer variable

- Syntax : **type** ***ptr_name** ;
- **int *ptr** ; declares a pointer **ptr** to variables of type int.
- **char *ch** ; declares a pointer **ch** to variables of type char.
- **double *dblPtr**; declares a pointer **dblPtr** to variables of type double.

Pointer variable

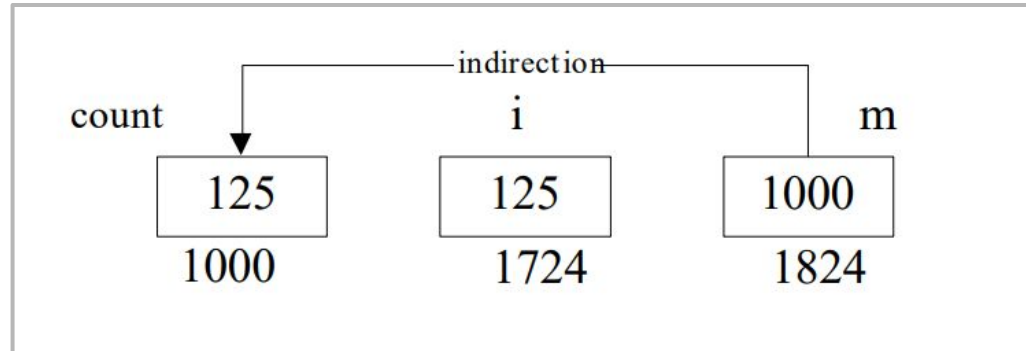
```
int y = 5;  
int *yPtr;  
yPtr = &y; //yPtr gets address of y  
yPtr "points to" y
```



Pointer Operators * and &

- **&** is a unary operator that returns the **address** a variable.
- ***** unary operator and it returns the **value** of the variable located at the address.

```
int *m ;  
int count=125, i ;  
  
m = &count ;  
  
i = *m ;
```



Pointers to Files

- FILE *inp; /* pointer to input file */
 - inp = fopen("distance.txt", "r");
 - fscanf(inp, "%lf", &item);
 - fclose(inp);
- FILE *outp; /* pointer to output file */
 - outp = fopen("distout.txt", "w");
 - fprintf(outp, "%.2f\n", item)
 - fclose(outp);

Pointers to Files

```
int
main(void)
{
    FILE *inp;          /* pointer to input file */
    FILE *outp;         /* pointer to output file */
    double item;
    int input_status;  /* status value returned by fscanf */

    /* Prepare files for input or output */
    inp = fopen("indata.txt", "r");
    outp = fopen("outdata.txt", "w");

    /* Read each item, format it, and write it */
    input_status = fscanf(inp, "%lf", &item);
    while (input_status == 1) {
        fprintf(outp, "%.2f\n", item);
        input_status = fscanf(inp, "%lf", &item);
    }

    /* Close the files */
    fclose(inp);
    fclose(outp);

    return (0);
}
```

File indata.txt

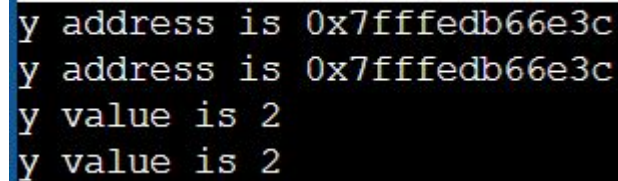
```
344 55 6.3556 9.4
43.123 47.596
```

File outdata.txt

```
344.00
55.00
6.36
9.40
43.12
47.60
```

Pointer Example

```
int main() {
int y=2;
int *p;
p = &y;
printf ("y address is %p \n", &y);
printf ("y address is %p \n", p);
printf ("y value is %d \n", y);
printf ("y value is %d \n", *p);
return 0;
}
```



```
y address is 0x7fffedb66e3c
y address is 0x7fffedb66e3c
y value is 2
y value is 2
```

Pointer Example

```
int i = 5;
int *ptr;           /* declare a pointer variable */
ptr = &i;          /* store address-of i to ptr */
printf("*ptr = %d\n", *ptr); /* refer to referee of ptr */
```

output
*ptr = 5

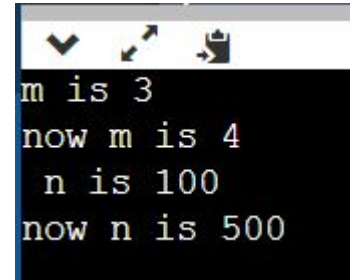
initialize pointers

```
int main()
{
    int x;
    int *p;
    scanf("%d",p); /*Incorrect ... the pointer is not initialized*/
    p = &x;
    scanf("%d",p); /* Correct */

    printf("%d",x);
}
```

Pointer Example

```
int m= 3, n=100, *p;  
p= &m;  
printf("m is %d\n",*p);  
m++;  
printf("now m is %d\n ",*p);  
p= &n;  
printf("n is %d\n",*p);  
*p= 500; /* *p is at the left of "= " */  
printf ("now n is %d\n ", n);
```



```
m is 3  
now m is 4  
n is 100  
now n is 500
```

Pointer Example

```
int main()  
{  
    int x, *p;  
    p = &x;  
    *p = 0;  
    printf("x is %d\n", x);  
    printf("*p is %d\n", *p);  
    *p += 1;  
    printf("x is %d\n", x);  
    (*p)++;  
    printf("x is %d\n", x);  
    return 0;  
}
```

Output:

x is 0

*p is 0

x is 1

x is 2

Pointer Example

Trace the execution of the following fragment

```
int m = 10, n = 5;  
int *mp, *np;  
mp = &m;  
np = &n;  
*mp = *mp + *np;  
*np = *mp - *np;  
printf("%d %d\n%d %d\n", m, *mp, n, *np);
```

Output:

15 15

10 10



Chapter 6

- Pointers as Function Parameters (Call by Reference)

Pointers as Function Parameters

- Write a function to exchange the values of two integer variables.

```
#include <stdio.h>
void swap( int*, int*);
void main( )
{
    int a, b ;
    printf( "Enter two numbers" ) ;
    scanf( " %d %d ", &a, &b ) ;
    printf( "a = %d ; b = %d \n", a, b ) ;
    swap( &a, &b ) ;
    printf( "a = %d ; b = %d \n", a, b ) ;
}
```

Call by Reference

```
void swap ( int *ptr1, int *ptr2 )
{
    int temp ;
    temp = *ptr2 ;
    *ptr2 = *ptr1 ;
    *ptr1 = temp ;
}
```

Pointers as Function Parameters

```
#include <stdio.h>
int sum(int,int);
int main()
{
    int num1=4,num2=5;
    int result;
    result=sum(num1,num2);
    printf("The result is %d",result);

    return 0;
}
int sum(int x,int y)
{
    return (x+y);
}
```

```
#include <stdio.h>
void sum(int*,int,int);
int main()
{
    int num1=4,num2=5;
    int result;
    sum(&result,num1,num2);
    printf("The result is %d",result);

    return 0;
}
void sum(int*res,int x,int y)
{
    *res=x+y;
}
```

Pointers as Function Parameters

```
#include <stdio.h>
void min_max(int, int, int*, int*);
int main()
{
    int x,y;
    int small,big;
    printf("Two integers: ");
    scanf (" %d %d" , &x , &y);
    min_max(x,y,&small,&big);
    printf("%d < = %d", small, big);
    return 0;
}
void min_max(int a, int b, int *min, int *max)
{
    if (a> b){
        *max= a;
        *min= b;
    }
    else{
        *max=b;
        *min=a;
    }
}
```

Function to find Max and Min for two numbers.

Pointers as Function Parameters

Function to find the sum and the difference between two numbers

```
#include <stdio.h>
int sum_difference (int, int, int*);
int main()
{
    int num1, num2, sum, diff;
    printf("Please enter two numbers: ");
    scanf("%d%d", &num1, &num2);
    diff=sum_difference (num1, num2, &sum);
    printf("Sum= %d and difference=%d", sum, diff);
    return 0;
}

int sum_difference (int x, int y, int* sum)
{
    *sum=x+y;
    return (x-y);
}
```

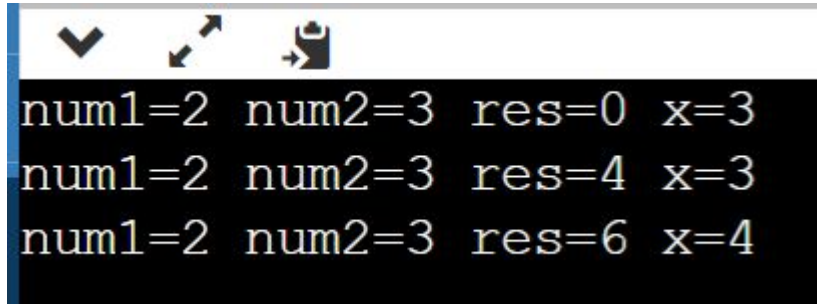
Pointers as Function Parameters

```
#include <stdio.h>
int main()
{
    int i = 0, j = 5;
    int *y;
    y=&j;
    for( i = 0; i <= 4; i++ )
    {
        *y = *y + i;
    }
    printf( "The final value of j is %d.\n", j );
    return 0;
}
```

The final value of j is 15.

Pointers as Function Parameters

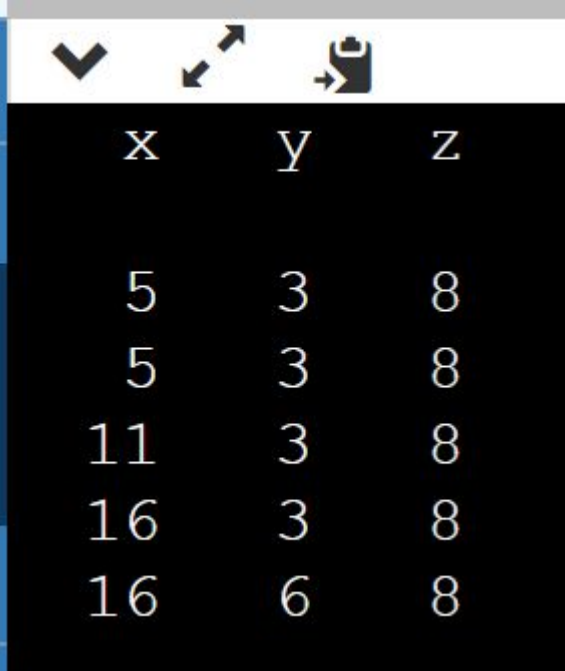
```
#include <stdio.h>
int x=2;
void fun1(int,int*,int);
void fun2(int,int*);
int main()
{
    int num1=2,num2=3,res=0;
    x=num1+1;
    printf("num1=%d num2=%d res=%d x=%d\n",num1,num2,res,x);
    fun1(num1,&res,num2);
    printf("num1=%d num2=%d res=%d x=%d\n",num1,num2,res,x);
    fun2(num2,&res);
    printf("num1=%d num2=%d res=%d x=%d\n",num1,num2,res,x);
    return 0;
}
void fun1(int x,int* y, int z)
{
    *y=x+z;
    *y=x+2;
}
void fun2(int y, int* z)
{
    *z=x+2;
    *z=y+3;
    x++;
}
```



```
num1=2 num2=3 res=0 x=3
num1=2 num2=3 res=4 x=3
num1=2 num2=3 res=6 x=4
```

Pointers as Function Parameters

```
#include <stdio.h>
void sum(int a, int b, int *cp);
int main(void)
{
    int x, y, z;
    x = 5; y = 3;
    printf("  x   y   z \n\n");
    sum(x, y, &z);
    printf("%4d%4d%4d\n", x, y, z);
    sum(y, x, &z);
    printf("%4d%4d%4d\n", x, y, z);
    sum(z, y, &x);
    printf("%4d%4d%4d\n", x, y, z);
    sum(z, z, &x);
    printf("%4d%4d%4d\n", x, y, z);
    sum(y, y, &y);
    printf("%4d%4d%4d\n", x, y, z);
    return (0);
}
void sum(int a, int b, int *cp)
{
    *cp = a + b;
}
```



| x | y | z |
|----|---|---|
| 5 | 3 | 8 |
| 5 | 3 | 8 |
| 11 | 3 | 8 |
| 16 | 3 | 8 |
| 16 | 6 | 8 |

Scope Rules

- **Local Variable** Inside a function or block
 - Can be accessed only in the scope of block.
 - **Void sum() { int result } ;**
- **Global variable** Outside of all function
 - Can be accessed from anywhere
- **Formal Parameters** In the function parameters
 - Can be accessed only in the scope of function.
 - **Void sum(int x , int y) { return x=y; }**

Scope Rules

- **Local Variable** Inside a function or block

```
int main()  
{  
    /* Declaration of local variable */  
    int a;  
  
    /* initialization */  
    a = 7;  
  
    printf ("value of a = %d\n", a);  
    return 0;  
}
```

Scope Rules

- **Global variable** Outside of all function

```
/* Declaration of global variable */  
int a;  
int main()  
{  
  
    /* initialization */  
    a = 7;  
  
    printf ("value of a = %d\n", a);  
    return 0;  
}
```



Thank You.

