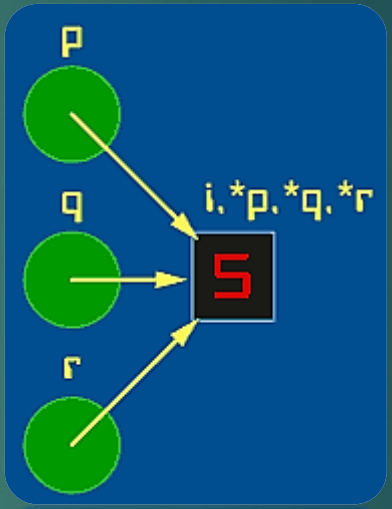
Chapter 9



Pointers

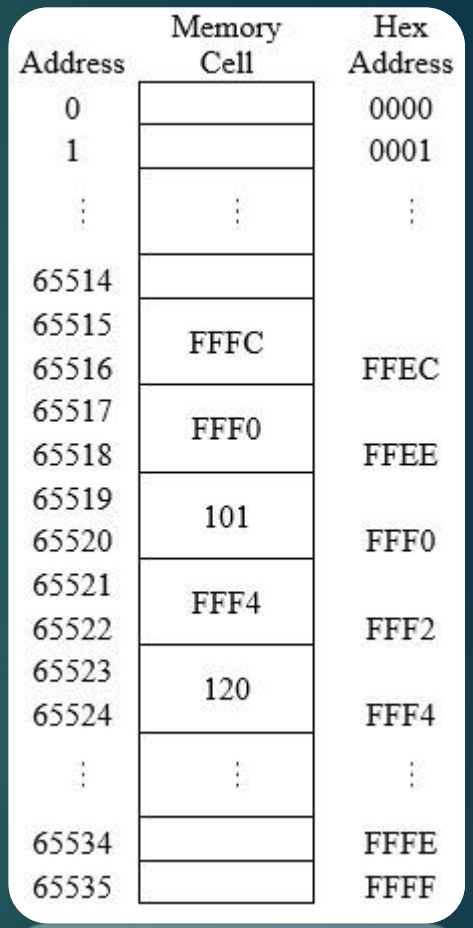
Er. Shiva K. Shrestha (HoD)

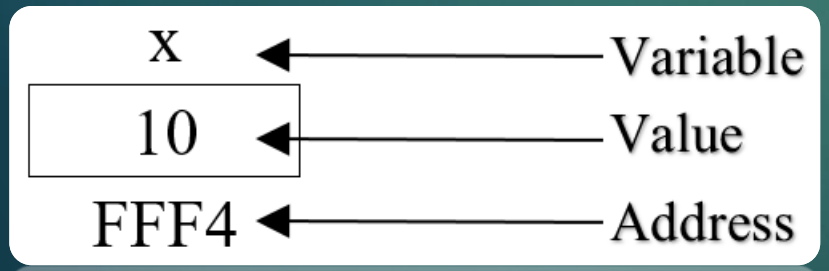
Department of Computer Engineering,

Khwopa College of Engineering

Pointer

- The Concept of Variable Representation
- Pointer Variable
- Reference or Address Operator (&)
  - marks_pointer = &marks;
- Dereference or Indirection Operator (*)
  - *marks_pointer = * (&marks) = marks;
- Declaring Variables of Pointer Types
  - Data_type * pointer_variable_name;





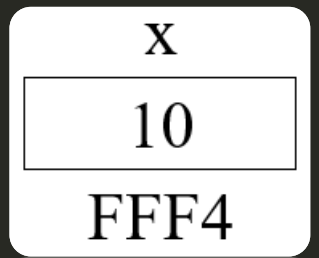
1. A program to display the contents of the pointer

PTR_1.c

```c
#include<stdio.h>
#include<conio.h>
void main(){
    int x;
    int *ptr;
    x=10;
    clrscr();
    ptr=&x;
    printf("x=%d  and ptr=%X",x,ptr);
    printf("\nx=%d and contents of ptr=%d",x,*ptr);
    getch();
}
```

DOSBox 0.74, Cpu speed: max

```
x=10  and ptr=FFF4
x=10 and contents of ptr=10
```



2. Assign a character variable to the pointer & display the contents of the pointer

DOSBox 0.74, Cpu speed: max 100%

```
Value of x=c and &x=FFF5
Pointer value ptr=c and &y=FFF4
```

PTR_2.C

```c
#include<stdio.h>
#include<conio.h>
void main(){
    char x,y;
    char *ptr;
    x='c'; /*assignment of character*/
    clrscr();
    ptr=&x;
    y=*ptr;
    printf("Value of x=%c and &x=%X", x,ptr);
    printf("\nPointer value ptr=%c and &y=%X",y,&y);
    getch();
}
```

x	y
c	c
FFF5	FFF4

3. A program to assign the pointer variable to another pointer and display the contents of the both the pointer variables

PTR_3.c

```c
#include<stdio.h>
#include<conio.h>
void main(){
    int x;
    int *ptr1,*ptr2;
    clrscr();
    x=10;
    ptr1=&x;
    ptr2=ptr1;
    printf("Value of x=%d and Address of x=%X",x,&x);
    printf("\nptr1: Content=%d, Value=%X and Address=%X",*ptr1,ptr1,&ptr1);
    printf("\nptr2: Content=%d, Value=%X and Address=%X",*ptr2,ptr2,&ptr2);
    getch();
}
```

x	ptr1	ptr2
10	FFF4	FFF4
FFF4	FFF2	FFF0

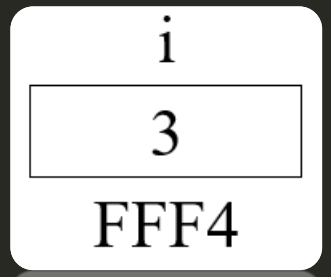
DOSBox 0.74, Cpu speed: max 100% cycles, Frame

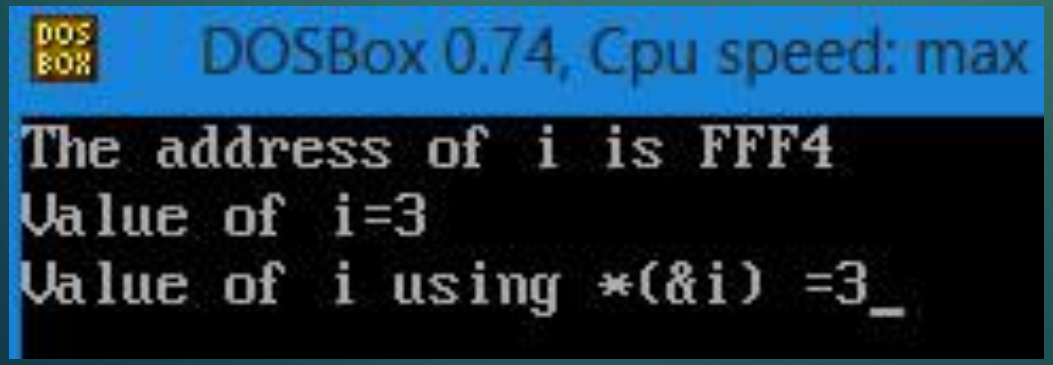
```
Value of x=10 and Address of x=FFF4
ptr1: Content=10, Value=FFF4 and Address=FFF2
ptr2: Content=10, Value=FFF4 and Address=FFF0
```

4. A program for assignment and usage of & operator

PTR_4_1.c

```c
#include<stdio.h>
#include<conio.h>
void main(){
    int i=3;
    clrscr();
    printf("The address of i is %X",&i);
    printf("\nValue of i=%d",i);
    printf("\nValue of i using *(&i) =%d",*(&i));
    getch();
}
```





Contd …

PTR_4_2.C

```c
#include<stdio.h>
#include<conio.h>
void main(){
    int *j,k=5;
    clrscr();
    j=&k;
    printf("Address of k=%X",&k);
    printf("\nAddress of j=%X",&j);
    printf("\nValue of j=%X",j);
    printf("\nValue of k=%d",k);
    printf("\nValue of k=%d",*j);
    getch();
}
```

j	k
FFF2	5
FFF4	FFF2

DOSBox 0.74, C

```
Address of k=FFF2
Address of j=FFF4
Value of j=FFF2
Value of k=5
Value of k=5
```

Double Pointer (Pointer to Pointer)

p	q	r
10	**FFF4**	*FFF2*
FFF4	*FFF2*	FFF0

PTR_4_3.C

```c
#include<stdio.h>
#include<conio.h>
void main(){
    int p=10,*q,**r;
    clrscr();
    q=&p;
    r=&q;
    printf("Address of p=%X",&p);
    printf("\nAddress of p=%X",q);
    printf("\nAddress of p=%X",*r);
    printf("\nAddress of q=%X",&q);
    printf("\nAddress of q=%X",r);
    printf("\nAddress of r=%X",&r);
    printf("\nValue of q=%X",q);
    printf("\nValue of r=%X",r);
    printf("\nValue of p=%d",p);
    printf("\nValue of p=%d",*(&p));
    printf("\nValue of p=%d",*q);
    printf("\nValue of p=%d",**r);
    getch();
}
```

DOSBox 0.74, Cpu

```
Address of p=FFF4
Address of p=FFF4
Address of p=FFF4
Address of q=FFF2
Address of q=FFF2
Address of r=FFF0
Value of q=FFF4
Value of r=FFF2
Value of p=10
Value of p=10
Value of p=10
Value of p=10_
```

5. A program to increment the pointer's address

PTR_5.c

```c
#include<stdio.h>
#include<conio.h>
void main(){
    int value,*ptr;
    clrscr();
    value=120;
    clrscr();
    ptr=&value;
    printf("ptr: Content=%d, Value=%X, Address=%X",*ptr,ptr,&ptr);
    ptr++;
    printf("\nptr: Content=%X, Value=%X, Address=%X",*ptr,ptr,&ptr);
    getch();
}
```

DOSBox 0.74, Cpu speed: max 100% cycles, Fr

```
ptr: Content=120, Value=FFF4, Address=FFF2
ptr: Content=0, Value=FFF6, Address=FFF2_
```

value	ptr	x	ptr1	ptr2
120	FFF4	101	FFF0	FFFC
FFF4	FFF2	FFF0	FFEE	FFEC
65524	65522	65520	65518	65516

```
DOSBox 0.74, Cpu speed: max 100% cycles, Frameskip  0, Pro
&value=FFF4,  &p=FFF2,  &x=FFF0,  &p1=FFEE,  &p2=FFEC
&value=65524, &p=65522, &x=65520, &p1=65518, &p2=65516

Value of p=FFF4
Memory address after increment=FFF6
Memory address after decrement=FFF4
x: Value=101, Address=FFF0
p1: Content=101, Value=FFF0, Address=FFEE
Address of (p2=p1+2)=FFEC
p2: Content=78, Value=FFF4, Address=FFEC_
```

```c
#include<stdio.h>
#include<conio.h>
void main(){
    int value,*p;
    int x,*p1,*p2;
    clrscr();
    printf("&value=%X,  &p=%X,  &x=%X,  &p1=%X,  &p2=%X",&value,&p,&x,&p1,&p2);
    printf("\n&value=%u, &p=%u, &x=%u, &p1=%u, &p2=%u\n",&value,&p,&x,&p1,&p2);
    value=120;
    p=&value;
    printf("\nValue of p=%X",p);
    p++;
    printf("\nMemory address after increment=%X",p);
    p--;
    printf("\nMemory address after decrement=%X",p);
    x=101;
    printf("\nx: Value=%d, Address=%X",x,&x);
    p1=&x;
    printf("\np1: Content=%d, Value=%X, Address=%X",*p1,p1,&p1);
    p2=p1+2; /* 2*2 Bytes */
    printf("\nAddress of (p2=p1+2)=%X",&p2);
    printf("\np2: Content=%X, Value=%X, Address=%X",*p2,p2,&p2);
    getch();
}
```

Address	Memory Cell	Hex Address
0		0000
1		0001
⋮	⋮	⋮
65514		
65515	FFFC	
65516		FFEC
65517	FFF0	
65518		FFEE
65519	101	
65520		FFF0
65521	FFF4	
65522		FFF2
65523	120	
65524		FFF4
⋮	⋮	⋮
65534		FFFE
65535		FFFF

Some Terminologies

- Bad Pointer – Declared, but uninitialized pointer
- Void Pointer – points to variables of any data type
- Null Pointer – points nowhere or nothing
- Double Pointer – **ptr
- Array of Pointer – data_type *pointer_name[size]

Pointers & Arrays

Equivalent Expressions of Arrays & Pointers

Type of Array	To be Accessed	Technique	
		Array Notation	Pointer Notation
1D	Address of ith Element	&marks[i]	(marks+i)
	Value of ith Element	marks[i]	*(marks+i)
2D	Address of Element at i^{th} Row & j^{th} Column	&marks[i][j]	(*(marks+i)+j)
	Value of Element at i^{th} Row & j^{th} Column	marks[i][j]	*(*(marks+i)+j)

Remember: i = *(&i)

Relationship between 1D Array & Pointer

Q. WAP to access the array element using pointer

Note: i = *(&i)

```
DOSBox 0.74, Cpu speed: max 100% cycles, Fran
1       2       3       4       5       _
```

PTR_12.c | PTR_12_1.c | PTR_12_2.c

```c
#include<stdio.h>
#include<conio.h>
void main(){
    int i,myArray[5]={1,2,3,4,5};
    clrscr();
    for(i=0;i<5;i++){
        printf("%d\t",*(myArray+i));
        //printf("%d\t",myArray[i]);
    }
    getch();
}
```

PTR_12_1.c | PTR_12_2.c | PTR_12_3.C

```c
#include<stdio.h>
#include<conio.h>
void main(){
    int myArray[]={1,2,3,4,5};
    int *ptr2myArray,i;
    clrscr();
    ptr2myArray=myArray;
    for(i=0;i<5;i++){
        printf("%d ",*(ptr2myArray++));
    }
    getch();
}
```

```
DOSBox 0.74
1 2 3 4 5
```

Array is internal pointer …

String & Pointer

Array - Internal Pointer

```
int x[5]={1,2,3,4,5};
int *p;
p=x; /* p=&x[0] */
```

Element	x[0]	x[1]	x[2]	x[3]	x[4]
Value	1	2	3	4	5
Address	1000	1002	1004	1006	1008

```
p=&x[0];
p=&x[1];
p=&x[2];
p=&x[3];
p=&x[4];
```

PTR_13_v2.c

```c
#include<conio.h>
#include<stdio.h>
int main(){
    char x[]="Welcome to Khwopa!\n";
    char *y="C Programming is easy.";
    printf("%s",x);
    printf("%s",y);
    getch();
    return 0;
}
```

C:\Users\ErSKS\Google Drive (c.khwopa

```
Welcome to Khwopa!
C Programming is easy.
```

2D Array & Pointer

WAP to add two 3*3 matrices using pointer

P17_V2.C

```c
#include<stdio.h>
#include<conio.h>
#define M 3
#define N 3
int main(){
    int i, j;
    int (*a)[N], (*b)[N], (*sum)[N];
    clrscr();
    printf("Enter first matrix:\n");
    for (i = 0; i < M; i++){
        for (j = 0; j < N; j++){
            //scanf("%d",&a[i][j]);
            scanf("%d",*(a+i)+j);
        }
    }
    printf("Enter second matrix:\n");
    for (i = 0; i < M; i++){
        for (j = 0; j < N; j++){
            //scanf("%d",&b[i][j]);
            scanf("%d",*(b+i)+j);
        }
    }
    printf("\nThe sum matrix is:\n");
    for (i = 0; i < M; i++){
        for (j = 0; j < N; j++){
            //sum[i][j] = a[i][j] + b[i][j]
            *(*(sum+i)+j) = *(*(a+i)+j) + *(*(b+i)+j);
            //printf("%d\t", sum[i][j]);
            printf("%d\t", *(*(sum+i)+j));
        }
        printf("\n");
    }
    getch();
    return 0;
}
```

DOSBox 0.74, Cpu speed: max 100% cycles, Frameskip 0, Progra...

```
Enter first matrix:
1       1       1
1       1       1
1       1       1
Enter second matrix:
2       2       2
2       2       2
2       2       2

The sum matrix is:
3       3       3
3       3       3
3       3       3
```

Pointer Arithmetic

C support four arithmetic operators

1. Addition +
2. Subtraction -
3. Increment ++
4. Decrement --

PTR_7_1.C

```c
#include<stdio.h>
#include<conio.h>
void main(){
    int x,y,*ptr;
    clrscr();
    x=10;
    ptr=&x;
    printf("Value of x=%d and pointer=%d",x,*ptr);
    y=++ *ptr;  /*y=11*/
    printf("\nValue of y=%d and pointer=%d",y,*ptr);
    getch();
}
```

x	y	ptr
10	11	&x
&x	&y	&ptr

DOSBox 0.74, Cpu speed: max 100% cycles, Frameskip 0, Program: TC

```
Value of x=10 and pointer=10
Value of y=11 and pointer=11_
```

P_7_1.C

```c
#include<stdio.h>
#include<conio.h>
void main(){
    int x,y,*p;
    clrscr();
    x=10, y=20;
    p=&x;
    printf("x=%d, &x=%X, y=%d, &y=%X\n",x,&x,y,&y);
    y=*p + 5;
    printf("\nx=%d, y=%d, *p=%d, p=%X, &p=%X", x, y, *p, p, &p);
    y=++ *p;  /*y=11*/
    printf("\nx=%d, y=%d, *p=%d, p=%X, &p=%X", x, y, *p, p, &p);
    p--;
    y=100;
    printf("\nx=%d, y=%d, *p=%d, p=%X, &p=%X", x, y, *p, p, &p);
    getch();
}
```

DOSBox 0.74, Cpu speed: max 100% cycles, Frameskip 0, Progra...

```
x=10, &x=FFF4, y=20, &y=FFF2

x=10, y=15, *p=10, p=FFF4, &p=FFF0
x=11, y=11, *p=11, p=FFF4, &p=FFF0
x=11, y=100, *p=100, p=FFF2, &p=FFF0_
```

PTR_7_2.c

```c
#include<stdio.h>
#include<conio.h>
void main(){
    int x,y,*x_pointer,temp;
    clrscr();
    temp=3;
    x=5*(temp+5);
    x_pointer=&temp;
    y=6*(*x_pointer+5);
    printf("x=%d",x);
    printf("\ny=%d",y);
    printf("\nAddress of temp=%X",&temp);
    getch();
}
```

DOSBox 0.74, Cpu sp

```
x=40
y=48
Address of temp=FFF0
```

x	y	x_pointer	temp
40	48	FFF0	3
			FFF0

PTR_7_3.c

```c
#include<stdio.h>
#include<conio.h>
void main(){
    int x,y,*p1,*p2;
    clrscr();
    x=25;
    p1=&x;
    printf("&x=%X, &y=%X, &p1=%X, &p2=%X\n",&x,&y,&p1,&p2);
    printf("Contents of pointer=%d",*p1);
    *p1=*p1+1; // ++*p1
    y=*p1;
    printf("\nx=%d, y=%d and (*p1=*p1+1)=%d",x,y,*p1);
    (*p1)++;
    y=*p1;
    printf("\nx=%d, y=%d and (*p1)++=%d",x,y,*p1);
    ++ *p1;
    y=*p1;
    printf("\nx=%d, y=%d and ++ *p1=%d",x,y,++ *p1);
    ++ *p1;
    p2=p1;
    printf("\n\n*p1=%d",*p1);
    printf("\n*p2=%d",*p2);
    getch();
}
```

x	y	ptr	ptr2
25	26	&x	&x
&x	&y	&ptr	&ptr2

26
27
28
29
30

DOSBox 0.74, Cpu speed: max 100% cycles,

```
&x=FFF4, &y=FFF2, &ptr=FFF0, &ptr2=FFEE
Contents of pointer=25
x=26, y=26 and (*ptr=*ptr+1)=26
x=27, y=27 and (*ptr)++=27
x=29, y=28 and ++ *ptr=29

*ptr=30
*ptr2=30_
```

Pointer Operations

Er. Shiva K. Shrestha (HoD, Computer Department)
2019-02-09

Valid Operations

- Assignment to a pointer of the same type
- Assigning a pointer to pointer of type (void *) and back
- Adding/subtracting a pointer & an integer (including increment & decrement)
- Subtracting or comparing two pointers which point to members of the same array
- Assigning or comparing to zero

Invalid Operations

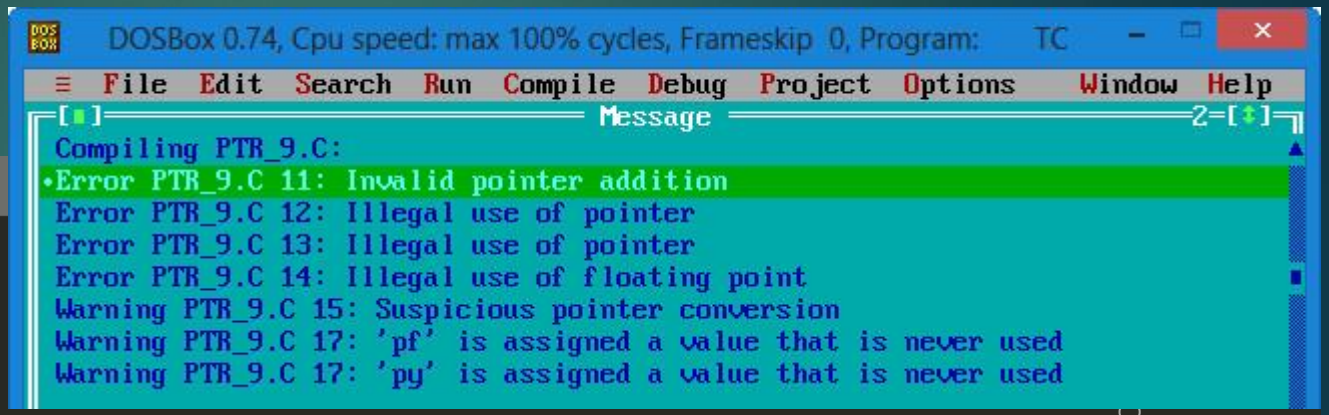
- Adding two pointers
- Multiply, divide, shift, mask pointers
- Add float or double numbers to a pointer
- Assign a pointer of different types without cast

Valid Pointer Operations

PTR_8.C

```c
#include<stdio.h>
#include<conio.h>
#define NULL 0
void main(){
    int x,y;
    int *px=&x;
    int *py;
    void *pv;
    clrscr();
    py=px;          /* Assignment to ptr of same type*/
    px=(int *) pv;  /* recast a (void *) pointer */
    pv=(void *)px;  /* recast to type (void *) */
    py=px+2;        /* Add/Sub ptr & integer is legal */
    px++;           /* Inc/Dec is legal*/
    if(px==NULL)    /* Compare to null pointer*/
    py=NULL;        /* Assign to null pointer*/
    getch();
}
```

Invalid Pointer Operations



PTR_9.c

```c
#include<stdio.h>
#include<conio.h>
#define NULL 0
void main(){
    int x,y;
    int *px,*py,*p;
    float *pf;
    clrscr();
    px=&x;
    py=&y;
    p=px+py;     /*Addition of two pointer is illegal */
    p=px*py;     /* Multiplication of two pointer is illegal */
    p=px/py;     /* Division of two pointer is illegal */
    p=px+10.5;   /* Addition of float to pointer is illegal */
    pf=px;       /* Assignment of different types of pointer is illegal */
    getch();
}
```

Pointers and Functions

Call By Reference

PTR_10.C

```c
#include<stdio.h>
#include<conio.h>
void swap(int a,int b){
    int temp;
    temp=a;
    a=b;
    b=temp;
}
void main(){
    int x=100,y=20;
    clrscr();
    printf("Values before swap\n");
    printf("x=%d and y=%d",x,y);
    swap(x,y);
    printf("\nValues after swap\n");
    printf("x=%d and y=%d",x,y);
    getch();
}
```

DOSBox 0.74, Cpu speed:

```
Values before swap
x=100 and y=20
Values after swap
x=100 and y=20_
```

PTR_10.C PTR_11.C

```c
#include<stdio.h>
#include<conio.h>
void swap(int *a,int *b){
    int temp;
    temp=*a;
    *a=*b;
    *b=temp;
}
void main(){
    int x=100,y=20;
    printf("Values before swap\n");
    printf("x=%d and y=%d",x,y);
    swap(&x,&y);
    printf("\nValues after swap\n");
    printf("x=%d and y=%d",x,y);
}
```

DOSBox 0.74, Cpu speed:

```
Values before swap
x=100 and y=20
Values after swap
x=20 and y=100
```

Structure & Pointer

C language allows declaring pointer to structure just like pointer to other ordinary variables. The declaration can be done in the following manner:

```
struct employee{
    char *name;
    char *roll_no;
    int salary;
}*emp1;
```

C8_16.C

```c
#include<stdio.h>
#include<conio.h>
int main(){
    struct book{
        char name[25];
        char author[25];
        int call_no;
    };
    struct book b1={"Let us C","YK",101};
    struct book *ptr;
    ptr=&b1;
    printf("%s\t%s\t%d\n",b1.name,b1.author,b1.call_no);
    printf("%s\t%s\t%d\n",ptr->name, ptr->author, ptr->call_no);
    getch();
    return 0;
}
```

C:\Users\...

```
Let us C        YK      101
Let us C        YK      101
```

Applications of Pointer

- enhances execution speed of a program
- saves memory space
- used to construct different data structures such as linked lists, queues, stacks, etc.
- supports dynamic allocations and de-allocations of memory segments

Find smallest no. using pointer

PTR_15.C

```c
#include<stdio.h>
#include<conio.h>
void main(){
    int i,n,small,*ptr,a[10];
    clrscr();
    printf("How many elements? ");
    scanf("%d",&n);
    for(i=0;i<n;i++){
      scanf("%d",&a[i]);
    }
    //Assign address of a[0] to pointer variable
    //It can be done in two ways ptr=&a[0] or ptr=a
    ptr=a;
    small=*ptr;
    ptr++; //Pointer points to next location in an array
    for(i=1;i<n;i++){ //loop n-1 times to search smallest element
        if(small>*ptr){
            small=*ptr;
        }
        ptr++; //pointer is incremented to pointed a[i+1]
    }
    printf("\nSmallest element of the array: %d",small);
    getch();
}
```

DOSBox 0.74, Cpu speed: max 100% cy

```
How many elements? 5
45 -70 50 1 67

Smallest element of the array: -70_
```

Function Pointer

DOSBox 0.74, Cpu speed: max 100% cycles, Frameskip 0, Progra...

File Edit Search Run Compile Debug Project Options Window Help

FXN_PTR.C

```c
#include<stdio.h>
#include<conio.h>
int sum(int a, int b){
        return (a+b);
}
int (*ptr)(int, int);
void main(){
    int s1, s2;
    clrscr();
    s1 = sum(5,7);
    printf("\nSum = %d",s1);
    ptr = &sum;
    s2 = ptr(10,20);
    printf("\nSum = %d",s2);
    getch();
}
```

DOSBox 0.74, Cpu speed: max 100% cycles, Frameskip 0, Progra...

```
Sum = 12
Sum = 30_
```

14:5

F1 Help F2 Save F3 Open Alt-F9 Compile F9 Make F10 Menu

Functions returning Pointer Variable

int* larger(int *, int *);

PTR_16.C

```c
#include<stdio.h>
#include<conio.h>
int *larger(int *,int *); // Prototype
void main(){
    int a=10,b=20,*p;
    clrscr();
    p=larger(&a,&b); // Function Call
    printf("Largest Value=%d",*p);
    getch();
}
int *larger(int *x,int *y){
    if (*x>*y){
        return(x); // Address of a
    }else{
        return(y); // Address of b
    }
}
```

DOSBox 0.74, Cpu speed:

```
Largest Value=20_
```

Q/A?

Thank You!

Er. Shiva K. Shrestha

computer.khwopa@gmail.com