



Programming in 'C'

Pointers & Structures

Objectives

- To be able to:
 - describe what pointers are
 - understand the various uses of pointers
 - declare and use pointers
 - Data Types review
 - User Defined Types
 - Declaring Structures
 - Structure Definition
 - Accessing Structure elements.

Slide 2 (of 24)

What are Pointers?

- Pointers are variables;
- They hold the memory address of other variables;
- They must be of the same type as the variable to which they point.

Slide 3 (of 24)

Why use pointers?

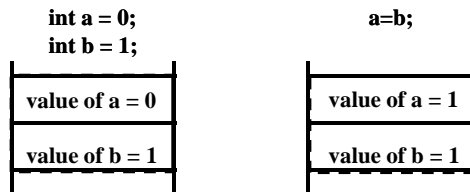
Pointers have three uses in C:

- to create dynamic data structures
 - data structures built up from blocks of memory allocated from the heap at run-time;
- to access data in arrays;
- allow variable parameters passed to functions.

Slide 4 (of 24)

Variables that hold data

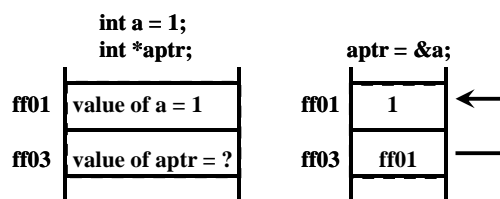
- The use of variable names give access to contents of storage reserved for variable value:



Slide 5 (of 24)

Variables that hold addresses (Pointers)

- A pointer gives access to the memory address of a variable:



Slide 6 (of 24)

Pointers (2)

• Pointer Declarations

- like any type of declaration, but includes *
- ```
int *aptr; // reserve storage for
 // pointer to integer
char *string; // reserve storage for
 // pointer to character
```

### • Address Operator &

```
int a;
int *aptr;
aptr = &a; // sets aptr to address
 // of variable 'a'
```

Slide 7 (of 24)

---

---

---

---

---

---

---

---

## Pointers (3)

### • Pointer de-reference operator \*

```
char c1 = 'z', c2;
char *cptr;
cptr = &c1;
c2 = *cptr; // 'c2' becomes the value
 // pointed to by 'cptr'
```

### • Incrementing a pointer

- e.g.: `bufptr++;`
- pointer contains address of next storage location

Slide 8 (of 24)

---

---

---

---

---

---

---

---

## Pointers Example

```
/* NAME OF SOURCE FILE; POINT_EX.C
 PROGRAM DESCRIPTION; Demonstrate pointers

 LIBRARIES; */
#include<stdio.h>

// FUNCTION PROTOTYPES;
int TwoTimes(int CopyValue); //pass by value
void TwoTime_NoReturn(int *PtrValue); //pass by ref.

//===== MAIN BLOCK =====
```

Slide 9 (of 24)

---

---

---

---

---

---

---

---

## Pointers Example

```
void main(void){
 int FValue, RtrnVal;
 int *FPtr;
 clrscr();
 printf("\nEnter a number between 1 and 100: ");
 scanf("%d",&FValue);

 RtrnVal = TwoTimes(FValue);
 printf("\nFValue is %d, \nRtrnVal is %d",FValue, RtrnVal);

 printf("\nNow passed as pointer");
 FPtr = &FValue; //FPtr becomes the value of address of FValue
 TwoTime_NoReturn(FPtr);//pass a copy of the address of FValue

 printf("\nFValue is %d (new)",FValue);
 getch();
}
```

Slide 10 (of 24)

---

---

---

---

---

---

---

---

## Pointers Example

Demo

```
/* ***** FUNCTION ***** */
int TwoTimes(int CopyValue){
 int RetV;

 printf("\n\nFunction TwoTimes called and executed");
 RetV = CopyValue * 2;

 return(RetV);
}

/* ***** FUNCTION ***** */
void TwoTime_NoReturn(int *PtrValue){

 printf("\n\nFunction TwoTime_NoReturn called and executed");
 *PtrValue *=2 ;
 //Multiply the value at the address held in "PtrValue" by 2
}
```

Slide 11 (of 24)

---

---

---

---

---

---

---

---

## Data Types (review)

- Data types:
  - Only hold one piece of data in a variable,  
e.g.: int Size;  
float ScaleFactor;  
char Option;

Slide 12 (of 24)

---

---

---

---

---

---

---

---

## Data Types (review)

- Arrays:
  - Can hold multiples of the same data type using the same variable name,  
e.g.: `int Marks [50];`  
`char StuName [27];`

Slide 13 (of 24)

---

---

---

---

---

---

---

---

## Structures

- User defined type in C
  - groups several basic types under one name
  - like a record in other languages

Slide 14 (of 24)

---

---

---

---

---

---

---

---

## Structures

- Declaration
  - declare a structure called time, with three int members:

```
struct{
 int hour; //struct members, type int
 int minute;
 int second;
}time; // struct name is time
```

Slide 15 (of 24)

---

---

---

---

---

---

---

---

## Problem

Write a program that will hold the following data on electric motors, each motor requiring 5 data items.

| Entity or data object: ElectricMotor |              |           |
|--------------------------------------|--------------|-----------|
| Variable                             | Content Type | Data Type |
| Model No                             | Alphanumeric | char      |
| Nom V                                | Real Number  | float     |
| Speed (rpm)                          | Whole Number | int       |
| Current (A)                          | Real Number  | float     |
| Output (W)                           | Real Number  | float     |

Slide 16 (of 24)

---

---

---

---

---

---

---

---

## Problem

| Entity or data object: ElectricMotor |      |      |       |      |       |       |
|--------------------------------------|------|------|-------|------|-------|-------|
|                                      | 0    | 1    | 2     | ..   | ..    | 49    |
| Model No                             | RE14 | RE28 | RE32  | RE54 | RE70  | RE85  |
| Nom V                                | 1.5  | 1.5  | 12.0  | 7.0  | 6.0   | 9.6   |
| Speed (rpm)                          | 6250 | 3750 | 19000 | 9600 | 11500 | 16500 |
| Current (A)                          | 0.62 | 0.53 | 6.0   | 2.1  | 7.10  | 10.3  |
| Output (W)                           | 0.4  | 0.44 | 42    | 7.6  | 21.2  | 28.1  |

Allows for the storage of up to 50 records or objects.  
Each column represents a record or object.

Slide 17 (of 24)

---

---

---

---

---

---

---

---

## Creating a Structure

```
//Global area of program

struct ElectricMotor {
 char Model[5];
 float NomV;
 int RPM;
 float CurrentA;
 float OutputW;
};

//===== main =====
void main(void){
 int Xyz;
 struct ElectricMotor Mtr;
```

---

---

---

---

---

---

---

---

## Creating a Structure

```
//Global area of program
struct ElectricMotor {
 char Model[5];
 float NomV;
 int RPM;
 float CurrentA;
 float OutputW;
};
//===== main =====
void main(void){
 int Xyz;
 struct ElectricMotor Mtr;
```

Annotations in the code:

- Keyword: points to `struct`
- Structure Tag: points to `ElectricMotor`
- Data Type: points to `float`
- Variable Name: points to `Mtr`

---

---

---

---

---

---

---

---

## Accessing Data Elements

```
void main(void){
 int Xyz;
 struct ElectricMotor Mtr;

 strcpy(Mtr.Model,"RT76");
 Mtr.NomV = 12.5;
 printf("\nEnter Speed (RPM): ");
 scanf("%d",&Mtr.RPM);
}
```

Annotation: dot operator for accessing structure data element. (points to `Mtr.RPM`)

Slide 20 (of 24)

---

---

---

---

---

---

---

---

## Structure Definition & Declaration

```
//Global area of program
struct ElectricMotor {
 char Model[5];
 float NomV;
 int RPM;
 float CurrentA;
 float OutputW;
}Mtr;
```

Structure variable declared at end of definition - becomes a Global variable.



Slide 21 (of 24)

---

---

---

---

---

---

---

---

## Structure Arrays

```
//Global area
struct Student{
 char StuName[30];
 int Marks;
}AllStudents[250];
//===== main =====
void main(void){
 int i;

 for(i=0;i<250;i++){
 printf("\nEnter Student Name: ");
 scanf("%s",AllStudents[i].StuName);
 printf("\nEnter Marks: ");
 scanf("%d",&AllStudents[i].Marks);
 }
}
```

Slide 22 (of 24)

---

---

---

---

---

---

---

---

## Summary (1) - Pointers

- Pointers:
  - variable that 'points' to a memory location;
  - must be of same type as data pointed to;
  - used in dynamic allocation of data types & structures;
  - allows variables values to be passed between functions;
  - can access data in arrays.

Slide 23 (of 24)

---

---

---

---

---

---

---

---

## Summary (2) - Structure

- Represents an object or record
- Is user defined type
- Can consist of of different data type elements
- Can include other structures
- Can be single objects or arrays.

Slide 24 (of 24)

---

---

---

---

---

---

---

---