| **structures & pointers**  **Structures : declaration, definition and initialization of structures, unions Pointers: Concepts, declaration, initialization of pointer variables, Accessing a Variable through its Pointer Chain of Pointers, Pointer Expressions, Pointer Increments and Scale Factor, Pointers and Arrays, examples** |
| --- |



**Structure**

a structure is a user defined data type.structure is the collection of variables of different data type.which are grouped under a common name for better handling.

**What is the need of structure**

Consider we want to store information about a car,it includes car name,fuel type,cc,milage.since we have four data.we have to initialize 4 variables to store this data.

If we want to store same information about100 cars we have to initialize 400 variables.

This is not practical,bcz, this will increase program size ,need more time to write etc.

**(note: we can’t use array bcz these information are not in the same data type.)**

In this case we can use structure.

**Defining structure ( we are creating a structure)**

In order to use a structure firstly we have define or create a structure.we can do this anywhere inside a program(ie,inside or outside the main|( ) function).

**Syntax:**

**Struct structure name**

**{**

**Data type member 1;**

**Data type member 2;**

**….…..**

**….……**

**….…**

**};**

* “Struct” is the keyword for structure.
* ”struct name” is the name of structure .it is also called as tag of structure.
* Members are variables of different data types.data type should be mentioned while declaring members.

Eg: consider we have store information about 2 books,it include books price and no.of pages in it.

**Struct book**

**{**

**Int pages;**

**Float price;**

**};**

Here pages are integer type member (ie,variable) and price is float type member,(note:structure variables are not declared in this)

**Declaration of structure variables**

In this section we are allocating memory for structure data

Structure variable declaration can be done along with structure definition. We can do this in 2 ways.

1. **Syntax:**

**Struct structure name**

**{**

**Data type member 1;**

**Data type member 2;**

**….…..**

**….……**

**….…**

**} variable 1, variable 2, variable 3 ……. ;**

**Eg:**

**Struct book**

**{**

**Int pages;**

**Float price;**

**}book 1,book 2;**

**In this case book 1,book 2 is the structure variables**

1. **Syntax:**

**Struct**

**{**

**Data type member 1;**

**Data type member 2;**

**….…..**

**….……**

**….…**

**} variable 1, variable 2, variable 3 ……. ;**

**In this method we are not used tag**

**Note**

**We can define a structure with out tag,but all the structure variables should be declared in the definition it self.**

**Eg:**

**Struct**

**{**

**Int pages;**

**Float price;**

**}book 1,book 2;**

1. **In this method structure variables are declaring outside structure definition.**

**Syntax:**

**Struct structure name**

**{**

**Data type member 1;**

**Data type member 2;**

**….…..**

**….……**

**….…**

**} ;**

**Struct structure name variable 1, variable 2, variable 3 …. ;**

**Eg:**

**Struct book**

**{**

**Int pages;**

**Float price;**

**};**

**Struct book book 1,book 2;**

**Initialization of structure variables**

In this section we are adding data to structure variable

1. **Compile time initialization**
2. **Syntax:**

**Struct structure name**

**{**

**Data type member 1;**

**Data type member 2;**

**….…..**

**….……**

**….…**

**}variable1={data1,data2…..},variable2={data1,data2…..},variable3={data 1,data2…..} ……. ;**

**Eg:**

**Struct book**

**{**

**Int pages;**

**Float price;**

**}book 1={10,20.5},book 2 ={30,30.4};**

1. **Syntax:**

**Struct**

**{**

**Data type member 1;**

**Data type member 2;**

**….…..**

**….……**

**….…**

**}variable1={data1,data2…..},variable2={data1,data2…..},variable3={data 1,data2…..} ……. ;**

**Eg:**

**Struct**

**{**

**Int pages;**

**Float price;**

**}book 1={10,20.5},book 2 ={30,30.4};**

1. **Syntax:**

**Struct structure name**

**{**

**Data type member 1;**

**Data type member 2;**

**….…..**

**….……**

**….…**

**} ;**

**Struct structure name variable 1={data 1,data2…..}, variable 2={data 1,data2…..}, variable 3={data 1,data2…..},……;**

**Eg:**

**Struct book**

**{**

**Int pages;**

**Float price;**

**};**

**Struct book book 1={10,20.5},book 2 ={30,30.4};**

1. Another method is using dot operator **(“ . ”)**

**Syntax:**

**Variable 1 . member 1=data;**

**Variable 1 . member 2=data;**

**….…………**

**….…………**

**Eg:**

**Struct**

**{**

**Int pages;**

**Float price;**

**}book 1,book 2;**

**book1.pages=10;**

**book 1.price=20.5;**

**book 2.pages =30;**

**book2.price=30.4;**

1. **Run time initialization**

**In order to do this we have to use dot operator**

**Syntax**

**Scanf(“control strings”,Variable 1 . member 1,Variable 1 . member 2…….);**

**Eg:**

**Struct**

**{**

**Int pages;**

**Float price;**

**}book 1,book 2;**

**Scanf(“%d%f%d%f”,&book1.pages,&book 1.price,&book 2.pages ,&book2.price);**

**Accessing of members of structure**

we have to use dot operator for this

**Eg:**

**Struct**

**{**

**Int pages;**

**Float price;**

**}book 1,book 2;**

**Scanf(“%d%f%d%f”,&book1.pages,&book 1.price,&book 2.pages ,&book2.price);**

**Printf(“%d”,book1.pages);**

**Example program: program to print all values in structure**

**#include <stdio.h>**

**#include <conio.h>**

**Void main()**

**{**

**Clrscr();**

**Struct book**

**{**

**Int pages;**

**Float price;**

**}book 1,book 2;**

**Scanf(“%d%f%d%f”,&book1.pages,&book 1.price,&book 2.pages ,&book2.price);**

**Printf(“%d%f%d%f”,book1.pages,book 1.price,book 2.pages ,book2.price);**

**Getch();**

**}**

**Explanation:**

* here structure name is book,it can store a int data in variable pages (member) and a float data in variable price.
* Structure variable are book1 and book2.that is we can store data regarding book1 and book2 in this structure.
* Here we employed runtime initialization of structure variables

**Example program:**program to store details of two students

**#include <stdio.h>**

**#include <conio.h>**

**Void main()**

**{**

**Clrscr();**

**Int x;**

**Struct student**

**{**

**Char name[10];**

**Int a;**

**Float b;**

**}stdnt1,stdnt2;**

**Printf(“enter details of student 1”);**

**Scanf(“%S%d%f”,stdnt1.name,&stdnt1.a,&stdnt1.b );**

**Printf(“enter details of student 2”);**

**Scanf(“%S%d%f”,stdnt2.name,&stdnt2.a,&stdnt2.b );**

**Printf(“choose student “);**

**Scanf(“%d”,&x);**

**If(x==1)**

**{**

**Printf(“ details of student 1 is given below”);**

**printf(“student name=%S”,stdnt1.name);**

**Printf(“roll number=%d”,stdnt1.a);**

**Printf(“percentage mark=%f”,stdnt1.b);**

**}**

**Else**

**{**

**Printf(“ details of student 2 is given below”);**

**printf(“student name=%S”,stdnt2.name);**

**Printf(“roll number=%d”,stdnt2.a );**

**Printf(“percentage mark=%f”,stdnt2.b);**

**}**

**Getch();**

**}**

**Explanation**

* Here we are storing details of two students in structure(details include student name,roll number and mark percentage) .”**student”** is the structure name.
* **Stdnt1 and stdnt2** is the structure variables.
* **Name[10] ,a,b** are the members of structure.**”name[10]”** is a string (or character array )that can store up to 10 characters.we don’t have to use **‘&’** in scanf ,if we are using a string as variable. **“a”** is integer type variable using to store roll number. **“b”** is a float type variable using to store mark percentage .
* Here we are used run time initialization to add data to structure variables.
* If we enter 1 in student choosing section ,we will get details about student 1 on screen.
* Else,,we will get details about student 2 on screen.

**(Note:in the above example we defined structure inside main function and variables are declared along with it)**

**#include <stdio.h>**

**#include <conio.h>**

**Struct student**

**{**

**Char name[10];**

**Int a;**

**Float b;**

**}stdnt1,stdnt2;**

**Void main()**

**{**

**Clrscr();**

**Int x;**

**Printf(“enter details of student 1”);**

**Scanf(“%S%d%f”,stdnt1.name,&stdnt1.a,&stdnt1.b );**

**Printf(“enter details of student 2”);**

**Scanf(“%S%d%f”,stdnt2.name,&stdnt2.a,&stdnt2.b );**

**Printf(“choose student “);**

**Scanf(“%d”,&x);**

**If(x==1)**

**{**

**Printf(“ details of student 1 is given below”);**

**printf(“student name=%S”,stdnt1.name);**

**Printf(“roll number=%d”,stdnt1.a);**

**Printf(“percentage mark=%f”,stdnt1.b);**

**}**

**Else**

**{**

**Printf(“ details of student 2 is given below”);**

**printf(“student name=%S”,stdnt2.name);**

**Printf(“roll number=%d”,stdnt2.a );**

**Printf(“percentage mark=%f”,stdnt2.b);**

**}**

**Getch();**

**}**

**Note: In the above example structure definition and structure variable declaration is done outside main() function**

**#include <stdio.h>**

**#include <conio.h>**

**Struct student**

**{**

**Char name[10];**

**Int a;**

**Float b;**

**};**

**Void main()**

**{**

**Clrscr();**

**Int x;**

**Struct student stdnt1,stdnt2;**

**Printf(“enter details of student 1”);**

**Scanf(“%S%d%f”,stdnt1.name,&stdnt1.a,&stdnt1.b );**

**Printf(“enter details of student 2”);**

**Scanf(“%S%d%f”,stdnt2.name,&stdnt2.a,&stdnt2.b );**

**Printf(“choose student “);**

**Scanf(“%d”,&x);**

**If(x==1)**

**{**

**Printf(“ details of student 1 is given below”);**

**printf(“student name=%S”,stdnt1.name);**

**Printf(“roll number=%d”,stdnt1.a);**

**Printf(“percentage mark=%f”,stdnt1.b);**

**}**

**Else**

**{**

**Printf(“ details of student 2 is given below”);**

**printf(“student name=%S”,stdnt2.name);**

**Printf(“roll number=%d”,stdnt2.a );**

**Printf(“percentage mark=%f”,stdnt2.b);**

**}**

**Getch();**

**}**

**Note: in the above example structure is defined outside the main() function but structure variables are declared inside main function.**

**Example:program to store and print name and role number of two students**

**#include <stdio.h>**

**#include <conio.h>**

**Void main()**

**{**

**Clrscr();**

**Struct student**

**{**

**Char name[10];**

**Int a;**

**}stdnt1,stdnt2;**

**Scanf(“%S%d”,stdnt1.name,&stdnt1.a,stdnt2.name ,&stdnt2.a);**

**Printf(“%S%d”,stdnt1.name,stdnt1.a,stdnt2.name ,stdnt2.a);**

**Getch();**

**}**

**Array of structures**

if we want to store and process details of two students(details include name,roll number ,percentage mark) we can use structure.in this case we have to declare two structure variables.

If we want to store and process details of 100 students we have to declare 100 structure variables.that is not practical.

In order to overcome this we can use array of structure concept .In this ,we are using arrays to store structure variables.This is possible bcz all structure variables are same data type.

**Syntax:**

**Struct structure name**

**{**

**Data type member 1;**

**Data type member 2;**

**….…..**

**….……**

**….…**

**}array name[array size] ;**

**Syntax:**

**Struct structure name**

**{**

**Data type member 1;**

**Data type member 2;**

**….…..**

**….……**

**….…**

**} ;**

**Struct structure name array name[array size];**

**Eg:**

**#include <stdio.h>**

**#include <conio.h>**

**Void main()**

**{**

**Clrscr();**

**Struct student**

**{**

**Char name[10];**

**Int a;**

**};**

**Struct student max[10];**

**Scanf(“%S%d”,max[0].name,&max[0].a);**

**Printf(“%S%d”,max[0].name,max[0].a);**

**Getch();**

**}**

**In the above example,we declared structure variable to store 10 students details(we used an array for this purpose) .but here we initialized only one structure variable .that is we stored and printed only one student details.**

**Example**: **program to to store and print name,roll number of 10 students.**

**#include <stdio.h>**

**#include <conio.h>**

**Void main()**

**{**

**Clrscr();**

**Struct student**

**{**

**Char name[10];**

**Int a;**

**};**

**Struct student max[10];**

**For(i=0;i<10;i++)**

**{**

**Printf(“enter details of student\n”);**

**Scanf(“%S%d”,max[i].name,&max[i].a);**

**}**

**For(i=0;i<10;i++)**

**{**

**Printf(“ details of student is\n”);**

**printff(“%S%d”,max[i].name,max[i].a);**

**}**

**Getch();**

**}**

**Typedef keyword**

**“Typedef” is a keyword which is using to replace existing keyword of a data type using new one which is created by user .(ie,“typedef” will create a alternate keyword for the existing data type).**

**Synatx :**

**Typedef existing data type new data type;**

**Eg:**

**Typedef int max;**

**If we use this instruction ,we can use keyword “max” to declare integer type variable.(we can use any name instead of max,but it must follow rules regarding keyword naming)**

Consider a program for adding two elements

**#include <stdio.h>**

**#include <conio.h>**

**Void main()**

**{**

**Clrscr();**

**Typedef int eye;**

**Eye x,y,s;**

**Printf(“enter elements\n”);**

**Scanf(“%d%d”,&x,&y);**

**S=x+y;**

**Prnitf(“%d\n”,s);**

**Getch();**

**}**

**Here we used our new keyword “eye” for declaring integer type variables.**

**(note: there is no change in control strings.)**

**union**

Union is same concept of structure,the difference is in keyword using and memory allocation only.rest of all are same as structures

**Pointers**

Pointers are the variables which is using to hold address a variable.

**Pointer declaration**

**Syntax:**

**Data type\* variable name;**

* **Data type** is the type of data present in the variable whose address need to be stored in pointer.
* **“ \* “** called as **value operator/indirection operator/dereferencing operator.**
* **Variable name:** We can give any name to pointer variable,but should follow rules regarding variable naming.

Eg:

**consider a variable declaration**

**Int a;**

**“a” is the variable which can hold integer type data.We can store the address of data present in variable “a” using pointer.for that, firstly we have to declare pointer.**

**Ie,**

**Int \* p;**

**“P “is our pointer,now “p” is ready to store address of data present in variable ” a”.**

**Initialization of pointer**

This is the process of adding address of a variable to the pointer. in order to this,we have to use address operator **“&”.**

**Syntax:**

**Pointer name = & name of variable whose data address need to be stored in pointer.**

**Eg:**

**P=&a;**

**This instruction will store address of data present in variable “a” to pointer p.**

**Accessing of variable data using pointer**

Consider a situation in which pointer is initialized.(ie, we stored the address of a data present in a variable in pointer).

By using the value operator **” \* ”** along with the pointer ,we can access the data stored in that variable.

Eg:

**Consider a program to add two integers.**

**#include <stdio.h>**

**#include <conio.h>**

**Void main()**

**{**

**Clrscr();**

**Int a,b,r;**

**Printf (“enter values\n”);**

**scanf(“%d%d”,&a,&b);**

**r=a+b;**

**Printf(%d\n”,r);**

**Getch();**

**}**

**We can do the same program using pointer also.**

**Ie,**

**#include <stdio.h>**

**#include <conio.h>**

**Void main()**

**{**

**Clrscr();**

**Int a,b,r;**

**Int\*p1,\*p2;**

**Printf (“enter values\n”);**

**scanf(“%d%d”,&a,&b);**

**P1=&a;**

**P2=&b;**

**r=\*p1+\*p2;**

**Printf(%d\n”,r);**

**Getch();**

**}**

**Explanation**

In the above example we used **“ \*p1”** for accessing data in variable **“a”** and **“ \*p2”** for accessing data in variable **“b”**

**Note**

We can perform any kind of operations on variable data using value operator.there is an exceptional case we have to be considered while manipulating data.

ie, if we are using division on data using pointer we have to use bracket, otherwise there is a chance of error.

**Eg:**

**R=\*p1/\*p2;**  this statement can cause confusion in compiler(I’m not mentioning the reason for this issue) .so use the following statement

**R=(\*p1)/(\*p2);** this is the true statement for division using pointers.

**Note**

We can also perform the the following 3 operations on “address “stored in a pointer.

They are subtraction(there is a limitation) ,increment and decrement.in order to do this we want pointer names only.(we don’t have to use a value operator).