**NOTES**

* Structures are user-defined data types and is collection of hetergenous data types. Defined using struct keyword followed by structure name. Semicolons are added to the end.

struct student

{

int age;

Char name[10];

}st1,st2; //way of declaring structure variable

Initialization: student st1={10,”yash”};

* Object of structure can have same name as that of structure.

>> struct name

{

int age;

};

int main(){

struct name name;

cin>>name.age;//is valid

}

>> struct { //valid

int age;

}name;

int main(){

cin>>name.age;//is valid

}

* Defining of structure doesn’t allocates the memory for structure. Just like other data types we need to define structure variable in order to allocate memory for structure. Example:- student s; (struct keyword is not mandatory for declaring structure variable).
* **typedef DECLARATION:**

typedef enables programmers to create a new data type name for an existing data type.By using typedef,no new data is created rather an alternate name is given to existing datatype for better readability:

>>Syntax: typdef existing\_type new\_type

>>Example: typdef int yash;

yash a=5; is same as int a=5;

>>Example: typdef char name[10];

# name name1,name2; same as

# char name1[10] name2[10]

>>name1[3] to access fourth element,name1 to access address of first element

* C doesn’t allow declaration of variable at time of defining typdef so declare variable in another statement;

Typedef int yash a=5; //illegal statement

* All structure declaration in c++ are typedef by default,so using struct keyword at time of defining structure variable is optional. It is used when using structure’s pointer or reference type.

* At time of declaring structure variable,int and float values are initialized to zero and char and string values to ‘\0’ by default;
* Members of structures are accessed using dot operator. Example:- st1.age,st1.name
* ->,.(dot),[] have highest priority of all operators.

Example: st1.age++ is actually (st1.age)++

* C++ doesn’t permit comparison of one variable type with another variable type

struct student{

int age;

}s={10},r;

struct employee{

int age;

char name[10];

}e;

>>// e=s is illegal statement

>> r=s is legal coz they are same type of variables

>>typdef student yash

Yash m=s; //valid

* Nested structures:

>> struct name

{

char f\_name[10];

char l\_name[10];

};

struct age{

int age;

};

struct employee

{ int emp\_no;

struct name n;

struct age a;

};

using namespace std;

int main() {

employee e;

cout<<"employee name:";

cin>>e.n.f\_name;

cout<<"employee age:";

cin>>e.a.age;

cout<<"employee number:";

cin>>e.emp\_no;

}

>>Self refrential structure: self refrential structure contain refrence to data of its same type. In addition,it also contains a pointer to data of same type as that of structure.

Example:-

Struct node{

Int data;

Struct node \*next;}

|  |  |
| --- | --- |
| **Structure** | **Union** |
| 1.The keyword  **struct** is used to define a structure | 1. The keyword union is used to define a union. |
| 2. When a variable is associated with a structure, the compiler allocates the memory for each member. The size of structure is greater than or equal to the sum of  sizes of its members. The smaller members may end with unused slack bytes. | 2. When a variable is associated with a union, the  compiler allocates the  memory by considering the size of the largest memory. So, size of union is equal to the size of largest member. |
| 3. Each member within a structure is assigned unique storage area of location. | 3. Memory allocated is shared by individual members of union. |
| 4. The address of each member will be in ascending order This indicates that memory for each member will start at different offset values. | 4. The address is same for all the members of a union. This indicates that every member begins at the same offset value. |
| 5 Altering the value of a member will not affect other members of the structure. | 5. Altering the value of any of the member will alter other member values. |
| 6. Individual member can be accessed at a time | 6. Only one member can be accessed at a time. |
| 7. Several members of a structure can initialize at once. | 7. Only the one member of a union can be initialized. |