Enumeration (or enum) in C

Enumeration (or enum) is a user defined data type in C. It is mainly used to assign names to integral constants, the names make a program easy to read and maintain.

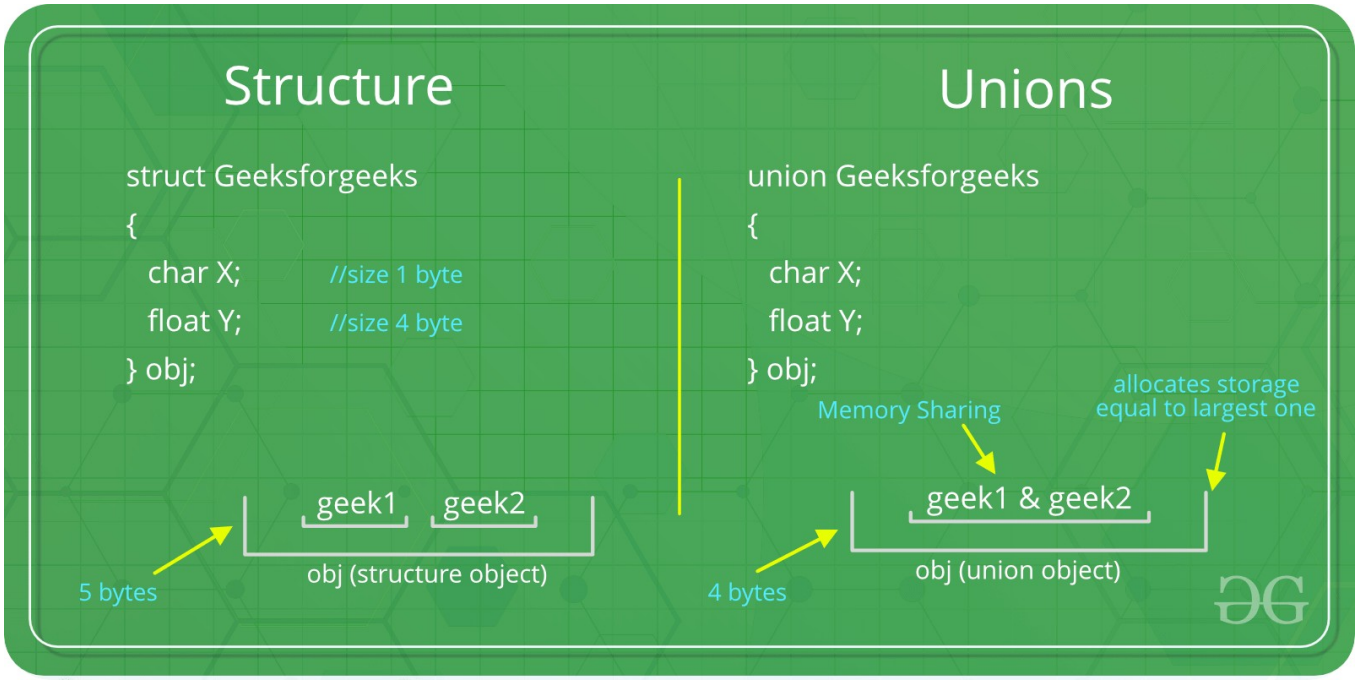
How to initialize structure members?

Structure members cannot be initialized with declaration.

The reason for above error is simple, when a datatype is declared, no memory is allocated for it. Memory is allocated only when variables are created.

Union in C

Like Structures, union is a user defined data type. In union, all members share the same memory location.



How is the size of union decided by compiler?

Size of a union is taken according the size of largest member in union.

If you have to store about a person -- that he is a child, adult or senior citizen then you will prefer a union saving space in the bargain, because you know a person won't be child and adult at the same time.

On the other hand if you are storing his preferences of the type of coffee -- say hot, cold, etc etc -- then its a structure as you might have someone preferring more than one type.

Unions can be used when no more than one member needs to be accessed at a time. That way, you can save some memory instead of using a struct.

Can we use bit fields outside of structure and union in C?

Use of typedef with a structure

When we use typedef with structure then it creates the alias of the structure. There is no need to write struct keyword every time with a variable declaration that means typedef save extra keystroke and make the code cleaner and readable.

struct hack in C

The struct hack technique gives the permission to the user to create a variable length member in the structure.

In struct hack techniques, we need to create an array whose length is 0 (some compiler does not support the 0 size array).When we create a zero size array then structure becomes the incomplete type. Basically, an incomplete type structure is a type that has lack of information about its member.

As I have mentioned above, if we create an incomplete type member in the structure, the structure becomes incomplete types and this technique is called struct hack.

In below structure I am creating a character array to store student name, I am giving the length of the array 0 (some compiler does not support 0 length array, in that scenario we have to take the length of the array 1).

```
typedef struct
{

int RollNumber;

int TotalMarks;

char Name[0];

}sStudentInfo;
```

Why is struct hack required?

Let’s take an example to understand the above question. First I want to declare a structure which contains the information (price, name, expiry date ..etc) of medicine.

```
typedef struct
{
 int Price;

 int ExpiryYears;

 char Name[MaxSize];

}sMedicineInfo;
```

In above structure, the name of the medicine should be dynamic. If the name of the medicine is less then MaxSize, it causes the memory loss but if the medicine name greater then the MaxSize, you could be faced the code crashing issues. With the help of struct hack we can resolve this issues and able to create the variable length array whose size could be changed as per the requirements.

Why not use a pointer?

Using the pointer we can also create the dynamic length array but problem is that in pointers take extra (4 or 8 bytes depending on the system) memory.When we have created a pointer to the structure then we have to explicitly allocate the memory for the pointers but if we used struct hack then there is no need to allocate memory again for the array.

union https://aticleworld.com/	structure
union keyword is used to define the union type.	structure keyword is used to define the union type.
Memory is allocated as per largest member.	Memory is allocated for each member.
All field share the same memory allocation.	Each member have the own independent memory.

We can access only one field at a time.
 	
union Data {
int a; // can't use both a and b at once
char b;
} Data;
 	
union Data x;
x.a = 3; // OK
x.b = 'c'; // NO! this affects the value of x.a
	We can any member any time.
	
	struct Data {
	int a; // can use both a and b simultaneously
	char b;
	} Data;
	
	struct Data y;
	y.a = 3; // OK
	y.b = 'c'; // OK

Altering the value of the member affect the value of the other member.	Altering the value of the member does not affect the value of the other member.
Flexible array is not supported by the union.	Flexible array is supported by the structure.
With the help of union we can check the endianess of the system.	We cannot check the endianess. https://aticleworld.com/

Similarities between Structure and Union

1. Both are user-defined data types used to store data of different types as a single unit.
2. Their members can be objects of any type, including other structures and unions or arrays. A member can also consist of a bit field.
3. Both structures and unions support only assignment = and sizeof operators. The two structures or unions in the assignment must have the same members and member types.
4. A structure or a union can be passed by value to functions and returned by value by functions. The argument must have the same type as the function parameter. A structure or union is passed by value just like a scalar variable as a corresponding parameter.
5. **‘.’** operator is used for accessing members.

Structure Member Alignment, Padding and Data Packing

In the case of structure or union, the compiler inserts some extra bytes between the members of structure or union for the alignment, theses extra unused bytes are called padding bytes and this technique is called padding.

Padding increases the performance of the processor at the penalty of memory. In structure or union data members aligned as per the size of the highest bytes member to prevent the penalty of performance.

Note: *In the case of structure and union we can save the wastage of memory to rearrange the structure members in the order of largest size to smallest.*

Note: *We can change the alignment of structure, union or class using the “pack” pragma directive, but sometimes it becomes a crucial reason for the compatibility issues in your program. So it’s better always use the default packing of the compiler.*

When you create the structure then compiler inserts some extra bytes between the members of structure for the alignment. These extra unused bytes are called padding bytes and this technique is called structure padding in C. A structure padding in C increases the performance of the processor at the penalty of memory.

If you want you can avoid the structure padding in C using the pragma pack (#pragma pack(1)) or attribute (__attribute__((__packed__))).

Bit Fields in C

In C, we can specify size (in bits) of structure and union members. The idea is to use memory efficiently when we know that the value of a field or group of fields will never exceed a limit or is withing a small range.

Following are some interesting facts about bit fields in C.

1) A special unnamed bit field of size 0 is used to force alignment on next boundary.

```
struct test1
{
unsigned int x: 5;
unsigned int y: 8;
};
```

Size is 4 bytes

```
struct test2
{
unsigned int x: 5;
unsigned int: 0;
unsigned int y: 8;
};
```

Size is 8 bytes

2) We cannot have pointers to bit field members as they may not start at a byte boundary.

4) In C++, we can have static members in a structure/class, but bit fields cannot be static.

Array of bit fields is not allowed. For example, the below program fails in compilation.

unsigned int x[10]: 5;