Union

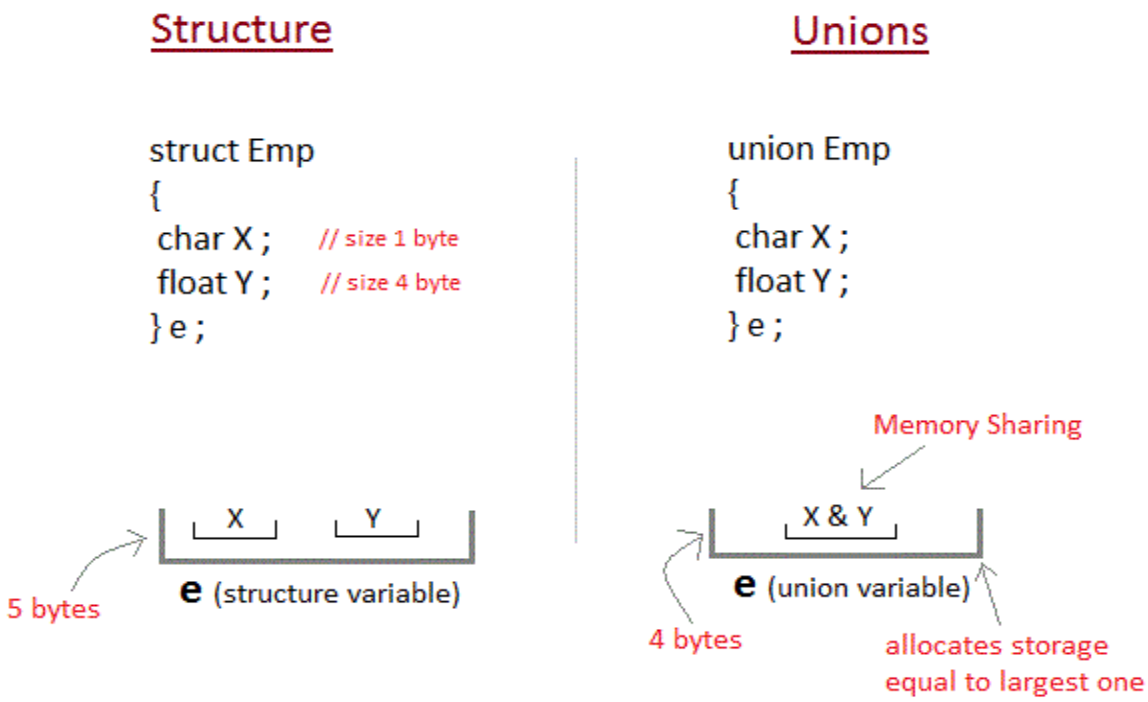
C Union is also like structure, i.e., collection of different data types which are grouped together. Each element in a union is called member.

- Union and structure in C are same in concepts, except allocating memory for their members.
- Structure allocates storage space for all its members separately.
- Whereas, Union allocates one common storage space for all its members

- We can access only one member of union at a time. We can't access all member values at the same time in union. But, structure can access all member values at the same time. This is because, Union allocates one common storage space for all its members. Where as Structure allocates storage space for all its members separately.
- Many union variables can be created in a program and memory will be allocated for each union variable separately.

Note:

- We can access only one member of union at a time. We can't access all member values at the same time in union.
- But, structure can access all member values at the same time. This is because, Union allocates one common storage space for all its members. Where as Structure allocates storage space for all its members separately.



Syntax:

```
union tag_name
{
data type var_name1;
data type var_name2;
data type var_name3;
};
```

Example:

```
union student
{
int  mark;
char name[10];
float average;
};
```

Declaring union using normal variable:

```
union student report;
```

Initializing union using normal variable:

```
union student report = {100, “Mani”, 99.5};
```

Accessing union members using normal variable:

```
report.mark;
report.name;
report.average;
```

DIFFERENCE BETWEEN STRUCTURE AND UNION IN C:

C Structure	C Union
Structure allocates storage space for all its members separately.	Union allocates one common storage space for all its members. Union finds that which of its member needs high storage space over other members and allocates that much space
Structure occupies higher memory space.	Union occupies lower memory space over structure.
We can access all members of structure at a time.	We can access only one member of union at a time.

Structure example:
struct student
{
int mark;
char name[6];
double average;
};
	Union example:
	union student
	{
	int mark;
	char name[6];
	double average;
	};

For above structure, memory allocation will be like below. int mark – 2B char name[6] – 6B double average – 8B Total memory allocation = 2+6+8 = 16 Bytes	For above union, only 8 bytes of memory will be allocated since double data type will occupy maximum space of memory over other data types. Total memory allocation = 8 Bytes

Example:

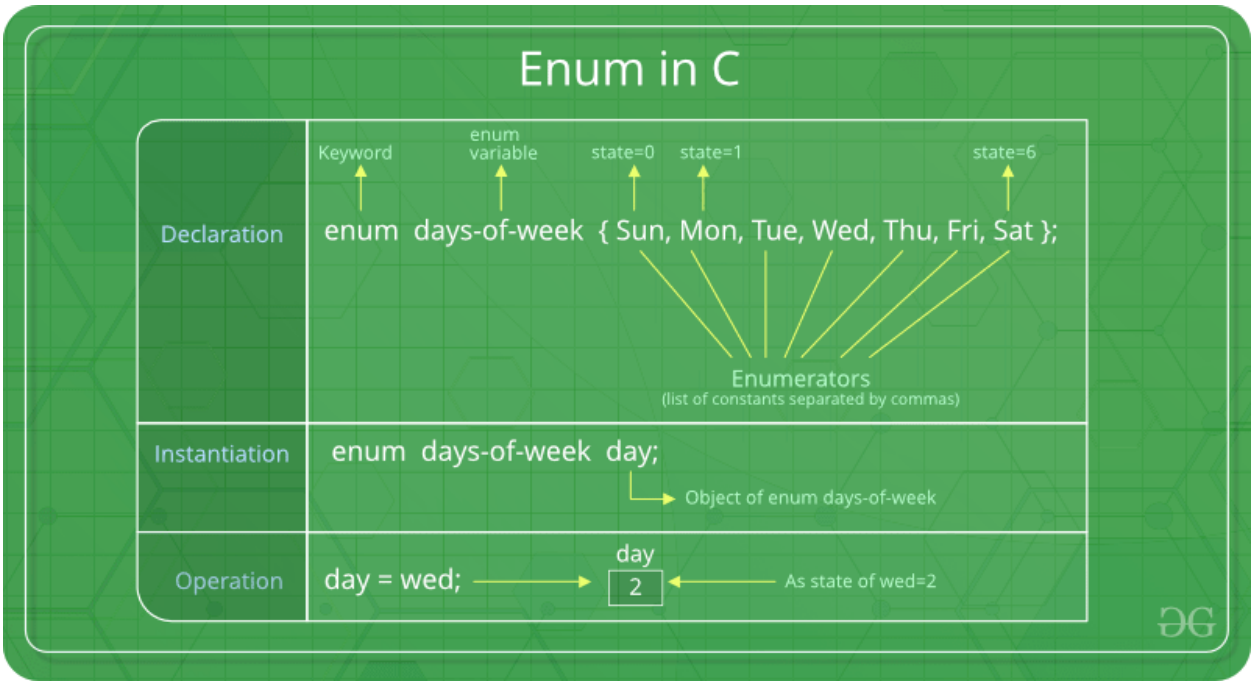
```
#include <conio.h>
 #include <stdio.h>
union item
{
   int a;
   float b;
   char ch[10];
};

void main( )
{
   union item it;
   printf("Enter the item no.:");
   scanf("%d",&it.a);
   printf("Enter the item price:");
   scanf("%f",&it.b);
   printf("Enter the item Name:");
   scanf("%s",&it.ch);
   printf("%d\n", it.a);
   printf("%f\n", it.b);
   printf("%s\n", it.ch);

   getch();
}
```

```
H:\CPROGRAM\NONAME02.exe
Enter the item no.:12
Enter the item price:56.63
Enter the item Name:apple
1819308129
116269156435107020800000000.000000
apple
```

As you can see here, the values of `item number` and `item price` get corrupted and only variable `item name` prints the expected result. This is because in union, the memory is shared among different data types. Hence, the only member whose value is currently stored will have the memory.

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.



```
#include <stdio.h>
#include <conio.h>
enum week {Sunday, Monday, Tuesday, Wednesday, Thursday, Friday, Saturday};

void main()
{
   // creating today variable of enum week type
   enum week today;
   today = Wednesday;
   printf("Day %d",today+1);
   getch();
}
```