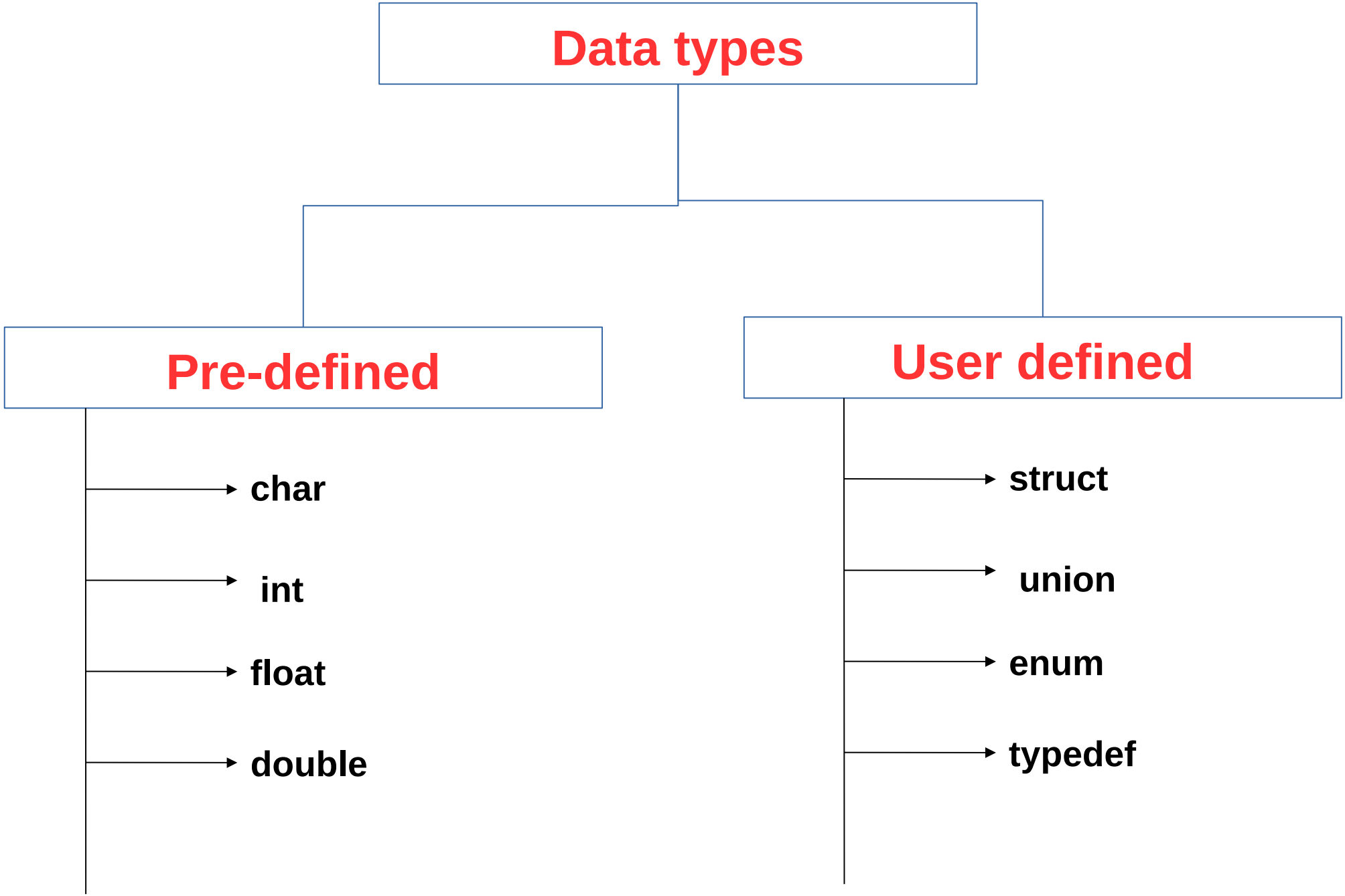
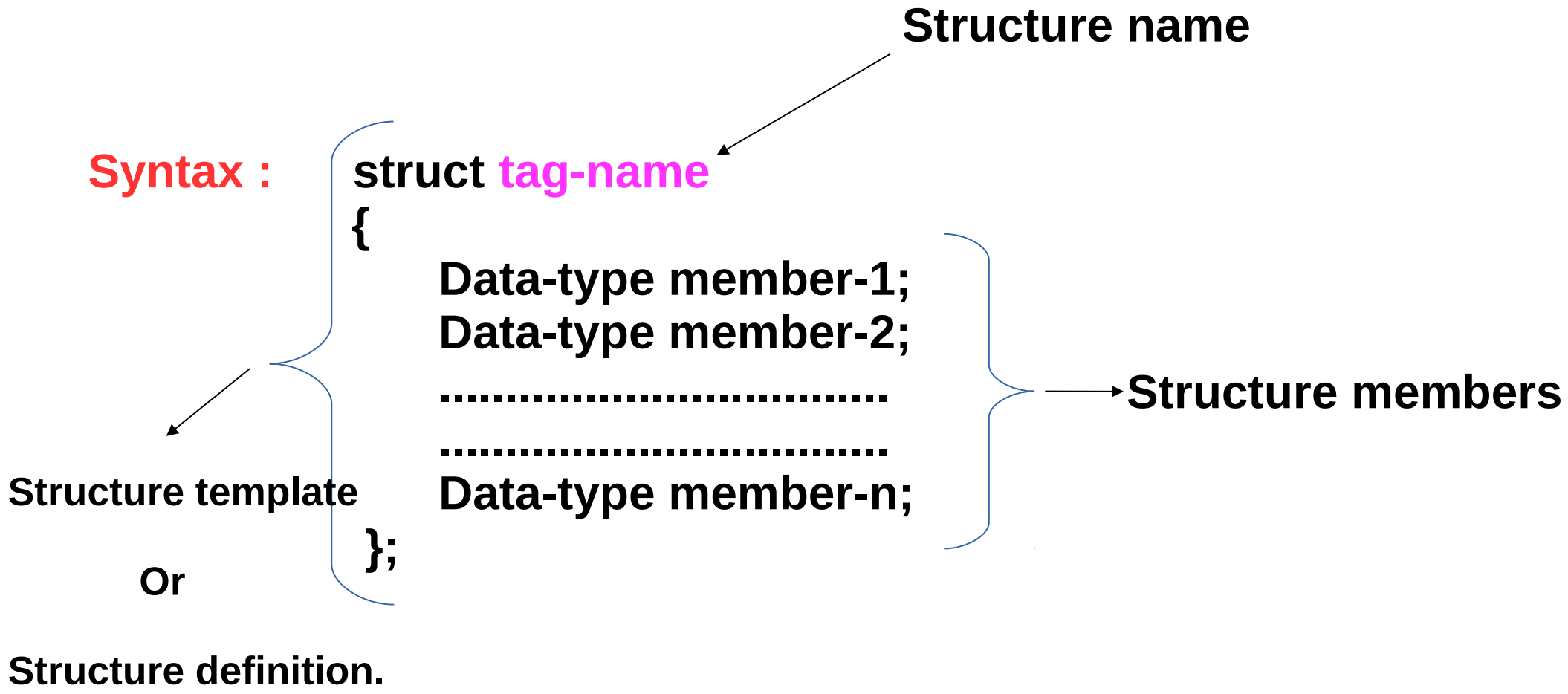

Data types
Pre-defined
User defined
char
int
float
double
struct
union
enum
typedef

structure

It is a collection of different data items which are stored in contigious memory location.



Note : structure tag-name is optional.

--> for structure definitions there is no memory is allocated.
--> structure memory is basically allocated when a structure variable is declared.

Q) How to declare a structure variable?
A) in 2 ways it is possible to declare a structure varible.

1. along with structure definition.
2. using structure tag-name.

1. along with structure definition :

```
struct tag-name
{
    ......
    ......
}var1, var2,var3...;
```

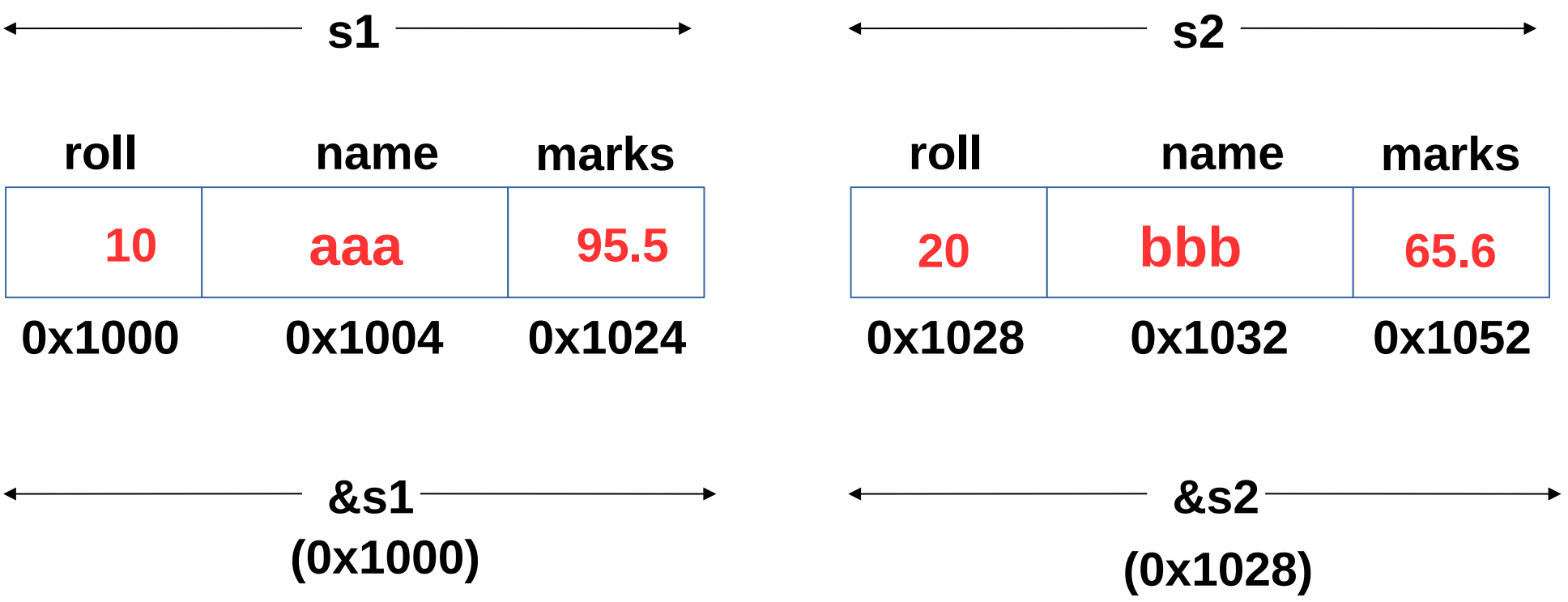
2. using structure tag-name :

```
struct tag-name
{
    ......
    ......
};
```

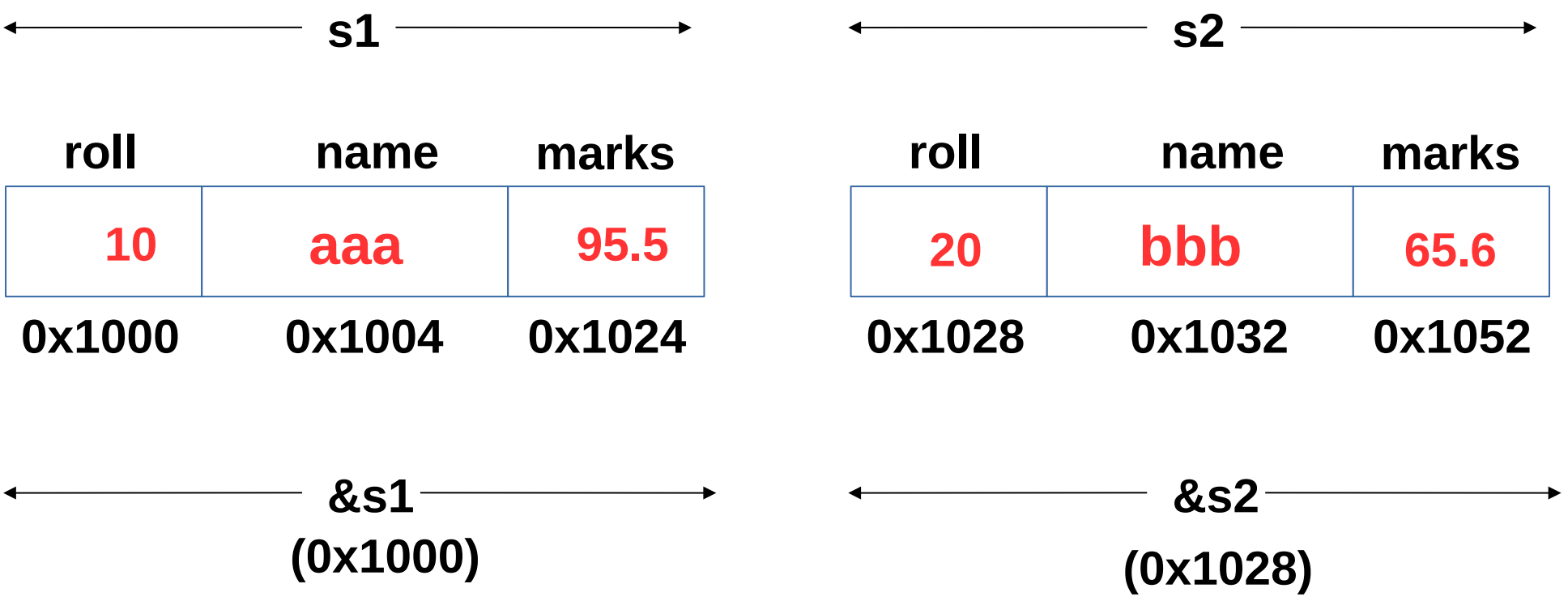
```
struct tag-name var1,var2,var3...;
```

A student record ---> roll, name & marks

```
struct student
{
      int roll;
      char name[20];
      float marks;
}s1 = {10,”aaa”,95.5} , s2 = {20,”bbb”,65.6};
```



```
struct student
{
    int roll;
    char name[20];
    float marks;
}s1,s2;

struct student s1 = {10,"aaa",95.5} , s2 = {20,"bbb",65.6};
```



Q) How to access the structure members?

A) using 2 operators.

1) using . (dot) operator
2) using -> (arrow) operator

Use .(dot) operator when you want to access the structure members from structure variable.

Use ->(arrow) operator when you want to access the structure members from structure variable address.

s1 variable members access.

s1.roll	or	(&s1)->roll
s1.name	or	(&s1)->name
s1.marks	or	(&s1)->marks

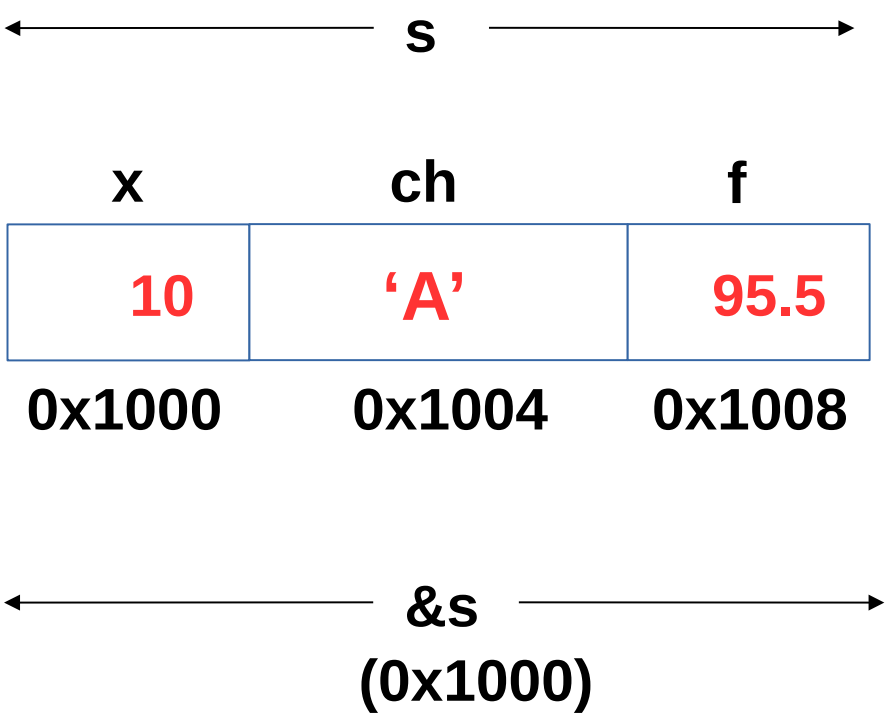
s2 variable members access.

s2.roll	or	(&s2)->roll
s2.name	or	(&s2)->name
s2.marks	or	(&s2)->marks

Note : 1. structure tag-name is optional if structure variable is declared along with structure definition.

2. structue tag-name is must, if structure variable need to be declared after structure definition.

```
struct st
{
    int x;
    char ch;
    float f;
};

struct st s;
```



Data access format :

s.x	or	(&s)->x
s.ch	or	(&s)->ch
s.f	or	(&s)->f

Address access format :

&s.x	or	&(&s)->x
&s.ch	or	&(&s)->ch
&s.f	or	&(&s)->f

Example 1 :

```
1 #include<stdio.h>
2 struct student
3 {
4         int roll;
5         char name[20];
6         float marks;
7 }s1 = {10,"aaa",99.5},s2 = {20,"bbb",65.6};
8 int main()
9 {
10         printf("s1 data...\n");
11         printf("roll - %d   %d\n",s1.roll,(&s1)->roll);
12         printf("name - %s  %s\n",s1.name,(&s1)->name);
13         printf("marks - %f  %f\n",s1.marks,(&s1)->marks);
14
15         printf("s2 data...\n");
16         printf("roll - %d   %d\n",s2.roll,(&s2)->roll);
17         printf("name - %s  %s\n",s2.name,(&s2)->name);
18         printf("marks - %f  %f\n",s2.marks,(&s2)->marks);
19 }
```

Example 2 :

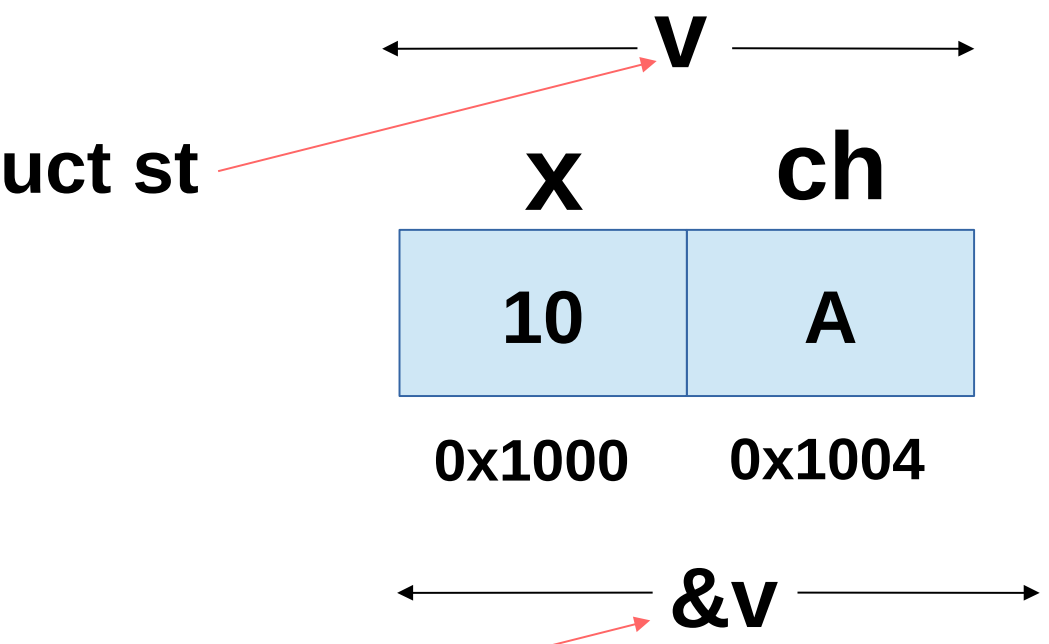
```
#include<stdio.h>
struct student
{
        int roll;
        char name[20];
        float marks;
};
int main()
{
        struct student s1 = {10,"aaa",99.5},s2 = {20,"bbb",65.6};
        printf("s1 data...\n");
        printf("roll - %d   %d\n",s1.roll,(&s1)->roll);
        printf("name - %s  %s\n",s1.name,(&s1)->name);
        printf("marks - %f  %f\n",s1.marks,(&s1)->marks);

        printf("s2 data...\n");
        printf("roll - %d   %d\n",s2.roll,(&s2)->roll);
        printf("name - %s  %s\n",s2.name,(&s2)->name);
        printf("marks - %f  %f\n",s2.marks,(&s2)->marks);
}
```

Example 3 :

```
#include<stdio.h>
int main()
{
        struct
        {
                int x;
                char ch;
                float f;
        }s = {10,'A',4.5};

  printf("%d %c %f\n",s.x,s.ch,s.f); //data of members
  printf("%d %c %f\n",(&s)->x,(&s)->ch,(&s)->f); //data of members

  printf("%p %p %p\n",&s.x,&s.ch,&s.f); //addr of members
  printf("%p %p %p\n",&(&s)->x,&(&s)->ch,&(&s)->f);//addr of members
}
```

```
struct st
{
        int x;
        char ch;
};
```

```
struct st v = {10,'A'};
```



(syntatical datatype representation)

Example 4 :

write a program to read a structure variable data at runtime and display it.

```
#include<stdio.h>
struct st
{
        int x;
        char ch;
};
int main()
{
        struct  st v;
        printf("Enter the x & ch values\n");
        scanf("%d %c",&v.x,&v.ch);

        printf("v.x = %d  v.ch = %c\n",v.x,v.ch);
}
```

//write a program to design the functions to read and print the data for structure variable.

```
#include<stdio.h>
struct st
{
	int x;
	char ch;
};
int main()
{
	struct st v;
	struct st *p = &v;
/*
	printf("Enter the x,ch values\n");
	scanf("%d %c",&p->x,&p->ch);
	printf("print --> %d %c\n",p->x,p->ch);
*/
	printf("Enter the x,ch values\n");
	scanf("%d %c",&(*p).x,&(*p).ch);
	printf("print --> %d %c\n",(*p).x,(*p).ch);

}
```

Example 5 : //write a program to scan the data to a structure variable and display it.

```
#include<stdio.h>
int main()
{
	struct st
	{
		int x;
		char ch;
		float f;
	};

	struct st s;
	printf("Enter the structure data 1)int 2)char 3)float\n");
	scanf("%d %c%f",&s.x,&s.ch,&s.f);

	printf("x - %d  ch - %c  f - %f\n",s.x,s.ch,s.f);
}
```

Example 6 :

```
//write a program to scan the data to a structure variable and display it.
#include<stdio.h>
int main()
{
	struct st
	{
		int x;
		char ch;
		float f;
	};

	struct st s;
	printf("Enter the structure data 1)int 2)char 3)float\n");
	//scanf("%d %c%f",&s.x,&s.ch,&s.f);
	scanf("%d %c%f",&(&s)->x,&(&s)->ch,&(&s)->f);

	//printf("x - %d  ch - %c  f - %f\n",s.x,s.ch,s.f);
	printf("x - %d  ch - %c  f - %f\n",(&s)->x,(&s)->ch,(&s)->f);
}
```

Example 7 :

```
#include<stdio.h>
void fun();
int main()
{
	struct st
	{
		int x;
		char ch;
	};

	fun();
}
void fun()
{
	struct st v = {10,'A'}; //error
}
```

Example 8 :

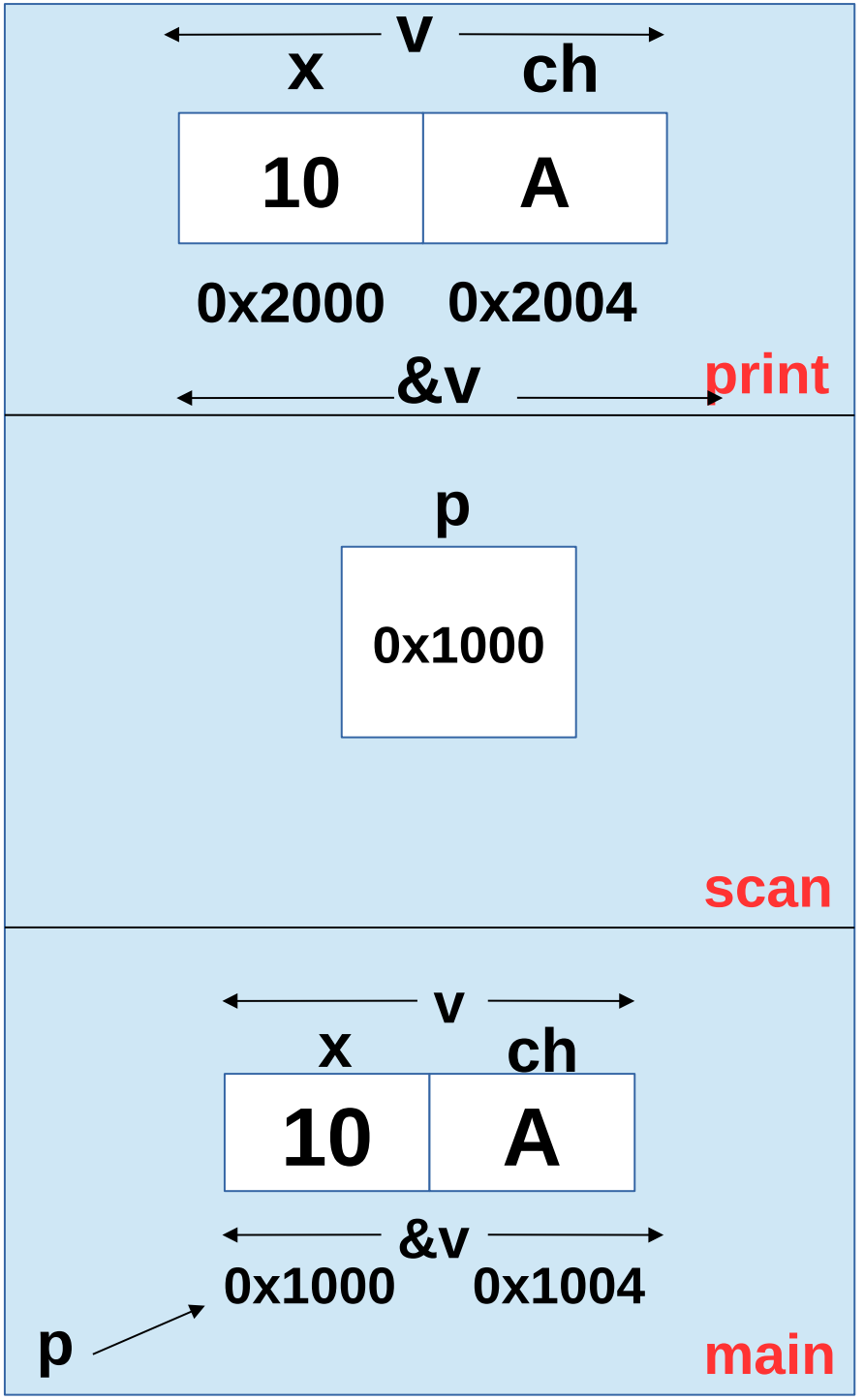
```
#include<stdio.h>
struct st
{
	int x;
	char ch;
};
void fun();
int main()
{
	fun();
}
void fun()
{
	struct st v = {10,'A'}; //no error
	printf("%d %c\n",v.x,v.ch);
}
```

Example 9 :

```
//write a program to pass a structure variable to a function.
#include<stdio.h>
struct st
{
	int x;
	char ch;
};
void fun(struct st);
int main()
{
	struct st v = {10,'A'};
	fun(v);
}
void fun(struct st v)
{
	printf("in fun(), %d %c\n",v.x,v.ch);
}
```

Example 10: WAP to design the functions scan() and print() to scan and print the struct variable data.

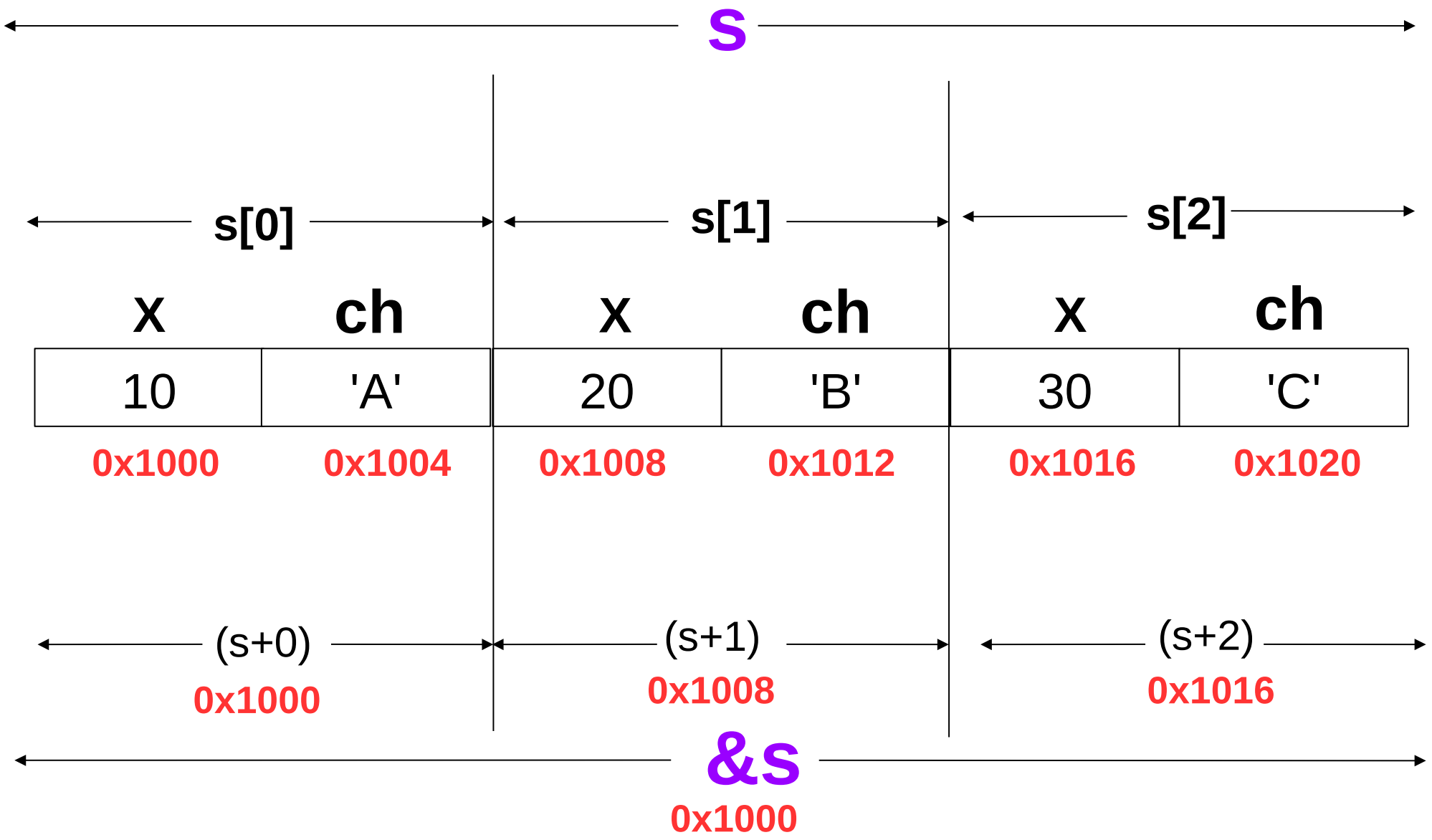
```
#include<stdio.h>
struct st
{
        int x;
        char ch;
};
void scan(struct st *);
void print(struct st);
int main()
{
        struct st v;
        scan(&v);
        print(v);
}
void scan(struct st *p)
{
        printf("enter the data 1)int 2)char\n");
        scanf("%d %c",&p->x,&p->ch);
}
void print(struct st v)
{
        printf("v.x = %d  v.ch = %c\n",v.x,v.ch);
}
```



Structure Array :

Example 11:

```
#include<stdio.h>
struct st
{
        int x;
        char ch;
};
int main()
{
        struct st s[3] = {{10,'A'},{20,'B'},{30,'C'}};
        int i;
        for(i=0;i<3;i++)
        printf("%d %c\n",s[i].x,s[i].ch);
}
```


s
s[0]
s[1]
s[2]
x
ch
x
ch
x
ch
10
'A'
20
'B'
30
'C'
0x1000
0x1004
0x1008
0x1012
0x1016
0x1020
(s+0)
0x1000
(s+1)
0x1008
(s+2)
0x1016
&s
0x1000

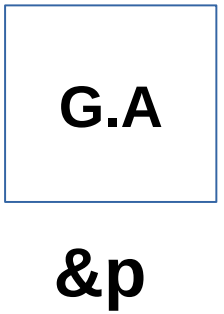
Example 12:

```
#include<stdio.h>
struct st
{
        int x;
        char ch;
};
int main()
{
        struct st s[3];
        int i;
        printf("Enter the 3 structure data\n");
        for(i=0;i<3;i++)
        scanf("%d %c",&s[i].x,&s[i].ch);

        for(i=0;i<3;i++)
        printf("%d %c\n",s[i].x,s[i].ch);
}
```

Example 13:

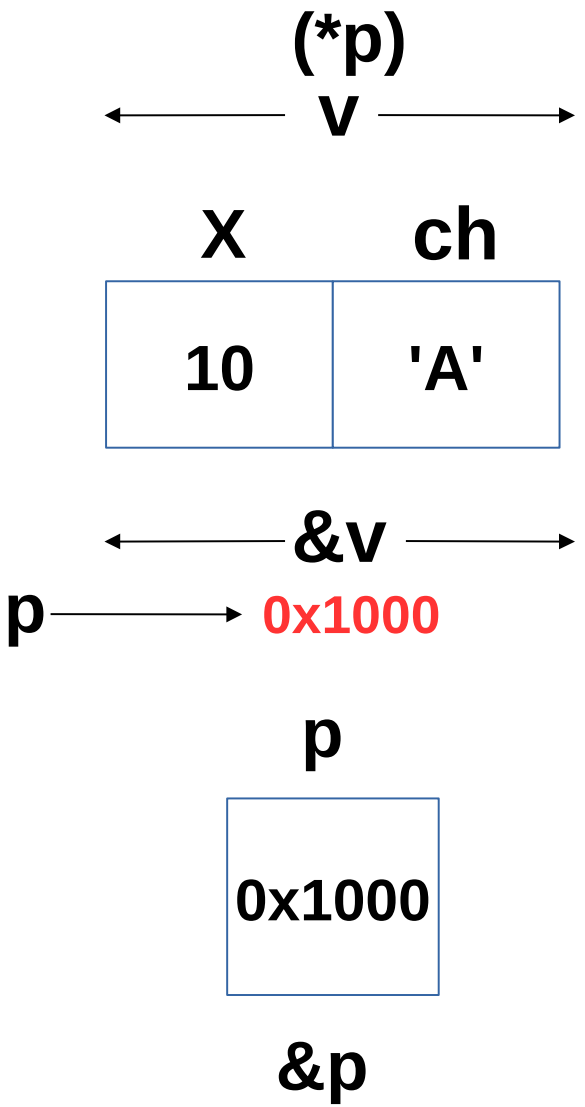
```
#include<stdio.h>
struct st
{
        int x;
        char ch;
};
int main()
{
        struct st *p = {10,'A'};
        printf("%d %c\n",p->x,p->ch);
}
```

(G.A)->x, (G.A)->ch



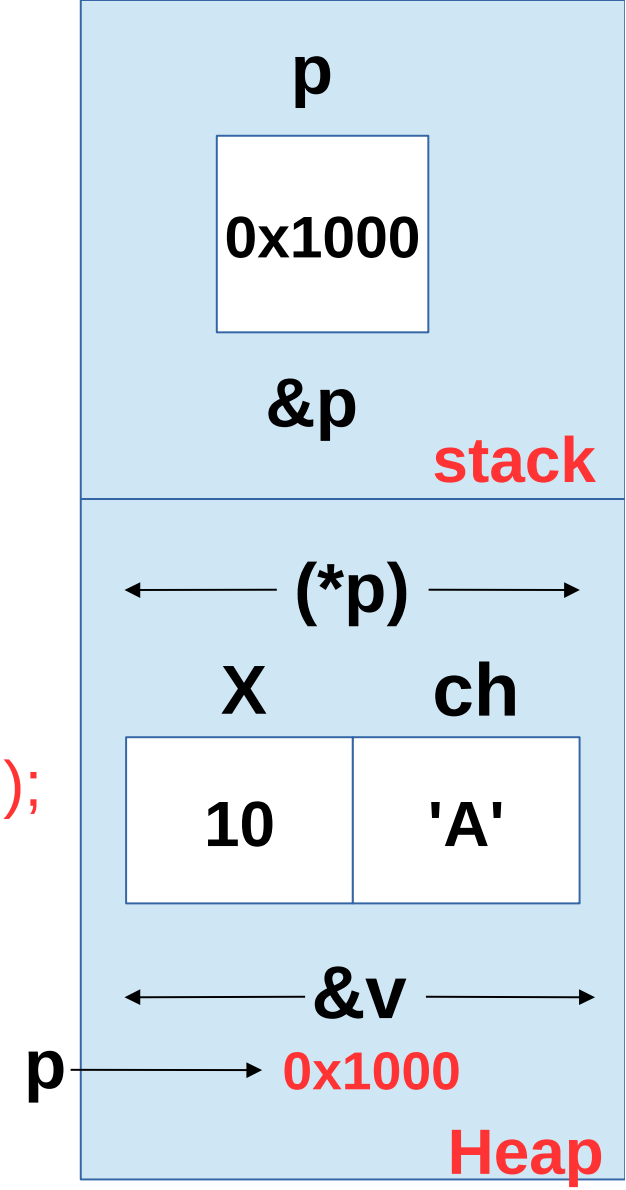
Example 14:

```
#include<stdio.h>
struct st
{
        int x;
        char ch;
};
int main()
{
        struct st v = {10,'A'};
        struct st *p = &v;
        printf("%d  %c\n",p->x,p->ch);
        printf("%d %c\n",(*p).x,(*p).ch);
}
```



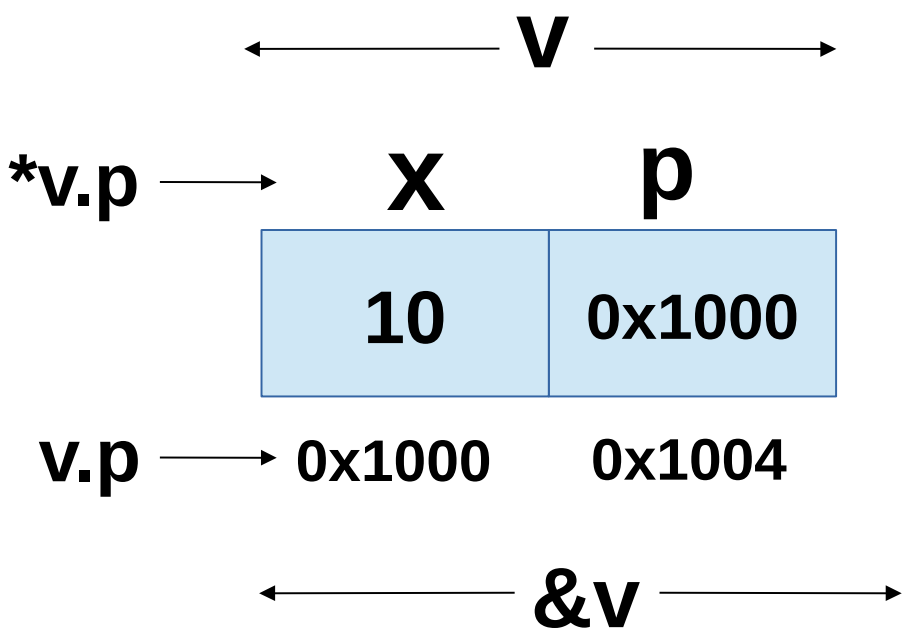
Example 15:

```
#include<stdio.h>
#include<stdlib.h>
struct st
{
        int x;
        char ch;
};
int main()
{
        struct st *p = (struct st *)malloc(sizeof(struct st));
/*
        p->x = 10;
        p->ch = 'A';
*/
        printf("Enter the x,ch data\n");
        scanf("%d %c",&p->x,&p->ch);

        printf("%d  %c\n",p->x,p->ch);
        printf("%d  %c\n",(*p).x,(*p).ch);
}
```



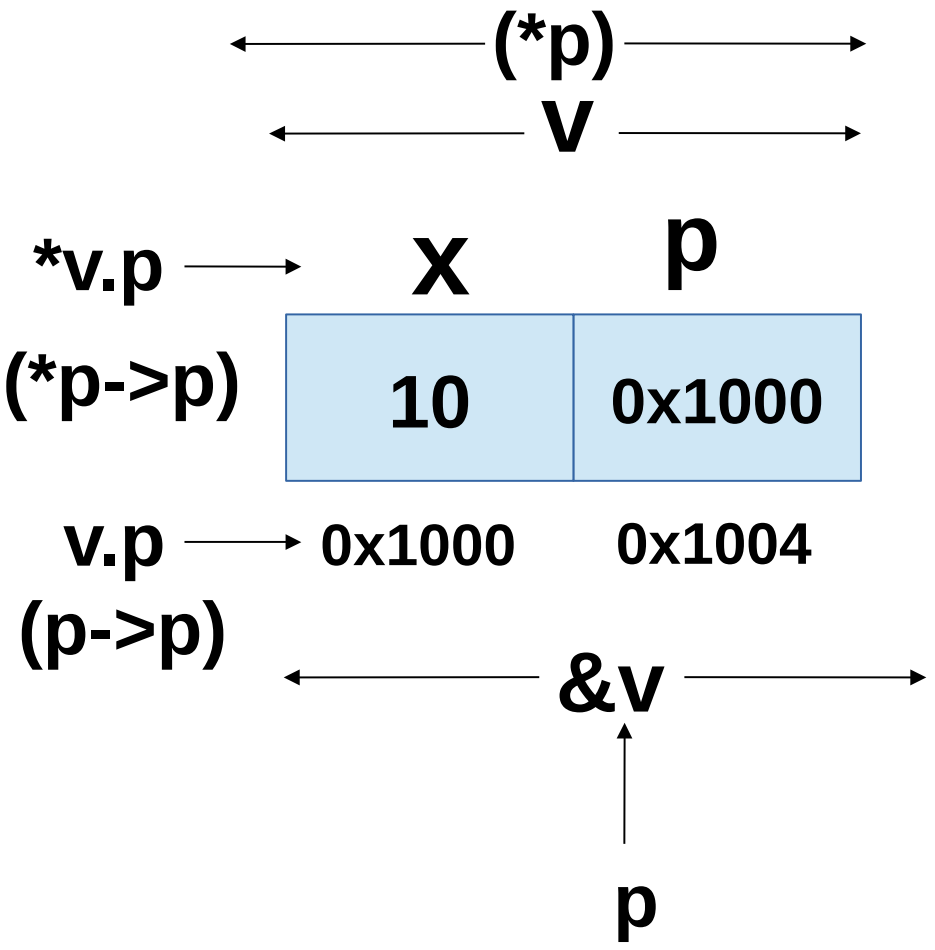
Example 16:

```
#include<stdio.h>
struct st
{
        int x;
        int *p;
};
int main()
{
   struct st v = {10,&v.x};

   printf("&v.x = %p\n",&v.x);
   printf("v.x = %d  v.p = %p\n",v.x,v.p);
   printf("*v.p = %d\n",*v.p);
}
```



Example 17:

```
#include<stdio.h>
struct st
{
        int x;
        int *p;
};
int main()
{
   struct st v = {10,&v.x};
   struct st *p = &v;

   printf("&v.x = %p\n",&v.x);
   printf("%d  %p\n",p->x,p->p);
   printf("%d %d\n",*p->p,*((*p).p));
}
```



Example 18:

```
#include<stdio.h>
struct st
{
        int x;
        struct st v; //invalid
        char ch;
};
int main()
{

}
//Note : A structure should not contain same structure variable as its member.
//A structure variable memory is unable to allocate, if the structure definition is incomplete.
```

Example 19:

```
#include<stdio.h>
struct st
{
        int x;
        struct st *p;//self-referential structure pointer.
        char ch;
};
int main()
{

}
//If a structure containes same structure pointer variableas its member is called as self-referential structure pointer.
```

Example 20:

```
#include<stdio.h>
struct st
{
        auto int x;   //invalid
        static int y; //invalid
        register z;  //invalid
};
int main()
{
}
//Note : storage class can be provided only for structure variables but not for struture members.
```

Example 21:

```
#include<stdio.h>
struct st
{
        auto int x;   //invalid
        auto int y; //invalid
        auto int z;  //invalid
};
int main()
{
}
//Note : storage class can be provided only for structure variables but not for struture members.
```

Example 22:

```
#include<stdio.h>
struct st
{
        int x;   //invalid
        int y; //invalid
        int z;  //invalid
};
int main()
{
        auto struct st v1;
        static struct st v2;
        register struct st v3;
}
```

Example 23:

```
#include<stdio.h>
struct st1
{
	int x;
	int y;
};
struct st2
{
	int a;
	int b;
};
int main()
{
	struct st1 v1;
	struct st2 v2 = {10,20};
	v1 = v2; //invalid
}
//Note : Assignment b/w 2 different structure variable is invalid.
```

Example 24:

```
#include<stdio.h>
struct st
{
        int x;
        int y;
};
int main()
{
        struct st v1,v2 = {10,20};
        v1 = v2; //valid
}
//Note : Assignment b/w same structure variable is valid.
```

Example 25:

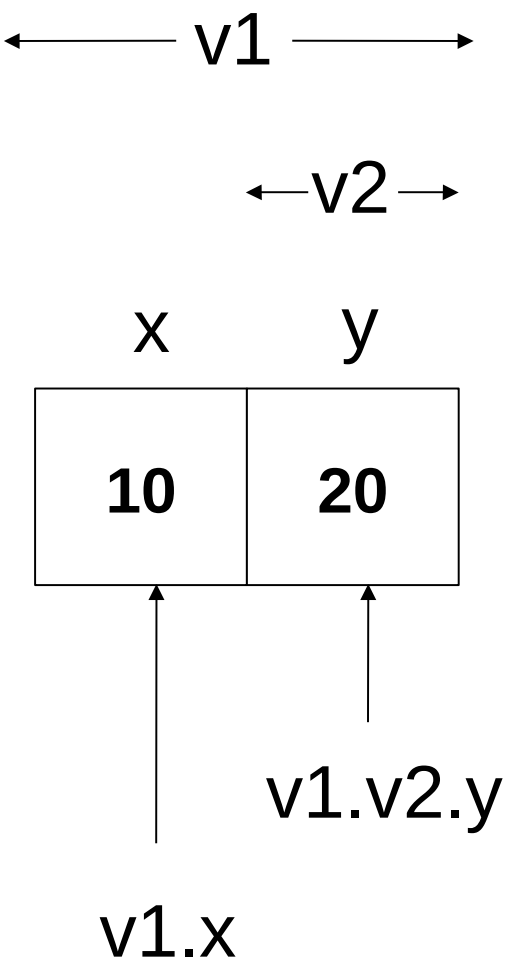
```
#include<stdio.h>
struct st
{
        int x;
        int y;
};
int main()
{
        struct st v1 = {10,20},v2 = {11,22},v3;
        //v3 = v1 + v2; //invalid
        v3.x = v1.x + v2.x;
        v3.y = v1.y + v2.y;
        printf("%d %d\n",v3.x,v3.y);
}
/*
Note : Only some of the operators can perform its operations directly on structure variable like (., -> , *, &, [], sizeof )
*/
```

Example 26:

```
#include<stdio.h>
struct st
{
	int x = 10; //invalid
	int y = 20;
};
int main()
{
	struct st v;
}
/*
structure members should not be initialized inside the structure definition. Because there is no memory is allocated

When the memory is allocated to a structure?
A. after declaration of a structure variable.
*/
```

Example 27:

```
#include<stdio.h>
struct st1                      //struct st2
{                               //{
        int x;                  //   int y;
        struct st2              // };
        {                       // struct st1
                int y;          //  {
        }v2;                    //       int x;
}v1 = {10, {20} };              //       struct st2 v2;
int main()                      //   }v1;
{
        printf("%d %d\n",v1.x,v1.v2.y);
}
```



Example 28:

```
#include<stdio.h>
struct st1                          //struct st2
{                                   //{
        int x;                     //      int y;
        struct st2                  //};
        {                           //struct st1
                int y;              //{
        }v2 = {20};//error          //    int x;
}v1 = {10};                           // struct st1 v2 = {20};
int main()                          //}v1 = {10};
{

}
```

Example 29:

```
#include<stdio.h>
struct st1
{
        int x;
        struct st2
        {
                int y;
                struct st3
                {
                        int z;
                }v3;
        }v2;
}v1 = {10,{20,{30}}};
int main()
{
        printf("v1.x = %d\n",v1.x);
        printf("v1.v2.y = %d\n",v1.v2.y);
        printf("v1.v2.v3.z = %d\n",v1.v2.v3.z);

        printf("%d\n",v2.y);
}
```

```
/*struct st3
 {
        int z;
 };
 struct st2
 {
        int y;
        struct st3 v3;
 };
 struct st1
 {
        int x;
        struct st2 v2;
 }v1 = {10,{20,{30}}};
*/
```

x --> v1.x
y --> v1.v2.y
z --> v1.v2.v3.z