Unions

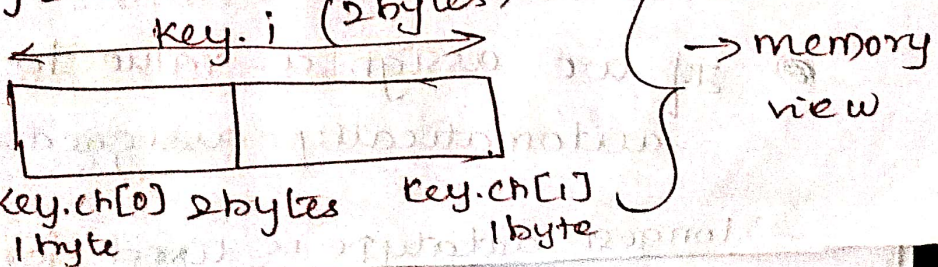
Unions are derived data types that enables us to treat the same memory space as a number of different variables.

ex)

```
#include <stdio.h>
union u
{
    short int i;
    char ch[2];
};
union u key;
void main()
{
    key.i = 512;
    Printf("key.i = %d", key.i);
    Printf("key.ch[0] = %d\n", key.ch[0]);
    Printf("key.ch[1] = %d\n", key.ch[1]);
}
```

O/P:

```
key.i = 512
key.ch[0] = 0
key.ch[1] = 2
```



Key occupies 2 bytes of memory

key.i → 2 bytes (access together)

key.ch[0] & key.ch[1] → 2 bytes (access individually)

512 binary equivalent → ?

(16 bits)

00000010 00000000

2 0

key.ch[0] key.ch[1]

$2^9 = 512$

bout o/p

↳ is reverse. CPUs follow little endian architecture when 2-byte member is stored in memory,

Lower byte is stored before the higher byte as [00000000 00000010]

↓

converted to decimal prints 0 & 2

In big-endian arch → reversal of bytes doesn't happen.

Note:

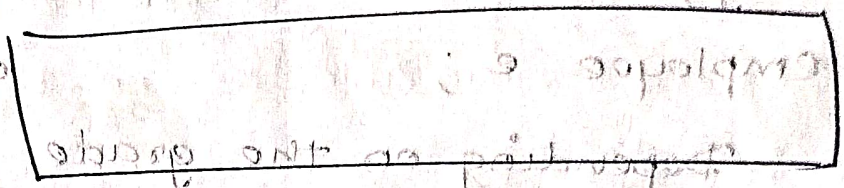
1) we cannot assign different values to different union members.

If we assign a value to key.i → it automatically assigned to key.ch[0] & key.ch[1].

2) longest datatype is used for allocating mem.

ex 2:)

```
union b
{
    double d;
    float f[2];
    short int i[4];
    char ch[8];
};
union b data;
```



8 bytes

data.d → complete 8 bytes

data.f[0] & data.f[1] → 2 set of 4 bytes

data.i[0] ... data.i[3] → 4 set of 2 bytes

data.ch[0] ... data.ch[7] → 8 set of 1 bytes

Size of the union = size of longest element in the union.

Utility of unions

Consider a scenario, i need to store information of employee as

Name, grade, age

if grade = HSK - hobby, credit card No.

if grade = SSK

↓

store vehicle No, distance from the company.

If we create a structure,

```
struct employee
{
    char n[20]; char grade[4]; int age;
    char hobby[10]; int crcardno;
    char vehno[10]; int dist;
};
struct employee e;
```

Depending on the grade only hobby/ credit

vehno/ dist

↓

anyone is only used.

→ Both dont get used leading to wastage of memory with every structure var, we create.

Soln:

can be avoided by creating a union between these elements.

```
struct info1
{
    char hobby[10];
    int crcardno;
};
struct info2
{
```

```
    char rollno[10]; int dist;
};
union info
{
    struct info1 a;
    struct info2 b;
};
struct employee
{
    char n[20];
    char grade[4];
    int age;
    union info f;
};
struct employee e;
```

Files

when data is large → cannot store everything in memory as it has limited space

→ also memory is volatile, contents would be lost once the program is terminated.

→ Hence store the data in files on disk so that it can be later retrieved & used.

2 types

- Text files
- Binary files.