struct bit fields

- Bit fields allow specifying exact number of bits required to store a member
- The member has to be either an integer or an enum type

```cpp
#include <iostream>
using namespace std;

enum days{
    sunday, monday, tuesday, wednesday, thursday, friday, saturday
};
struct date{
    days weekday:3;      // weekday upto 8 can be stored,
    unsigned dd:5;       // Day upto 32 can be stored,
    unsigned mm:4;       // Month upto 16 can be stored,
    unsigned yy:11;      // Year upto 2048 can be stored.

};
int main(){

    date d1;
    d1.weekday = wednesday;
    d1.dd = 28;
    d1.mm = 8;
    d1.yy = 2019;

    cout << "Today is " << d1.weekday << " day of the week and "
         << " it is " << d1.dd << "/" << d1.mm << "/" << d1.yy
         << endl;
}
```

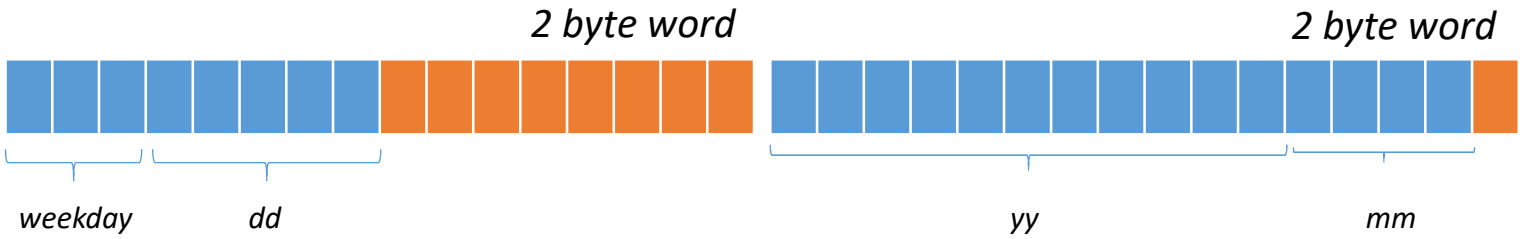
```
G:\CHN-103\Lxx_typedef,enum,struct,union,bitset\structBitfield.exe
Today is 3 day of the week and  it is 28/8/2019

Process returned 0 (0x0)   execution time : 0.094 s
Press any key to continue.
```

struct bit fields

- Unnamed bit fields may be used as padding or to align to storage boundaries.

```
 7  struct date{
 8      days        weekday :3;     // weekday upto 8 can be stored,
 9      unsigned    dd      :5;     // Day upto 32 can be stored,
10      unsigned            :8;     // may help with 2 word (byte) alignment
11      unsigned    yy      :11;    // Year upto 2048 can be stored.
12      unsigned    mm      :4;     // Month upto 16 can be stored,
13      unsigned            :1;     // may help with 2 word (byte) alignment
14  };
```




```
 7  struct date{
 8      days        weekday :3;     // weekday upto 8 can be stored,
 9      unsigned    dd      :5;     // Day upto 32 can be stored,
10      unsigned            :0;     // may help with 2 word (byte) alignment
11      unsigned    yy      :11;    // Year upto 2048 can be stored.
12      unsigned    mm      :4;     // Month upto 16 can be stored,
13      unsigned            :0;     // may help with 2 word (byte) alignment
14  };
```

union datatype

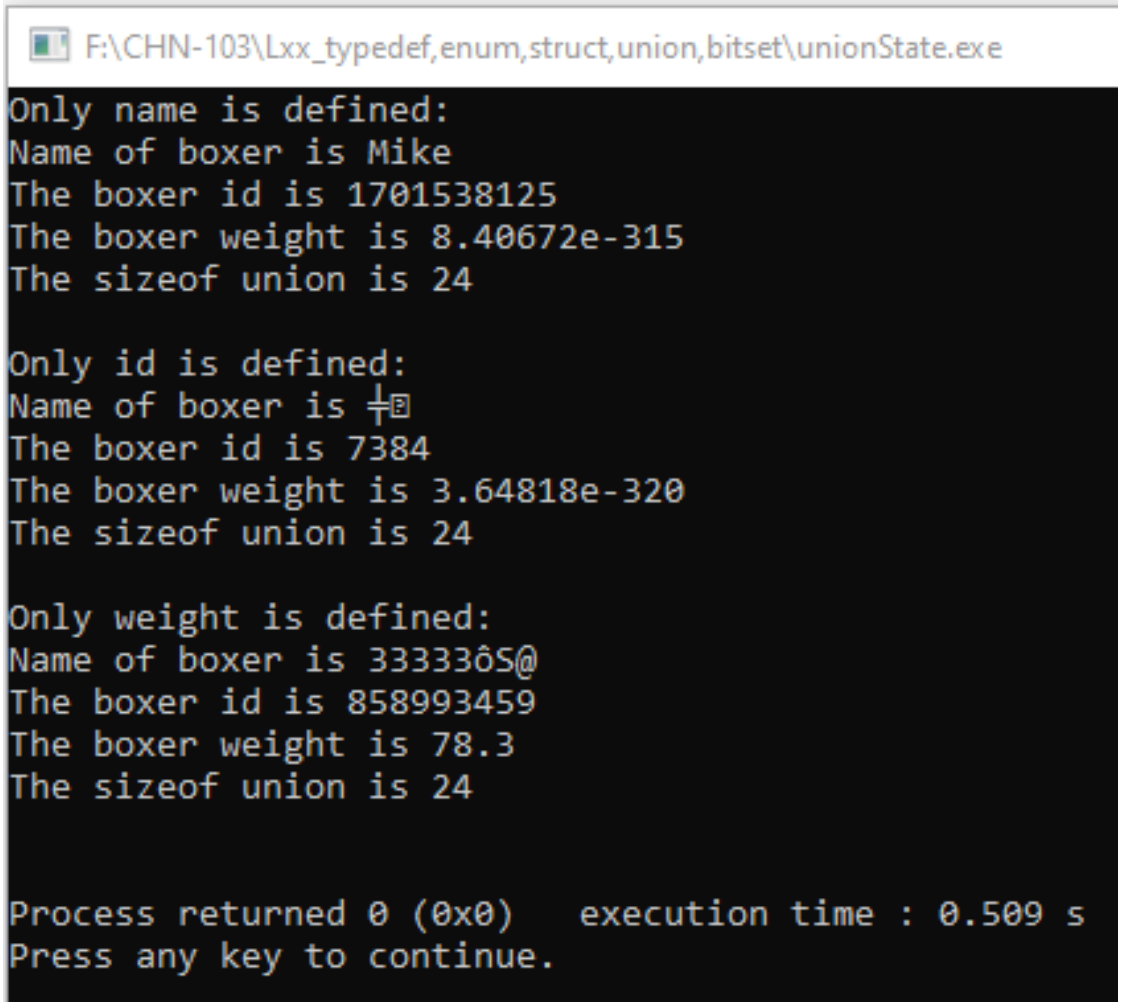
- Unions are similar to *struct* datatype, however only one data member is active at a time.
- Most recently assigned is active, rest are not active.
- Size of union is equal to largest data member.
- Unions can also be implemented like *class* with member functions however all members are public by default like *struct*.

```
1  #include <iostream>
2  using namespace std;
3
4  union dtype{
5      char    name[20];    // will occupy 20 bytes
6      long    id;          // will occupy 4 bytes
7      double  weight;      // will occupy 8 bytes
8  };
```

union datatype

```
10  int main(){
11
12      cout << "Only name is defined:" << endl;
13      dtype boxer = {"Mike"};
14      cout << "Name of boxer is " << boxer.name << endl;
15      cout << "The boxer id is " << boxer.id << endl;
16      cout << "The boxer weight is " << boxer.weight << endl;
17      cout << "The sizeof union is " << sizeof(boxer) << endl << endl;
18
19      cout << "Only id is defined:" << endl;
20      boxer.id = 7384;
21      cout << "Name of boxer is " << boxer.name << endl;
22      cout << "The boxer id is " << boxer.id << endl;
23      cout << "The boxer weight is " << boxer.weight << endl;
24      cout << "The sizeof union is " << sizeof(boxer) << endl << endl;
25
26      cout << "Only weight is defined:" << endl;
27      boxer.weight = 78.3;
28      cout << "Name of boxer is " << boxer.name << endl;
29      cout << "The boxer id is " << boxer.id << endl;
30      cout << "The boxer weight is " << boxer.weight << endl;
31      cout << "The sizeof union is " << sizeof(boxer) << endl << endl;
32
33  }
```

union datatype



union datatype - anonymous

```
1  #include <iostream>
2
3  using namespace std;
4
5  int main(){
6
7      union{
8          char name[20];
9          double dd;
10     };
11     cin.getline(name,20);
12     dd = 3.24e-90;
13     cout << "This union contains " << name << " and " << dd << endl;
14 }
```

By default the anonymous *union* is inserted in the same scope where it is defined.

```
F:\CHN-103\Lxx_typedef,enum,struct,union,bitset\unionAnonymous.exe
raju
This union contains X└─efZ-φoXwÇpB and 3.24e-090

Process returned 0 (0x0)   execution time : 8.201 s
Press any key to continue.
```

class datatype

```
1  #include <iostream>
2  using namespace std;
3  class complex {
4      private:
5          double Re, Im;
6      public:
7          complex();                      // Constructor function
8          complex(double, double);        // Overloading constructor
9          ~complex();                     // Destructor function
10         void setRe(double);             // Accessor function
11         void setIm(double);             // Accessor function
12         double getRe();                 // Accessor function
13         double getIm();                 // Accessor function
14 };
```

Unlike *struct* and *union*, the members of *class* are **private** by default.

class datatype

```
16  complex::complex(){
17      Re = 0.0; Im = 0.0;
18      cout << "\nConstructor is called." << endl;
19  }
20  complex::complex(double re, double im){
21      Re = re; Im = im;
22      cout << "\nParameterized Constructor is called." << endl;
23  }
24  complex::~complex(){
25      cout << "\nDestructor is called." << endl;
26  }
27  void complex::setRe(double dd){
28      Re = dd;
29  }
30  void complex::setIm(double dd){
31      Im = dd;
32  }
33  double complex::getRe(){    // This is a member function of class complex
34      return (Re);
35  }
36  double complex::getIm(){    // This is a member function of class complex
37      return (Im);
38  }
```

class datatype

```
40  int main(){
41
42      complex c1;
43      cout << "Size of complex number c1 is " << sizeof(c1) << endl;
44      cout << c1.getRe() << "+i" << c1.getIm() << endl;
45
46      c1.setRe(2.34); c1.setIm(-1.34);
47      cout << c1.getRe() << "+i" << c1.getIm() << endl;
48      {
49          complex c2;
50          c2.setRe(4.34); c2.setIm(0.89);
51          cout << c1.getRe() << "+i" << c1.getIm() << endl;
52          cout << c2.getRe() << "+i" << c2.getIm() << endl;
53      }
54      complex c3(1.24,-9.35);
55      cout << c3.getRe() << "+i" << c3.getIm() << endl;
56
57  }
```

class datatype