第七章 用户自定义数据类型

§7.1.1 结构体

- 结构体是一种构造数据类型
- 用途：把不同类型的数据组合成一个整体------自定义数据类型
- ★结构体类型定义

```
struct    [结构体名]
{
    类型标识符   成员名;
    类型标识符   成员名;
    ..............
};
```

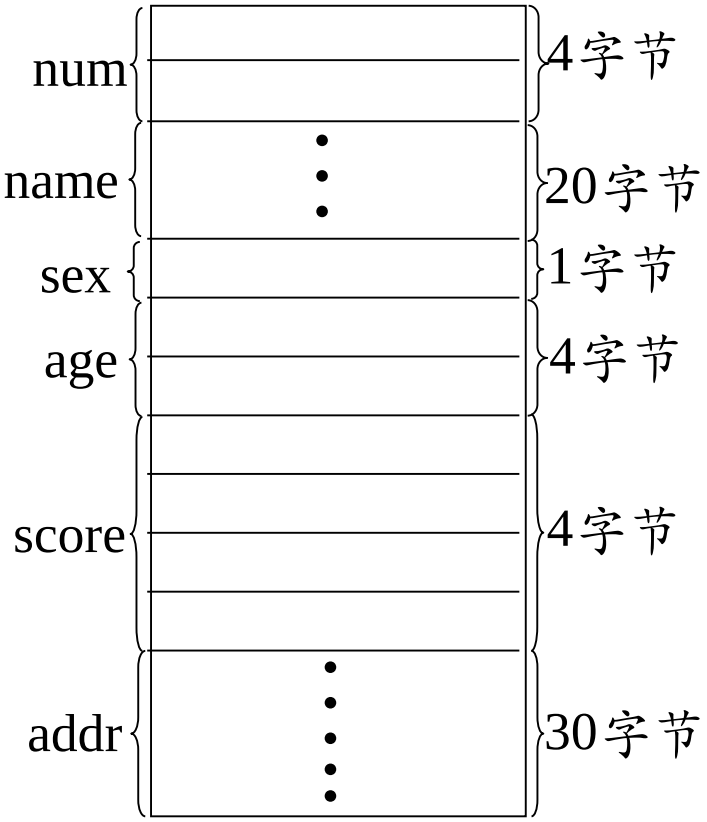
合法标识符
可省:无名结构体

成员类型可以是基本型或构造型

struct是关键字，不能省略

```
例   struct  student
     {     int num;
           char  name[20];
           char sex;
           int age;
           float score;
           char addr[30];
     };
```

结构体类型定义描述结构的组织形式，不分配内存



§7.1.2 结构体变量的定义

★先定义结构体类型，再定义结构体变量

❖一般形式：

```
struct    结构体名
{
    类型标识符   成员名;
    类型标识符   成员名;
      ……
};
struct  结构体名  变量名表列;
```

```
例   struct   student
     {      int num;
            char  name[20];
            char sex;
            int age;
            float score;
            char addr[30];
     };
     struct student   stu1,stu2;
```

struct 在C++中可省略

★定义结构体类型的同时定义结构体变量

一般形式:

```
struct    结构体名
{
    类型标识符   成员名;
    类型标识符   成员名;
      ..............
}变量名表列;
```

例
```
struct   student
{       int num;
        char  name[20];
        char sex;
        int age;
        float score;
        char addr[30];
}stu1,stu2;
```

★说明

❖结构体类型与结构体变量概念不同

- 类型:不分配内存; 变量:分配内存
- 类型:不能赋值、存取、运算; 变量:可以

❖结构体可嵌套

```
例 struct  date
   {    int month;
        int day;
        int year;
   };
  struct  student
  {   int  num;
      char name[20];
      struct  date  birthday;
  }stu;
```

num	name	birthday		
		month	day	year

§7.1.3 结构体变量的引用

★引用规则

❖ 结构体变量不能整体引用,只能引用变量成员

引用方式: 结构体变量名.成员名

成员(分量)运算符
优先级: 2
结合性:从左向右

```
例   struct  student
     {     int num;
           char  name[20];
           char sex;
           int age;
           float score;
           char addr[30];
     }stu1,stu2;
```

stu1.num=10;

stu1.score=85.5;

stu1.score+=stu2.score;
stu1.age++;

cout<<stu1; (×)

§7.1.3 结构体变量的引用

★引用规则

- ❖ 结构体变量不能整体引用,只能引用变量成员

  引用方式: 结构体变量名.成员名

- ❖可以将一个结构体变量赋值给另一个结构体变量

```
例   struct  student
     {      int num;
            char  name[20];
            char sex;
            int age;
            float score;
            char addr[30];
     }stu1,stu2;
```

stu2=stu1; (√)

§7.1.4 结构体变量的初始化

★形式一：

```
struct    结构体名
{
    类型标识符   成员名;
    类型标识符   成员名;
      ……
};
struct 结构体名 结构体变量={初始数据};
```

例
```
struct student
{       int num;
        char name[20];
        char sex;
        int age;
        char addr[30];
};
struct student stu1={112,“Wang Lin”,‘M’,19, “200 Beijing Road”};
```

★形式二：

```
struct    结构体名
{
    类型标识符   成员名;
    类型标识符   成员名;
      ……
}结构体变量={初始数据};
```

例
```
struct  student
{       int num;
        char  name[20];
        char sex;
        int age;
        char addr[30];
}stu1={112,“Wang Lin”,‘M’,19, “200 Beijing Road”};
```

§7.1.5 结构体数组

★结构体数组的定义

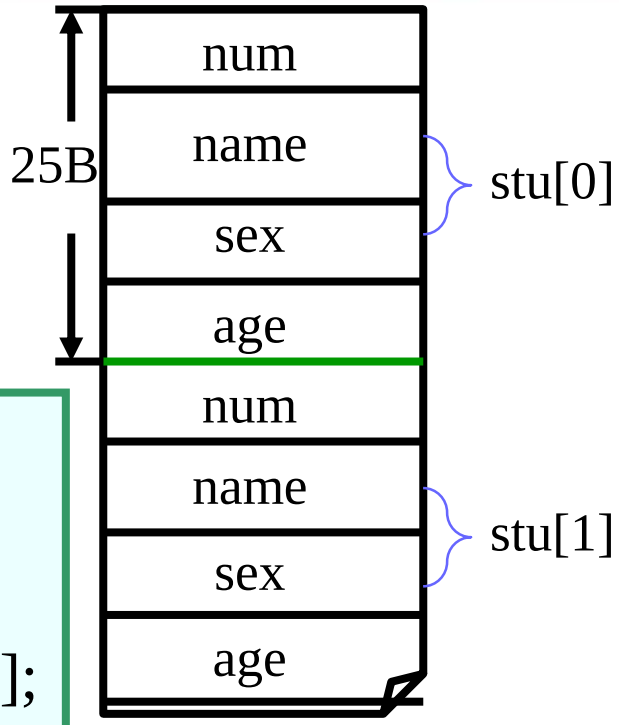
二种形式:

形式一:

```
struct  student
    {    int  num;
         char name[20];
         char sex;
         int age;
    };
struct  student   stu[2];
```

形式二:

```
 struct  student
    {    int  num;
         char name[20];
         char sex;
         int age;
    }stu[2];
```



例 统计候选人选票

```
struct person
{   char name[20];
    int count;
}leader[3]={"Li",0,"Zhang",0,"Wang",0};
void main()
{   int i,j;  char   leader_name[20];
   for(i=1;i<=10;i++)
   {   cin>>leader_name;
       for(j=0;j<3;j++)
          if(strcmp(leader_name,leader[j].name)==0)
               leader[j].count++;
   }
   for(i=0;i<3;i++)
      cout<<leader[i].name<<':'<<leader[i].count<<endl;
}
```

name	count
Li	0
Zhang	0
Wang	0

§7.1.6 结构体和指针

存放结构体变量在内存的起始地址

★指向结构体变量的指针

❖定义形式：struct 结构体名 *结构体指针名;

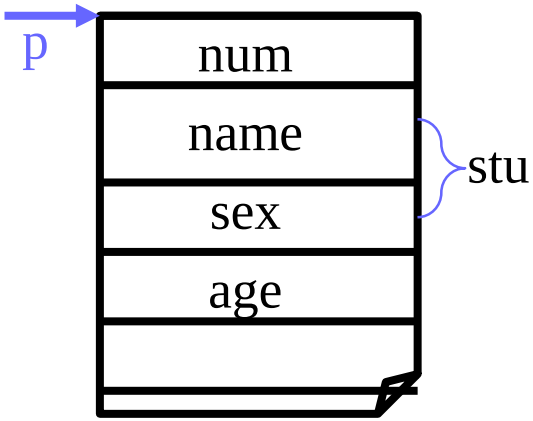
例 struct student *p;

❖使用结构体指针变量引用成员形式

```
例   int   n;
     int  *p=&n;
    *p=10;      ⇔ n=10
```

```
struct   student    stu;
struct   student    *p=&stu;
stu.num=101;  ⇔  (*p).num=101
```

```
struct student
{    int num;
     char name[20];
     int age;
}stu;
struct student  *p=&stu;
```



❖使用结构体指针变量引用成员形式

```
struct   student    stu;
struct   student    *p=&stu;
stu.num=101;  ⇔  (*p).num=101
```

结构体变量名.成员名

⇔ (*结构体指针名).成员名

⇔ 结构体指针名—>成员名

指向运算符
优先级: 2
结合方向：从左向右

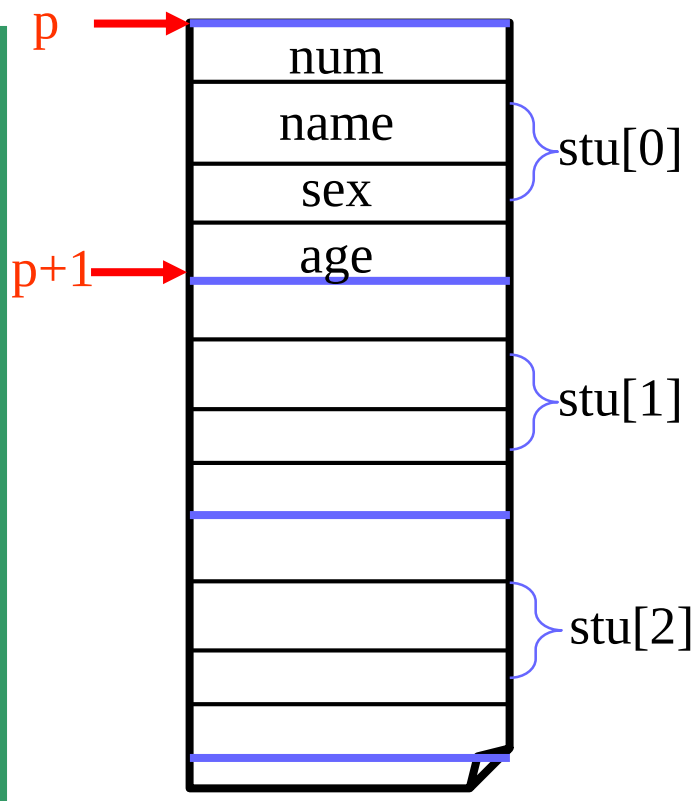
stu1.num=101; ⇔ (*p).num=101 ⇔ p->num

```
void main()
{   struct student
    {       long int num;
            char name[20];
            char sex;
            float score;
    }stu_1,*p;
    p=&stu_1;
    stu_1.num=89101;
    strcpy(stu_1.name,"Li Lin");
    p->sex='M';
    p->score=89.5;
    cout<<(*p).num<<p->name<<stu_1.sex<<p->score;
}
```

★指向结构体数组元素的指针

例 指向结构体数组元素的指针

```
struct student
{    int num;
     char name[20];
     char sex;
     int age;
}stu[3]={{10101,"Li Lin",'M',18},
              {10102,"Zhang Fun",'M',19},
              {10104,"Wang Min",'F',20}};
void main()
{  struct student *p;
   for(p=stu;p<stu+3;p++)
       cout<<p->num<<p->sex<<p->age;
}
```



7.1.7 动态分配和撤销内存的运算符new和delete

➢ 动态分配和撤销内存空间

- 在C语言中是利用库函数malloc和free来分配和撤销内存空间的。
- C++提供了较简便而功能较强的**运算符new和delete**来取代malloc和free函数。
- 注意： new和delete是运算符，不是函数。

```
例如： new int;
//开辟一个存放整数的存储空间，
//返回一个指向该存储空间的地址(即指针)
```

- new运算符使用的一般格式为：

✓ new 类型； 或 new 类型[]； 或 new 类型(初始化)；

- 用new分配数组空间时不能指定初值。
- 如果由于内存不足等原因而无法正常分配空间，则new会返回一个空指针NULL，用户可以根据该指针的值判断分配空间是否成功。

- delete运算符使用的一般格式为：

✓ delete 指针变量； 或 delete [] 指针变量；

例如：

```
new int(100);
```

//开辟一个存放整数的空间，并指定该整数的初值为100，返回一个指向该存储空间的地址

```
new char[10];     //开辟一个存放字符数组(包括10个元素)的空间，返回首元素的地址
new int[5][4];  //开辟一个存放二维整型数组(大小为5*4)的空间，返回首元素的地址
float *p=new float(3.14159);
```

//开辟一个存放单精度数的空间，并指定该实数的初值为3.14159，将返回的该空间的地址赋给指针变量p

例如：

撤销用new开辟的存放单精度数的空间

```
delete  p;
```

撤销用 “new char[10];”开辟的字符数组空间

```
delete [ ] pt;
```

例 作业P188第一题:

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

void swap(int  *p1, int  *p2)
{ int p; p=*p1; *p1=*p2; *p2=p; }

void main()
{  int *p1=new int,*p2=new int,*p3=new int;
   cin>>*p1>>*p2>>*p3;
   if(*p1>*p2)   swap(p1,p2);
   if(*p1>*p3)   swap(p1,p3);
   if(*p2>*p3)   swap(p2,p3);
   cout<<*p1<<*p2<<*p3;
   delete p1,p2,p3;
}
```

例 作业P188第一题:

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

void swap(int  *p1, int  *p2)
{ int p; p=*p1; *p1=*p2; *p2=p; }

void main()
{  int *p=new int[3];
   cin>>*p>>*(p+1)>>*(p+2);
   if(*p>*(p+1))   swap(p,p+1);
   if(*p>*(p+2))   swap(p,p+2);
   if(*(p+1)>*(p+2))   swap(p+1,p+2);
   cout<<*p<<*(p+1)<<*(p+2);
   delete [] p;
}
```

例 作业P188第一题:

```
#include <iostream>
using namespace std;
void swap(int &a, int &b)
{ int t; t=a;a=b;b=t; }
void main()
{  int k[3];
   cin>>k[0]>>k[1]>>k[2];
   if(k[0]>k[1])  swap(k[0],k[1]);
   if(k[0]>k[2])  swap(k[0],k[2]);
   if(k[1]>k[2])  swap(k[1],k[2]);
   cout<<k[0]<<k[1]<<k[2];
}
```

例 作业P188第三题:

```
#include <iostream>
using namespace std;
void Input(int b[],int n)
{  for(int i=0;i<n;i++) cin>>b[i];}
void Output(int b[],int n)
{  for(int i=0;i<n;i++) cout<<b[i]<<' ';}
void Process(int b[],int n)
{  int i,Max=b[0],MaxPosition=0,Min=b[0],MinPosition=0;
   for(i=1;i<n;i++)
       if(Min>b[i]) {Min=b[i];MinPosition=i;}
   b[MinPosition]=b[0]; b[0]=Min;
   for(i=1;i<n;i++)
       if(Max<b[i]) {Max=b[i];MaxPosition=i;}
   b[MaxPosition]=b[n-1];b[n-1]=Max;
}
void main()
{  int a[10];
   Input(a,10); Process(a,10); Output(a,10);
}
```

例
```
#include <iostream>
using namespace std;
void Input(int *p,int n)
{  for(int i=0;i<n;i++) cin>>*(p+i);}
void Output(int *p,int n)
{  for(int i=0;i<n;i++) cout<<*(p+i)<<' ';}
void Process(int *p,int n)
{  int i,Max=*p,MaxPosition=0,Min=*p,MinPosition=0;
   for(i=1;i<n;i++)
       if(Min>*(p+i)) {Min=*(p+i);MinPosition=i;}
   *(p+MinPosition)=*p; *p=Min;
   for(i=1;i<n;i++)
       if(Max<*(p+i)) {Max=*(p+i);MaxPosition=i;}
   *(p+MaxPosition)=*(p+n-1);*(p+n-1)=Max;
}
void main()
{  int *p=new int[10];
   Input(p,10); Process(p,10); Output(p,10);
   delete [] p;
}
```

例 作业P188第八题：

```
#include <iostream>
using namespace std;
void main()
{  char c;int i,Upper=0,Lower=0,Number=0,Space=0,Others=0;
   cout<<"Input a string:";
   while((c=getchar())!='\n')
   {  if(c>='a'&&c<='z') Lower++;
      else if(c>='A'&&c<='Z') Upper++;
      else if(c>='0' && c<='9') Number++;
      else if(c==' ') Space++;
      else Others++;
   }
    cout << "Uppers Num:" << Upper << endl;
    cout << "Lowers Num:" << Lower << endl;
    cout << "Digits Num:" << Number << endl;
    cout << "Spaces Num:" << Space << endl;
    cout << "Others Num:" << Others << endl;
}
```

例 作业P188第八题:

```
#include <iostream>
using namespace std;
void main()
{  char str[80];
   int i,Upper=0,Lower=0,Number=0,Space=0,Others=0;
   cout<<"Input a string:";  gets(str);
   for(i=0;i<strlen(str);i++)
   {  if(str[i]>='a'&&str[i]<='z') Lower++;
      else if(str[i]>='A'&&str[i]<='Z') Upper++;
      else if(str[i]>='0' && str[i]<='9') Number++;
      else if(str[i]==' ') Space++;
      else Others++;
   }
    cout << "Uppers Num:" << Upper << endl;
    cout << "Lowers Num:" << Lower << endl;
    cout << "Digits Num:" << Number << endl;
    cout << "Spaces Num:" << Space << endl;
    cout << "Others Num:" << Others << endl;
}
```

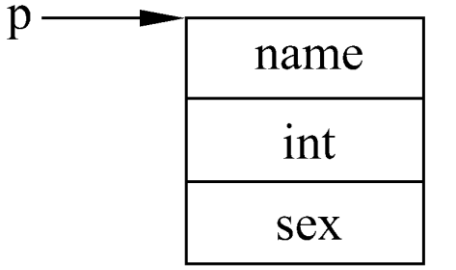
例
```
#include <iostream>
using namespace std;
void main()
{  char *str=new char[80];
   int i,Upper=0,Lower=0,Number=0,Space=0,Others=0;
   cout<<"Input a string:";  gets(str);
   for(i=0;i<strlen(str);i++)
   {  if(*(str+i)>='a'&&*(str+i)<='z') Lower++;
      else if(*(str+i)>='A' && *(str+i)<='Z') Upper++;
      else if(*(str+i)>='0' && *(str+i)<='9') Number++;
      else if(*(str+i)==' ') Space++;
      else Others++;
   }
    cout << "Uppers Num:" << Upper << endl;
    cout << "Lowers Num:" << Lower << endl;
    cout << "Digits Num:" << Number << endl;
    cout << "Spaces Num:" << Space << endl;
    cout << "Others Num:" << Others << endl;
    delete [] str;
}
```

例 开辟空间以存放一个结构体变量。

```
#include <iostream>
#include <cstring>
using namespace std;
struct Student
{ string name;
  int num;
  char sex;
};
```

```
int main( )
{ Student *p;
  p=new Student;
  p->name="Wang Fun";
  p->num=10123;
  p->sex='m';
  cout<<p->name<<endl;
  cout<<p->num<<endl;
  cout<<p->sex<<endl;
  delete p;
  return 0;
}
```

运行结果为
Wang Fun
10123
m



用 new student
开辟的空间

7.2 枚举类型 enum

- 枚举:是指将变量的值一一列举出来，变量的值只能在列举出来的值的范围内。
- 声明枚举类型的一般形式为:

  enum 枚举类型名 {枚举常量表列};

  - 例如:

```
enum weekday{sun,mon,tue,wen,thu,fir,sat};
```

枚举类型类型名

枚举元素

说明:

(1) 对枚举元素按常量处理，故称枚举常量。

(2) 枚举元素作为常量，它们是有值的，C++编译按定义时的顺序对它们赋值为0,1,2,3,…。

(3) 枚举值可以进行关系运算。

【例】判断用户输入的是星期几。

```
int main( ){
  enum week{ Mon = 1, Tues, Wed, Thurs, Fri, Sat, Sun } day;
  cin>>day;
  switch(day){
    case Mon: puts("Monday"); break;
    case Tues: puts("Tuesday"); break;
    case Wed: puts("Wednesday"); break;
    case Thurs: puts("Thursday"); break;
    case Fri: puts("Friday"); break;
    case Sat: puts("Saturday"); break;
    case Sun: puts("Sunday"); break;
    default: puts("Error!");
   }
   return 0;
}
```

§7.3 用typedef定义类型

★功能：用自定义名字为已有数据类型命名

★类型定义简单形式：　typedef　type　name;

类型定义语句关键字　已有数据类型名　用户定义的类型名

例　typedef　int　INTEGER;

例　typedef　float　REAL;

类型定义后,与已有类型一样使用

```
例　INTEGER　a,b,c;
    REAL　 f1,f2;
```

⇔

```
int　 a,b,c;
float　 f1,f2;
```

说明:

1. typedef 没有创造新数据类型

2. typedef 是定义类型,不能定义变量

3. typedef 与 define 不同

define	typedef
预编译时处理	编译时处理
简单字符置换	为已有类型命名

★typedef定义类型步骤

①按定义变量方法先写出定义体　如 int i;

②将变量名换成新类型名　如 int INTEGER;

③最前面加typedef　如 typedef int INTEGER;

④用新类型名定义变量　如 INTEGER i,j;

例 定义数组类型

① int a[100];

② int ARRAY[100];

③ typedef int ARRAY[100];

④ ARRAY a,b,c;

⇔ int a[100],b[100],c[100];

例 定义指针类型

① char *str;

② char *STRING;

③ typedef char *STRING;

④ STRING p,s[10];

⇔ char *p, *s[10];

例 定义函数指针类型

① int (*p)();

② int (*POWER)();

③ typedef int (*POWER)();

④ POWER p1,p2;

⇔ int (*p1)(), (*p2)();

例 定义结构体类型

① struct date
{ int month;
int day;
int year;
}d;

② struct date
{ int month;
int day;
int year;
}DATE;

③typedef struct date
{ int month;
int day;
int year;
}DATE;

④ DATE birthday, *p;

⇔ struct date
{ int month;
int day;
int year;
}birthday, *p;

```
例   typedef  struct club
     {      char  name[20];
            int  size;
            int  year;
     }GROUP;
     typedef   GROUP    *PG;
    PG    pclub;
```

GROUP为结构体类型
PG为指向GROUP的指针类型

⇔ GROUP *pclub;
⇔ struct club *pclub;

作业：

P_{211} 1

提示：要求使用结构体

例 作业P211第一题:

```
#include <iostream>
using namespace std;
struct Date
{  int day;  int month;  int year;};
void main(){
  Date d;  int i,Sum=0;
  int M[12] = {31,28,31,30,31,30,31,31,30,31,30,31};
  cout<<"Input a date(dd mm yy):";
  cin>>d.day>>d.month>>d.year;
  if((d.year%4==0 && d.year%100!=0) || d.year%400==0)
      M[1]=29;
  for(i=0;i<d.month-1;i++)
      Sum += M[i];
  Sum += d.day;
  cout<<"The date is "<<Sum<<"th.\n";
}
```