结构体

引言：用来描述对象特征的各类信息， 例如公司或用户信息，通常会被整合成记录。记录使相似对象的信息组织，表示与存储变得轻松了。

以日期为例：定义一天的日期要定义3个变量，如果要使定义第二个日期而不改变第一个日期的记录，要再次进行定义

```
int year1, int month1, int day1;
int year2, int month2, int day2;
```

```
struct today {
    int year;
    int month;
    int day;
}
```

```
#include <stdio.h>

int main() {
    int x = 1, y = 2, c;
    c = x + y;
    printf("%d + %d = %d\n", x, y, c);
    int x2 = 3, y2 = 4, c2;
    c2 = x2 + y2;
    printf("%d + %d = %d\n", x2, y2, c2);
    printf("%d\n", c > c2 ? c : c2);
    return 0;
}
```

```c
#include <stdio.h>

struct {
    int a;
    int b;
    int c;
} add[2];

int main() {
    for(int i = 0; i < 2; i++) {
        scanf("%d %d", &add[i].a, &add[i].b);
    }
    int max = 0;
    for(int i = 0; i < 2; i++) {
        add[i].c = add[i].b + add[i].a;
        if(add[i].c > max) {
            max = add[i].c;
        }
        printf("c%d = %d\n", i + 1, add[i].c);
    }
    printf("max = %d", max);
    return 0;
}
```

如果定义100个日期的变量，实现会变得十分苦难，为了减少定义时要进行的操作，我们引用结构体来进行定义。

结构体可以对不同类型的同组数据进行整合成一个整体。

结构体对成员的类型不加以控制可以使整型，浮点型，字符型...

结构体定义：

```c
struct 结构体名 {
    类型1 成员名1 :
    类型2 成员名2 :
    类型3 成员名3 :
    ...
    类型n 成员名n :
}
```

例：

```c
struct person {
    char name[100];
    int age;
    float height;
}
```

结构体的成员可以定义为任何所需的完整类型，包括之前已定义的结构类型。但是不能时长度可变的数组或者指向长度可变数组的指针。

例：

```
struct boby {
    float height;
    float weight;
}

struct Person {
    char name[100];
    int age;
    struct boby yourself;
}
```

等效于：

```
struct Person {
    char name[100];
    int age;
    float height;
    float weight;
}
```

结构体类型无法将自己的类型作为其成员的类型，因为自己的类型定义尚不完整， 要在定义结束后才算定义完整。但是结构体类型可以包含指向自己类型的指针。

```
struct cell {
    struct Person person;
     stuct cell *pnext;
}
```

如果在多个源代码的文件中使用同一个结构类型，应该将它的定义放在头文件中，再在各个源代码文件中包含该头文件。通常，同一头文件中也会定义操作该结构类型的函数原型。在所有包含给定头文件的源代码文件中均可以使用该结构类型及其对应的操作函数。

在定义结构体的过程中可以使用typedef对结构体起别名

```
struct person person1, person2;

typedef struct person per;
per person1, person2
```

```
typedef struct {
    char name[100];
    int age;
    float height;
    float weight;
} Person;
```

结构体的需要的空间

```
struct data_test {
    char a;
    short b;
    char c[2];
    double d;
    char e;
    int f;
    char g
}data;
```

存放成员	成员长度	内存对齐要求（字节数）	实际存放位置（相对于起始地址的偏移）
a	1	1	[0]
b	2	2	[2, 3]
c[0]	1	1	[4]
c[1]	1	1	[5]
d	8	4	[8, 15]
e	1	1	[16]
f	4	4	[20, 23]
g	1	1	[24]


```
struct person {
    char name[20]; //姓名  4个单元
    int age;       //年龄  1个单元 占用4个字节
    char gender;   //性别  1个单元 占用1个字节 浪费3个字节（如果够下一个存储则不再申请）
    float height;  //身高  1个单元 占用4个字节  不够存储再次申请1个单元
};
```

```
struct node1 {
    char a;   //1个单元
    char b;   //足够存储不再次申请
    int c;    //1个单元
}
8个字节

struct node2 {
    char a;    //1个单元
    int c;     //1个单元  不够存储再次申请
    char b;    //1个单元
}
12个字节
```

typedef声明

处理复杂名称类型时，一个便捷的方式是，为他们定义简单的同义词。使用typedef声明可定义同义词。

typedef声明以关键字typedef作为开头，后面接着普通对象或者函数的声明语法，但不能有存储类型或_Alignas限定符，以及不能有初始化器。

在typedef声明中的每个声明符，为给定的类型定义了一个标识符，这个标识符称为该类型typedef名称。除去关键字typedef，剩下的语法与声明一个对象或函数的给定类型是相同的，例：

```
typedef unsigned int UINT, UINT_FUNC();
typedef struct Point {
    doule x, y;
} point_t;
typedef float Matrix_t[3][10];
```

在上述声明作用域中，UINT是unsigned int的同义词，Point_t是结构类型struct Point的同义词。可以在声明中使用这些typedef名称

```
UINT ui = 10,  uiPtr = &ui;
```

变量ui 属于unsigned int类型,而uiPtr是指向unsigned int的指针

```
UINT_FUNC *funcPtr
```

指针fucnPtr指向一个函数该函数的返回值是unsigned int类型。该函数没有制定参数

```
Matrix_t *func(float *);
```

函数func()有一个参数，其类型是执行float的指针，返回值是指向Matri.x_t的指针

结构体的使用

通过定义结构体我们可以将一系列类型相同或不同的元素放在一起。例如如果我们希望定义一种存放个人基本信-息的结构，我们就可以通过定义结构体的关键字struct完成这一过程:

```
struct person {
    char name[20]; //姓名
    int age;       //年龄
    char gender;   //性别
    float height;  //身高
};
```

如果我们需要保存tom的信息，我们就不再需要分别声明四个不同类型变量，而只需要使用结构体类型直接声明结构体变量:

```
struct person tom;
```

```
struct person tom = {
    "Tom Cruise",
    54,
    'm',
    170.18f
};
```

我们也可以直接声明变量tom

```
struct {
    char name[20];  // 姓名
    int age;        // 年龄
    char gender;    // 性别
    float height;   // 身高
} tom;
```

结构体成员的访问

两种方式： 直接访问。如：stu1. age

用指针访问。先定义指向结构体的指针：struct student *p;

然后可以通过：(*p) . 成员变量 或 p->成员变量 来访问。//(结构体指针名).结构体成员元素名 都可以写成结构体指针名->结构体成员元素名。

计算总分

Description利用结构数组处理多个学生信息。给定若干个学生的信息，假设学生信息包括学号、姓名、3门课的成绩，计算每个学生的总分，并按要求进行输出。

Input先输入一个整数n，表示有n个学生的信息。接着输入每个学生的学号、姓名以及3门课程的成绩。

Output输出每个学生的学号、姓名以及总分。每个学生的信息占据一行。

HINT假设学生人数不超过100人。学生姓名为长度不超过20的字符。

Sample Input

3

1101

peter chen

90 91 92

1102

susan wang

87 88 89

1103

anney li

86 85 84

Sample Output

1101 peter chen 273

1102 susan wang 264

1103 anney li 255

代码;

```
#include <stdio.h>
#include <string.h>

typedef struct student {
    int num;
    char name[20];
    int su[3];
    int sum;
} stu;

int main() {
    int n;
    stu s[1000];
    scanf("%d", &n);
    for(int i = 0; i < n; i++) {
        scanf("%d", &s[i].num);
        getchar();
        gets(s[i].name);
        s[i].sum = 0;
        for(int j = 0; j < 3; j++) {
            scanf("%d", &s[i].su[j]);
            s[i].sum += s[i].su[j];
        }
    }
    for(int i = 0 ; i < 3; i++) {
        printf("%d %s %d\n", s[i].num, s[i].name, s[i].sum);
    }
    return 0;
}
```