结构体的定义与使用

2023年9月15日 16:38

8 结构体

8.1 结构体基本概念

结构体属于用户自定义的数据类型，允许用户存储不同的数据类型

8.2 结构体定义和使用

语法： `struct 结构体名 { 结构体成员列表 };`

通过结构体创建变量的方式有三种：

- struct 结构体名 变量名
- struct 结构体名 变量名 = { 成员1值， 成员2值...}
- 定义结构体时顺便创建变量

示例：

```
//结构体定义
struct student
{
    //成员列表
    string name;  //姓名
    int age;      //年龄
    int score;    //分数
}stu3; //结构体变量创建方式3


int main() {

    //结构体变量创建方式1
    struct student stu1; //struct 关键字可以省略

    stu1.name = "张三";
    stu1.age = 18;
    stu1.score = 100;

    cout << "姓名: " << stu1.name << " 年龄: " << stu1.age  << " 分数: " << stu1.score <<
```

```
    //结构体变量创建方式2
    struct student stu2 = { "李四",19,60 };

    cout << "姓名: " << stu2.name << " 年龄: " << stu2.age  << " 分数: " << stu2.score <<
endl;
```

```
    //结构体变量创建方式2
    struct student stu2 = { "李四",19,60 };

    cout << "姓名: " << stu2.name << " 年龄: " << stu2.age  << " 分数: " << stu2.score <<
endl;


    stu3.name = "王五";
    stu3.age = 18;
    stu3.score = 80;


    cout << "姓名: " << stu3.name << " 年龄: " << stu3.age  << " 分数: " << stu3.score <<
endl;

    system("pause");

    return 0;
}
```

总结1：定义结构体时的关键字是struct，不可省略

总结2：创建结构体变量时，关键字struct可以省略

总结3：结构体变量利用操作符 "." 访问成员

总体来说输出时符合一一对应的原则

结构体数组

2023年9月15日 16:38

8.3 结构体数组

作用： 将自定义的结构体放入到数组中方便维护

语法： `struct 结构体名 数组名[元素个数] = { {} , {} , ... {} }`

```
#include<iostream>
using namespace std;
struct student {
    string name;
    int age;
    int grade;
};
int main() {
    struct student array[3] = {
        {"张三",18,89 },
        {"李四",17,80},
        {"王五",15,87}};
    for (int i = 0; i < 3; i++) {
        cout << "  姓名为: " << array[i].name
            << "  年龄为:" << array[i].age
            << "  成绩为: " << array[i].grade << endl; }
```

结构体指针

2023年9月15日 18:09

8.4 结构体指针

作用： 通过指针访问结构体中的成员

- 利用操作符 `->` 可以通过结构体指针访问结构体属性

```
struct student
{
    //成员列表
    string name;  //姓名
    int age;      //年龄
    int score;    //分数
};


int main() {

    struct student stu = { "张三",18,100, };

    struct student * p = &stu;

    p->score = 80; //指针通过 -> 操作符可以访问成员

    cout << "姓名: " << p->name << " 年龄: " << p->age << " 分数: " << p->score << endl;

    system("pause");

    return 0;
}
```

> 总结：结构体指针可以通过 -> 操作符 来访问结构体中的成员

结构体嵌套结构体

2023年9月15日 20:52

8.5 结构体嵌套结构体

作用： 结构体中的成员可以是另一个结构体

例如： 每个老师辅导一个学员，一个老师的结构体中，记录一个学生的结构体

总结： 在结构体中可以定义另一个结构体作为成员，用来解决实际问题

```
//学生结构体定义
struct student
{
    //成员列表
    string name;  //姓名
    int age;      //年龄
    int score;    //分数
};

//教师结构体定义
struct teacher
{
    //成员列表
    int id; //职工编号
    string name;  //教师姓名
    int age;   //教师年龄
```

```
    struct student stu; //子结构体 学生
};


int main() {

    struct teacher t1;
    t1.id = 10000;
    t1.name = "老王";
    t1.age = 40;

    t1.stu.name = "张三";
    t1.stu.age = 18;
    t1.stu.score = 100;

    cout << "教师 职工编号:  " << t1.id << " 姓名:  " << t1.name << " 年龄:  " << t1.age << endl;

    cout << "辅导学员 姓名:  " << t1.stu.name << " 年龄: " << t1.stu.age << " 考试分数:  " << t1.stu.score << endl;

    system("pause");
```

结构体做函数参数

2023年9月15日 18:09

8.6 结构体做函数参数

作用：将结构体作为参数向函数中传递

传递方式有两种：

- 值传递
- 地址传递

```
int main() {

    student stu = { "张三",18,100};
    //值传递
    printStudent(stu);
    cout << "主函数中 姓名: " << stu.name << " 年龄:  " << stu.age << " 分数: " << stu.score << endl;

    cout << endl;

    //地址传递
    printStudent2(&stu);
    cout << "主函数中 姓名: " << stu.name << " 年龄:  " << stu.age  << " 分数: " << stu.score << endl;

    system("pause");

    return 0;
}
```

> 总结：如果不想修改主函数中的数据，用值传递，反之用地址传递

结构体中const的使用场景

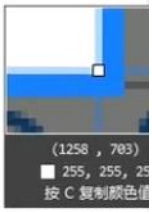
2023年9月15日 20:52

- 用指针（地址传递）的好处：减少内存

8.7 结构体中 const使用场景

作用： 用const来防止误操作

```
//学生结构体定义
struct student
{
    //成员列表
    string name;  //姓名
    int age;      //年龄
    int score;    //分数
};

//const使用场景
void printStudent(const student *stu) //加const防止函数体中的误操作
{
    //stu->age = 100; //操作失败，因为加了const修饰
    cout << "姓名: " << stu->name << " 年龄: " << stu->age << " 分数: " << stu->score << endl;

}

int main() {

    student stu = { "张三",18,100 };

    printStudent(&stu);

    system("pause");

```



加const使得不能修改实参的值

结构体案例1

2023年9月15日 21:20

8.8 结构体案例

8.8.1 案例1

案例描述：

学校正在做毕设项目，每名老师带领5个学生，总共有3名老师，需求如下

设计学生和老师的结构体，其中在老师的结构体中，有老师姓名和一个存放5名学生的数组作为成员

学生的成员有姓名、考试分数，创建数组存放3名老师，通过函数给每个老师及所带的学生赋值

最终打印出老师数据以及老师所带的学生数据。

```
#pragma once
#include<iostream>       //头文件
using namespace std;
#include<string>
#include<ctime>
struct student { string name; int age; int grade; };      //自定义的结构体
struct teacher { string tname; struct student array[5]; };
void Student1(struct teacher tarray[], int len);          //函数的声明
void allocatespace(struct teacher tarray[], int len);
```

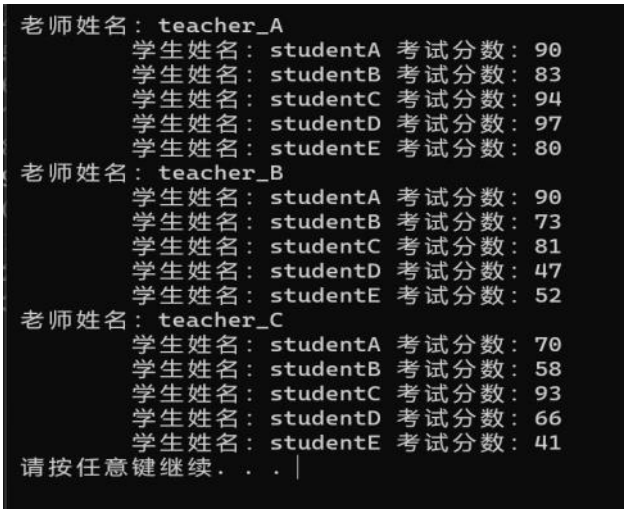
分文件编写

- 头文件中要写定义好的函数的声明。
- 头文件中要包括本地头文件，自定义的结构体，函数的声明。
- 源文件中要引用头文件。

```
#include"swap.h"  //头文件的引用
void Student1(struct teacher tarray[], int len) {
    for (int i = 0; i < len; i++) {
        cout << "老师姓名：" << tarray[i].tname << endl;
        for (int j = 0; j < 5; j++) {
            cout << "\t学生姓名：" << tarray[i].array[j].name << " 考试分数：" << tarray[i].array[j].grade << endl;
        }//   \t为转移字符，为了对齐数据
    }
}
```

```
#include"swap.h"
void allocatespace(struct teacher tarray[], int len) {
    string nameseed = "ABCDE";   //字符串的语法定义
    for (int i = 0; i < len; i++) {
        tarray[i].tname = "teacher_";
        tarray[i].tname += nameseed[i];  //字符串可以理解为本身就为以一个数组
        for (int j = 0; j < 5; j++) {
            tarray[i].array[j].name = "student";   //理解：tarray[i]为上面定义好的变量（已经赋好值）.
            tarray[i].array[j].name += nameseed[j];//   .array[j].name 为定义老师结构体中的学生结构体数组的变量，  .name为赋值类型为学生结构体中的name数据类型
            int random = rand() % 61 + 40;//范围为：40~100；去掉+40为：0~60；随机数生成定义
            tarray[i].array[j].grade = random;
        }
    }
}
```

```
int main() {
    srand((unsigned int)time(NULL));//随机数种子

    struct teacher tarray1[3];
    int len = sizeof(tarray1) / sizeof(tarray1[0]);
    allocatespace(tarray1, len); //函数的调用
    Student1(tarray1, len);
    system("pause");
    return 0;
}
```


老师姓名：teacher_A
学生姓名：studentA 考试分数：90
学生姓名：studentB 考试分数：83
学生姓名：studentC 考试分数：94
学生姓名：studentD 考试分数：97
学生姓名：studentE 考试分数：80
老师姓名：teacher_B
学生姓名：studentA 考试分数：90
学生姓名：studentB 考试分数：73
学生姓名：studentC 考试分数：81
学生姓名：studentD 考试分数：47
学生姓名：studentE 考试分数：52
老师姓名：teacher_C
学生姓名：studentA 考试分数：70
学生姓名：studentB 考试分数：58
学生姓名：studentC 考试分数：93
学生姓名：studentD 考试分数：66
学生姓名：studentE 考试分数：41
请按任意键继续. . .

结构体案例2

2023年9月15日 21:20

8.8.2 案例2

案例描述：

设计一个英雄的结构体，包括成员姓名，年龄，性别;创建结构体数组，数组中存放5名英雄。

通过冒泡排序的算法，将数组中的英雄按照年龄进行升序排序，最终打印排序后的结果。

五名英雄信息如下：

```
{"刘备",23,"男"},
{"关羽",22,"男"},
{"张飞",20,"男"},
{"赵云",21,"男"},
{"貂蝉",19,"女"},
```

```
#pragma once
#include<iostream>
using namespace std;
struct hero{ string name; int age; string sex; };
void bubblesort(struct hero harray[], int len);
void printhero(struct hero array[], int len);
```

```
#include"swap.h"
void bubblesort(struct hero harray[], int len) {
    for (int i = 0; i < len - 1; i++) {
        for (int j = 0; j < len - 1 - i; j++) {
            if (harray[j].age > harray[j + 1].age) {
                struct hero temp = harray[j];
                harray[j] = harray[j + 1];
                harray[j + 1] = temp;
            }
        }
    }
}
```

```cpp
#include"swap.h"
int main() {
    struct hero harray[5]{
        {"刘备",23,"男"},
        {"关羽",22,"男"},
        {"张飞",20,"男"},
        {"赵云",21,"男"},
        {"貂蝉",19,"女"},};
    int len = sizeof(harray) / sizeof(harray[0]);
    cout << "排序前：" << endl;
    printhero(harray, len);
    bubblesort(harray, len);
    cout << "排序后：" << endl;
    printhero(harray, len);
    system("pause");
    return 0;
}
```

```cpp
#include"swap.h"
void printhero(struct hero array[], int len) {
    for (int i = 0; i < len; i++) {
        cout << "英雄姓名为：" << array[i].name << "  年龄为：  "
            << array[i].age << "  性别为：  " << array[i].sex <<
            endl;
    }
}
```