# 目录

[目录 1](#_Toc520909991)

[第8章 C语言-结构体 2](#_Toc520909992)

[8.1 C语言结构体概述 2](#_Toc520909993)

[8.2 结构体变量定义与初始化 2](#_Toc520909994)

[8.2.1 构造结构体数据类型 2](#_Toc520909995)

[8.2.2 结构体变量定义 4](#_Toc520909996)

[8.2.3 结构体变量的初始化 4](#_Toc520909997)

[8.3 结构体变量使用 5](#_Toc520909998)

[8.3.1 结构体变量读写操作 5](#_Toc520909999)

[8.3.2 环形缓冲区 5](#_Toc520910000)

[8.4 结构体数组定义与初始化 10](#_Toc520910001)

[8.4.1 结构体数组定义 10](#_Toc520910002)

[8.4.2 结构体数组的初始化 10](#_Toc520910003)

[8.5 结构体数组使用 11](#_Toc520910004)

[8.5.1 结构体数组的读写操作 11](#_Toc520910005)

[8.6 结构体指针定义与初始化 12](#_Toc520910006)

[8.6.1 结构体指针定义 12](#_Toc520910007)

[8.6.2 结构体指针的初始化 13](#_Toc520910008)

[8.7 结构体指针使用 16](#_Toc520910009)

[8.7.1 结构体指针的使用 16](#_Toc520910010)

# C语言-结构体

## C语言结构体概述

数组：数据具有相同数据类型的集合 🡪不能存储不同数据类型的数据

存50个学生的数学成绩(double)：double score[50];

存50本书的价格double：double price[50];

学生信息：姓名、性别、学号、成绩、年龄…可以用数组来存？

不可以，数据类型不同

姓名：char 姓别：char 学号：int 成绩：double 年龄：int

为了决解这个问题 🡪 结构体

结构体：数据具有相同数据类型或者不同数据类型的集合

可以定义一个结构体来存储上面的学生信息🡪 定义了一个学生类型的结构体

## 结构体变量定义与初始化

### 构造结构体数据类型

在使用结构体前，要先构造好结构体类型

|  |
| --- |
| struct 结构体名  {  成员列表；  数据类型 成员名1；  数据类型 成员名2；  数据类型 成员名3；  数据类型 成员名4；  ...  };分号不能省略 |

struct：结构体类型关键字

结构体名：满足标识符的命名规则，具有含义

struct 结构体名：表示结构体类型 定义结构体变量：struct 结构体名 变量名

数据类型 [长度]：数组类型

指针类型：数据类型 \*

成员列表：数据类型 成员名；//定义普通变量一模一样

不能对成员进行初始化

成员与成员之间用分号隔开

一个成员占一行

注意：

1. 结构体类型大括号后面的分号不能省略；
2. 结构类型定义时系统不会为其分配空间，只有结构定义变量时系统才会为其分配空间
3. 结构体类型的成员可以是结构体，前提是先定义
4. 结构类型的定义单文件：放在头文件包含和主函数之间，在多文件中，放在.h文件中

|  |
| --- |
| #include <stdio.h>  struct Birthday  {  int year;  int month;  int day;  };  //构造了结构体类型  struct Student  {  char name[10];//存姓名  char sex[5];//存性别  int number;//存学号  double score;//存成绩  int age;//存年龄  struct Birthday bir;  };  //分号不能省略  int main(void)  {  return 0;  } |

### 结构体变量定义

上面已经将结构体类型构造好了，接下来就可以使用结构体类型来定义变量了

结构体类型定义变量有三种形式

1. 第一种

|  |
| --- |
| 构造结构体类型的同时定义结构体变量  struct 结构体名  {  成员列表；  数据类型 成员名1；  数据类型 成员名2；  数据类型 成员名3；  数据类型 成员名4；  ...  }变量名1，变量名2;分号不能省略  struct Birthday  {  int year;  int mon;  int day;  };  struct Student  {  char name[10];  char sex[5];  int number;  int score;  int age;  struct Birthday bri;  }stu1, stu2;//定义结构体类型的同时定义结构体变量  int main(void)  {  return 0;  } |

1. 第二种

|  |
| --- |
| 先构造结构体类型，再定义结构体变量  struct 结构体名  {  成员列表；  数据类型 成员名1；  数据类型 成员名2；  数据类型 成员名3；  数据类型 成员名4；  ...  }；  struct 结构体名 变量名1，变量名2；  struct Birthday  {  int year;  int mon;  int day;  };  struct Student  {  char name[10];  char sex[5];  int number;  int score;  int age;  struct Birthday bri;  };  int main(void)  {  struct Student stu1, stu2;//定义结构体变量  return 0;  } |

1. 第三种

|  |
| --- |
| 构造结构体类型的同时定义结构体变量，省略结构体名  struct  {  成员列表；  数据类型 成员名1；  数据类型 成员名2；  数据类型 成员名3；  数据类型 成员名4；  ...  }变量名1，变量名2;分号不能省略  //只是一次性，后面不能再定义结构体变量  //struct 结构体名：结构体类型  struct Birthday  {  int year;  int mon;  int day;  };  struct  {  char name[10];  char sex[5];  int number;  int score;  int age;  struct Birthday bri;  }stu1, stu2;//只是一次性，后面不能再定义结构体变量  //struct 结构体名：结构体类型  int main(void)  {  return 0;  } |

1. 升级改造

|  |
| --- |
| typedef：给数据类型取别名    先构造结构体类型，再定义结构体变量  typedef struct 结构体名  {  成员列表；  数据类型 成员名1；  数据类型 成员名2；  数据类型 成员名3；  数据类型 成员名4；  ...  }别名；//注意：这里此时是别名，不是结构体变量    别名 变量名;  struct 结构体名 变量名；  struct Birthday  {  int year;  int mon;  int day;  };  typedef struct Student  {  char name[10];  char sex[5];  int number;  int score;  int age;  struct Birthday bri;  }STU;//别名  int main(void)  {  struct Student stu1, stu2;  STU stu3;  return 0;  } |

### 结构体变量的初始化

1. 对结构体变量初始化时，要用大括号进行整体赋值，也只有初始化时才能用大括号进行整体赋值
2. 对结构体变量初始化，其实就是对结构体成员进行初始化
3. 对结构体变量初始化，数据间用逗号隔开
4. 对结构体变量初始化，系统按照顺序从第一个成员开始赋值
5. 对结构体变量初始化，也支持未完全初始化，未赋值的成员默认赋值0
6. 对结构体变量初始化，如果成员是结构体类型，可以按照顺序依次赋值，也可以用大括号括起来赋值

|  |
| --- |
| struct Birthday  {  int year;  int mon;  int day;  };  typedef struct Student  {  char name[10];  char sex[5];  int number;  int score;  int age;  struct Birthday bri;  }STU;//别名  int main(void)  {  struct Student stu1 = {"张三", "女", 124563, 100, 20, {1999, 10, 19}};  return 0;  } |

### 结构体变量的赋值

1. 当结构体变量定义好后，不能在使用大括号进行整体赋值
2. 如果成员是字符数组
3. strcpy函数
4. gets函数
5. 结构体变量可以进行直接赋值

|  |
| --- |
| struct Birthday  {  int year;  int mon;  int day;  };  typedef struct Student  {  char name[10];  char sex[5];  int number;  int score;  int age;  struct Birthday bri;  }STU;//别名  int main(void)  {  struct Student stu1 = {"张三", "女", 124563, 100, 20, {1999, 10, 19}};  struct Student stu2, stu3;    // stu2.name = "李四"; //这种写法是错误  strcpy(stu2.name, "李四");//第一种写法  // gets(stu2.name);//第二种写法  strcpy(stu2.sex, "男");//第一种写法  stu2.number = 125643;  stu2.score = 99;  stu2.age = 19;  stu2.bri.year = 2000;  stu2.bri.mon = 11;  stu2.bri.day = 2;  stu3 = stu2;//结构体变量整体赋值  return 0;  } |
|  |

## 结构体变量使用

格式：结构体变量名.成员名； .：结构体成员运算符

|  |
| --- |
| struct Birthday  {  int year;  int mon;  int day;  };  typedef struct Student  {  char name[10];  char sex[5];  int number;  int score;  int age;  struct Birthday bri;  }STU;//别名  int main(void)  {  struct Student stu1 = {"张三", "女", 124563, 100, 20, {1999, 10, 19}};  struct Student stu2, stu3;  // stu2.name = "李四"; //这种写法是错误  strcpy(stu2.name, "李四");//第一种写法  // gets(stu2.name);//第二种写法  strcpy(stu2.sex, "男");//第一种写法  stu2.number = 125643;  stu2.score = 99;  stu2.age = 19;  stu2.bri.year = 2000;  stu2.bri.mon = 11;  stu2.bri.day = 2;  stu3 = stu2;//结构体变量整体赋值  printf("%s %s %d %d %d %d %d %d\r\n", stu1.name, stu1.sex, stu1.number, stu1.score, stu1.age,  stu1.bri.year, stu1.bri.mon, stu1.bri.day);  printf("%s %s %d %d %d %d %d %d\r\n", stu2.name, stu2.sex, stu2.number, stu2.score, stu2.age,  stu2.bri.year, stu2.bri.mon, stu2.bri.day);  printf("%s %s %d %d %d %d %d %d\r\n", stu3.name, stu3.sex, stu3.number, stu3.score, stu3.age,  stu3.bri.year, stu3.bri.mon, stu3.bri.day);  return 0; |

课堂练习：比较三十个学生的成绩，输出成绩最高学生的信息

1先构造出来结构体类型

2定义结构体变量(三个)

3键盘输入三个学生信息

4.比较出分数最高信息的学生

5.输出分数最高信息的学生

## 结构体数组定义与初始化

### 结构体数组定义

一个结构体变量只能存一个学生的信息

50个学生，50个结构体变量来存。代码量大，麻烦

结构体数组：相当于多个结构体变量

整型数组：整型 [长度] 变量名 🡪 整型 数组名[长度]；

结构体数组：结构体类型 [长度] 变量名 🡪 结构体类型 数组名[长度]；

struct 结构体名 数组名[长度]；

定义结构体数组也有三种形式

1. 第一种

|  |
| --- |
| 构造结构体类型的同时定义结构体数组  struct 结构体名  {  成员列表；  数据类型 成员名1；  数据类型 成员名1；  数据类型 成员名1；  数据类型 成员名1；  ...  }数组名1[长度];分号不能省略  struct Student  {  char name[10];  char sex[5];  int number;  int score;  int age;  struct Birthday bri;  }stu[3];//此时就定义了一个结构体数组 |

1. 第二种

|  |
| --- |
| 先构造结构体类型，再定义结构体数组  struct 结构体名  {  成员列表；  数据类型 成员名1；  数据类型 成员名2；  数据类型 成员名3；  数据类型 成员名4；  ...  }；    struct 结构体名 数组名[长度]；//定义好了一个结构体数组  struct Student  {  char name[10];  char sex[5];  int number;  int score;  int age;  struct Birthday bri;  };  int main(void)  {  struct Student stu[3];//此时就定义了一个结构体数组  return 0;  } |

1. 第三种

|  |
| --- |
| 构造结构体类型的同时定义结构体数组，省略结构体名  struct  {  成员列表；  数据类型 成员名1；  数据类型 成员名2；  数据类型 成员名3；  数据类型 成员名4；  ...  }数组名[长度];分号不能省略  //只是一次性，后面不能再定义结构体数组  //struct 结构体名：结构体类型  struct  {  char name[10];  char sex[5];  int number;  int score;  int age;  struct Birthday bri;  }stu[3];//此时就定义了一个结构体数组  typedef struct Student  {  char name[10];  char sex[5];  int number;  int score;  int age;  struct Birthday bri;  }STU;  int main(void)  {  STU stu[3];//此时就定义了一个结构体数组  return 0;  } |

### 结构体数组的初始化

|  |
| --- |
| struct Birthday  {  int year;  int mon;  int day;  };  struct Student  {  char name[10];  char sex[5];  int number;  int score;  int age;  struct Birthday bri;  };  int main(void)  {  int i;  struct Student stu[3] = {  {"张三", "女", 123465, 88, 15, 2004, 10, 19}, //stu[0]  {"李四", "男", 123765, 78, 16, 2003, 11, 07}, //stu[1]  {"王五", "女", 145465, 89, 18, 2001, 11, 02} //stu[2]  };//此时就定义了一个结构体数组  //stu[1].name = "小明";  // strcpy(stu[1].name, "小茗");  // gets(stu[1].name);    for(i = 0; i < 3; i++)  {  printf("%s %s %d %d %d %d %d %d\r\n", stu[i].name, stu[i].sex, stu[i].number, stu[i].score,  stu[i].age, stu[i].bri.year, stu[i].bri.mon, stu[i].bri.day);  }  return 0;  } |

## 结构体数组使用

格式：数组名[下标].成员名； 下标的范围：0 – 长度-1

struct Stu stu[3];

stu[0].成员名 stu[1] stu[2]

|  |
| --- |
| #include <stdio.h>  //构造结构体类型  typedef struct Student  {  char name[10];  char sex[5];  int number;  int score;  int age;  };//构造结构体类型的同时定义结构体数组  int main(void)  {  int i;  struct Student stu[3] = {//stu[0] stu[1] stu[2]  {"张三", "女", 123456, 88, 17},  {"李四", "男", 123654, 89, 18},  {"王五", "男", 142536, 90, 20}  };//定义结构体数组 初始化  //数组名[下标].成员名；  for(i = 0; i < 3; i++)  {  printf("%s %s %d %d %d\r\n", stu[i].name, stu[i].sex, stu[i].number, stu[i].score, stu[i].age);  }  return 0;  } |

## 结构体指针定义与初始化

### 结构体指针定义

指针类型：数据类型 \*

定义一个指向整型的指针变量：整型 \*指针名；int \*p;

定义一个指向结构体类型的指针变量：结构体类型 \*指针名； struct 结构体名 \*指针名；

定义结构体指针也有三种；

1. 构造结构体类型的同时定义结构体指针
2. 先构造结构体类型，再定义结构体指针
3. 构造结构体类型的同时定义结构体指针，结构体名省略

### 结构体指针的初始化

struct 结构体名 \*指针名 = NULL;

## 结构体指针使用

格式：(\*指针名).成员名

指针名->成员名

\*指针名.成员名 //这种写法是错误的

|  |
| --- |
| #include <stdio.h>  #include <string.h>  struct Birthday  {  int year;  int mon;  int day;  };  struct Student  {  char name[10];  char sex[5];  int number;  int score;  int age;  struct Birthday bri;  };  int main(void)  {  int i;    struct Student stu[3] = {  {"张三", "女", 123465, 88, 15, 2004, 10, 19}, //stu[0]  {"李四", "男", 123765, 78, 16, 2003, 11, 07}, //stu[1]  {"王五", "女", 145465, 89, 18, 2001, 11, 02} //stu[2]  };//此时就定义了一个结构体数组  struct Student \*pstu = stu;//指向了stu数组的首元素地址  //stu[1].name = "小明";  // strcpy(stu[1].name, "小茗");  // gets(stu[1].name);    for(i = 0; i < 3; i++)  {  printf("%s %s %d %d %d %d %d %d\r\n", stu[i].name, stu[i].sex, stu[i].number, stu[i].score,  stu[i].age, stu[i].bri.year, stu[i].bri.mon, stu[i].bri.day);  }  printf("\r\n");  /\* for(i = 0; i < 3; i++)  {  printf("%s %s %d %d %d %d %d %d\r\n", (\*(pstu+i)).name, (\*(pstu+i)).sex, (\*(pstu+i)).number, (\*(pstu+i)).score,  (\*(pstu+i)).age, (\*(pstu+i)).bri.year, (\*(pstu+i)).bri.mon, (\*(pstu+i)).bri.day);  }\*/  for(i = 0; i < 3; i++)  {  printf("%s %s %d %d %d %d %d %d\r\n", (pstu+i)->name, (pstu+i)->sex, (pstu+i)->number, (pstu+i)->score,  (pstu+i)->age, (pstu+i)->bri.year, (pstu+i)->bri.mon, (pstu+i)->bri.day);  }  return 0;  } |

## 结构体类型空间分布

|  |
| --- |
| #include <stdio.h>    struct Student  {  char a;  short b;  int d;  double e;  float f; //8  };//别名 按照成员里面数据类型最大来对齐  int main(void)  {  struct Student stu;  //stu空间有多大  char \*p = &stu;  short \*f;  int \*g;  \*p = 'a';  f = p;  \*(f+1) = 10;//+1偏移两个字节  g = p;  \*(g+1) = 100;//+1偏移四个字节  printf("%c\r\n", stu.a);  printf("%d\r\n", stu.b);  printf("%d\r\n", stu.d);  return 0;  } |