

数据结构实验

实验二

实验二：顺序存储

一、实验目的

- 1、复习结构体、数组、指针；
- 2、掌握数组的静态创建与动态创建；
- 3、了解顺序存储的基本访问方法。

引例1

```
typedef struct score_grade
{ int score;
  char grade; } student;

void Input_Score(int n, student stu[])
//数组stu存储n个学生的分数与等级，通过键盘输入分数
{ int i;
  for(i=0;i<n;i++)
  { printf("请输入第%d个学生的分数(0—100): \n",i+1);
    scanf("%d",&stu[i].score);
    while(stu[i].score<0||stu[i].score>100)
    { printf("请重新输入第%d个学生的分数(0—100): \n",i+1);
      scanf("%d",&stu[i].score); } }
}
```

引例1

```
void Score_Grade(int n,student stu[])
//数组stu存储n个学生的分数与等级，转换分数为等级
{ int i;
  for(i=0;i<n;i++)
  { switch(stu[i].score/10)
    { case 10:
      case 9: stu[i].grade='A'; break;
      case 8:
      case 7: stu[i].grade='B'; break;
      case 6: stu[i].grade='C'; break;
      default: stu[i].grade='D'; } }
}
```

引例1

```
void Output_Score_Grade(int n, student stu[])
//数组stu存储n个学生的分数与等级，输出分数与等级
{ int i;
  student *p;
  printf("各个学生的分数与等级如下: \n");
  for(i=0,p=stu;i<n;i++,p++)
    printf("第%d个学生: \t分数为%d\t等级为%c\n", p->score,
    p->grade);
}
int main()
{ student stu[5];
  Input_Score(5,stu);
  Score_Grade(5,stu);
  Output_Score_Grade(5,stu);
}
```

引例2

```
typedef struct score_grade
{ int score;
  char grade; } student;

void Input_Score(int n, student *stu)
//指针stu所指数组存储n个学生的分数与等级，通过键盘输入分数
{ int i;
  for(i=0;i<n;i++)
  { printf("请输入第%d个学生的分数(0—100): \n",i+1);
    scanf("%d",&stu[i].score);
    while(stu[i].score<0||stu[i].score>100)
    { printf("请重新输入第%d个学生的分数(0—100): \n",i+1);
      scanf("%d",&stu[i].score); } }
}
```

引例2

```
void Score_Grade(int n, student *stu)
//指针stu所指数组存储n个学生的分数与等级，转换分数为等级
{ int i;
  for(i=0;i<n;i++)
  { switch(stu[i].score/10)
    { case 10:
      case 9: stu[i].grade='A'; break;
      case 8:
      case 7: stu[i].grade='B'; break;
      case 6: stu[i].grade='C'; break;
      default: stu[i].grade='D'; } }
}
```

引例2

```
void Output_Score_Grade(int n, student *stu)
//指针stu所指数组存储n个学生的分数与等级，输出分数与等级
{ int i;
  printf("各个学生的分数与等级如下: \n");
  for(i=0;i<n;i++)
    printf("第%d个学生: \t分数为%d\t等级为%c\n",i+1 (stu+i)->score,
    (stu+i)->grade);
}
```

引例2

```
int main()
{ int n;
  student *stu;
  printf("请输入数组大小 (1-100) : \n");
  scanf("%d",&n);
  while(n<1||n>100)
  { printf("请重新输入数组大小 (1-100) : \n");
    scanf("%d",&n); }
  stu=(student*)malloc(n*sizeof(student));
  if (!stu) exit(-1);
  Input_Score(n,stu);
  Score_Grade(n,stu);
  Output_Score_Grade(n,stu);
  free(stu);
}
```

引例2

思考1：动态创建数组的好处是什么？

思考2：数组元素的访问方法有些什么？

实验二：顺序存储

二、实验内容

- 1、（必做题）每个学生的成绩信息包括：学号、语文、数学、英语、总分、加权平均分；**采用动态方法创建数组**用于存储若干学生的成绩信息；输入学生的学号、语文、数学、英语成绩；计算学生的总分和加权平均分（语文占30%，数学占50%，英语占20%）；输出学生的成绩信息。

实验二：顺序存储

二、实验内容

- 2、（必做题）可以在数组末尾追加新学生的成绩信息；可以根据学号，删除该学生的成绩信息。
- 3、（选做题）可以根据学号或总分，升序排序学生的成绩信息。