09 - 一维数组

C++ 程序设计基础

SOJ 信息学竞赛教练组

2024 年 5 月 17 日

目录

多重循环

- 一重循环

多重循环

- 一重循环
- 二重循环

多重循环

- 一重循环
- 二重循环
  - 在一重循环的循环体中再写一个循环

多重循环

- 一重循环
- 二重循环
  - 在一重循环的循环体中再写一个循环
- 三重循环

多重循环

- 一重循环
- 二重循环
  - 在一重循环的循环体中再写一个循环
- 三重循环
- ...

目录

引入

- 使用循环求和的时候，都是一边输入数据一边维护当前的和，并没有将所有的数据都存储起来

引入

- 使用循环求和的时候，都是一边输入数据一边维护当前的和，并没有将所有的数据都存储起来
- 如果需要将求和的数据重新输出一遍，则没有办法实现

思考

当变量很多且需要记录他们的值时，该怎么办?

数组

- 数组的概念
  - 数组是用于存储多个同类型数据的结构
  - 多个数据存放在一片连续的内存空间中

```
int A[8];
```

A[0]	A[1]	A[2]	A[3]	A[4]	A[5]	A[6]	A[7]
71	80	62	91	99	82	44	35

数组

- 数组的概念
  - 数组是用于存储多个同类型数据的结构
  - 多个数据存放在一片连续的内存空间中
- 数组的优势
  - 代码简洁
  - 通用、易维护

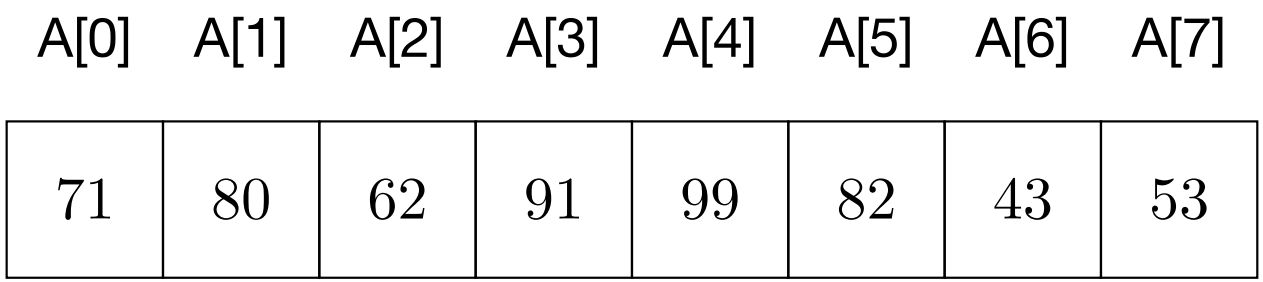
```
int A[8];
```

A[0]	A[1]	A[2]	A[3]	A[4]	A[5]	A[6]	A[7]
71	80	62	91	99	82	44	35

数组

- 数组的声明
  - **元素类型　数组名 [数组大小];**
  - `int A[8]; // 定义了可以存放 8 个整数的数组 A`

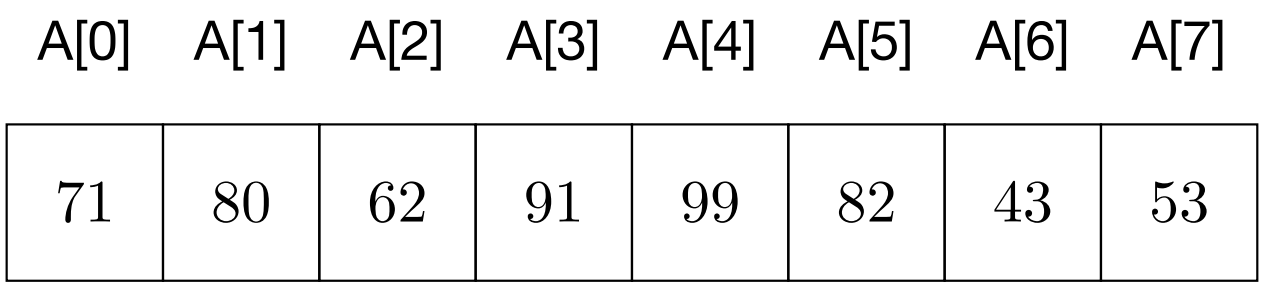
`int A[8];`



数组

- 数组的声明
  - **元素类型　数组名 [数组大小];**
  - `int A[8]; // 定义了可以存放 8 个整数的数组 A`
  - 数组中的基本单元叫作“元素”
  - 数组大小定义为可能存放元素数量的最大值（可适当再大一些）

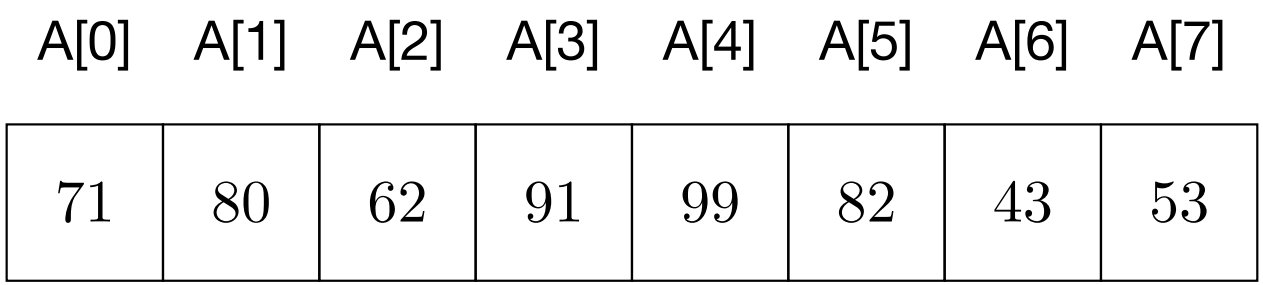
`int A[8];`



数组

- 数组的声明
  - **元素类型　数组名 [数组大小];**
  - `int A[8]; // 定义了可以存放 8 个整数的数组 A`
  - 数组中的基本单元叫作“元素”
  - 数组大小定义为可能存放元素数量的最大值（可适当再大一些）
  - 建议声明为全局变量

`int A[8];`



数组

- 数组的初始化
  - 可以通过一个赋值符号和花括号，将花括号中给定的数值序列，从第 0 位开始，依次赋值给数组中的每一个元素
    - `int A[8] = {71, 80, 62, 91, 99, 82, 43, 53};`

`int A[8];`

A[0]	A[1]	A[2]	A[3]	A[4]	A[5]	A[6]	A[7]
71	80	62	91	99	82	43	53

数组

- 数组的初始化
  - 可以通过一个赋值符号和花括号，将花括号中给定的数值序列，从第 0 位开始，依次赋值给数组中的每一个元素
    - `int A[8] = {71, 80, 62, 91, 99, 82, 43, 53};`
  - 数组大小可以指定或留空，会按照需要的最小空间申请内存
    - `double B[] = {1.0, 2.7, 3.14, 9.8};`

`int A[8];`

A[0]	A[1]	A[2]	A[3]	A[4]	A[5]	A[6]	A[7]
71	80	62	91	99	82	43	53

数组

- 数组元素的访问
  - 通过 **数组名 [下标]** 访问存储在数组中某一位置的值
    - 下标是指在元素在数组中的位置，从 0 开始
    - 下标的范围：$0 \sim size - 1$ ($size$ 为数组大小)

```
int A[8];
```

A[0]	A[1]	A[2]	A[3]	A[4]	A[5]	A[6]	A[7]
71	80	62	91	99	82	43	53

数组

- 数组元素的访问
  - 通过 **数组名 [下标]** 访问存储在数组中某一位置的值
    - 下标是指在元素在数组中的位置，从 0 开始
    - 下标的范围：$0 \sim size - 1$ ($size$ 为数组大小)
  - 例如：声明数组 `int A[5];`
    - 可以使用的数组下标：0，1，2，3，4
    - 对应的可以使用的数组元素：`A[0]`，`A[1]`，`A[2]`，`A[3]`，`A[4]`
    - 可以使用变量作为数组下标：`A[k]` ($0 \leq k \leq 4$)

`int A[8];`

A[0]	A[1]	A[2]	A[3]	A[4]	A[5]	A[6]	A[7]
71	80	62	91	99	82	43	53

例 9.1：成绩录入

编程题

- 6 年 A 班进行了一次小测，现要求编写程序，录入所有学生成绩，并输出所有学生的成绩，学生成绩如图所示。

71	80	62	91	99	82	43	53

- 样例输入
  无
- 样例输出

```
71 80 62 91 99 82 42 53
```

例 9.1：成绩录入

```cpp
#include <iostream>

using namespace std;

// 录入成绩
int A[8] = {71, 80, 62, 91, 99, 82, 43, 53};

int main() {
  // 输出所有成绩
  cout <<
    A[0] << " " << A[1] << " " << A[2] << " " <<
    A[3] << " " << A[4] << " " << A[5] << " " <<
    A[6] << " " << A[7] << " " << endl;

  return 0;
}
```

例 9.2：成绩查询

编程题

- 已知 6 年 A 班的小测成绩如图所示，编写程序，输入一个整数 k $(0 \leq k < 8)$，输出 k 号学生的分数。

71	80	62	91	99	82	43	53

- 样例输入
  5
- 样例输出
  82

例 9.2：成绩查询

```cpp
#include <iostream>

using namespace std;

int A[8] = {71, 80, 62, 91, 99, 82, 43, 53};

int main() {
  int k;
  cin >> k;
  cout << A[k] << endl;

  return 0;
}
```

思考

当 $k = -1$ 时运行会怎样?

数组

- 数组访问越界
  - 一个大小为 $size$ 的数组，那下标在 0 到 $size-1$ 的之间才是有意义的
  - 访问不在数组有意义区间的元素称为 **数组越界**

```
int A[8];
```

A[0]	A[1]	A[2]	A[3]	A[4]	A[5]	A[6]	A[7]
71	80	62	91	99	82	43	53

数组

- 数组访问越界
  - 一个大小为 $size$ 的数组，那下标在 0 到 $size-1$ 的之间才是有意义的
  - 访问不在数组有意义区间的元素称为 **数组越界**
  - 这可能会导致程序出现重大错误，但编译不会报错，需要有意识地避免数组越界
  - 尤其是当数组的下标是以变量的形式出现时，更要加以小心

```
int A[8];
```

A[0]	A[1]	A[2]	A[3]	A[4]	A[5]	A[6]	A[7]
71	80	62	91	99	82	43	53

目录

总结

- 数组的概念
- 数组的声明
- 数组的初始化
- 数组元素的访问
- 数组的应用
  - 数组求和
  - 求最值及其下标

Thank you!