一维数组及常量

常量

- 什么是常量呢?程序中使用的**具体**的数和字符,注意与变量进行区分,变量是运行过程中可能会改变的值
- 在程序的运行过程中,**不能被修改**
- **常量的类型**:

- **整型变量:**程序使用过程中使用的具体整数:十进制(非零数字开头,如 9)八进制(以数字 0 开头,0~7 构成)十六进制(0x 开头,0~9 及 A~F 组成) 如下面的数字分别是什么进制,分别代表什么数字(10, 010, 0x10),
  C++不直接支持表示二进制的整数
- **实型常量:**程序中使用的具体整数,如 3.14,科学记数法 -2.1E+2
- **布尔常量:**程序中使用的具体布尔值 true, false
- **字符常量**:程序中使用的具体单个 ASCII 字符, 用一对单引号将单个字符括起来'A','\'' '\\'(**转义字符**)
- **字符串常量:**程序中使用的具体的字符序列,用双引号括起来的字符序列, "abc",(**注意'a'与"a"的区别),使用 sizeof('a') sizeof("a")**
- **符号常量:**可以给常量取个名字,这个名字就是符号常量, 例如 const double PI = 3.1415;**一般符号常量用大写表示,变量用小写,**常量的**定义放在 main 函数外边**,使用符号常量的好处是 **1. 见名知意 2. 一改全改**

一维数组

- 为什么需要一维数组:思考这样一个问题,我们需要用一个程序求出 50 个学生的平均分,并且输出低于平均分的学生序号和对应的成绩,按照我们之前学过的方法是定义 50 个变量,一个个输入,然后求出平均分, 第二次的循环判断谁的分数低于平均分,然后打印出来该学生的成绩,但是 **1. 太繁琐了(50 个变量程序的意义就没有了)2. 不易扩展(如果需要 200,1000 个学生呢?)**
- 可以用一种新的数据类型来表示,这个类型叫做数组,**数组中存储的是大量相同性质的数据,**比如 100 个学生的成绩, 连续 30 天的气温, 森林里树的高度…

1. 一维数组的声明

- 数据类型 数组名称[元素个数]; int arr[50]; double height[100];
- int arr[50]; 如何使用其中的元素, arr[0]是第一个元素, arr[49]是最后一个元素,他们在内存中的存储是连续的
- 数据类型可以是 int double float bool char…
- 数组名称和变量名称命名规则是一样的
- **元素个数需要是常量 不可以 cin >> n; int arr[n];**

2. 一维数组的引用

- 通过数组名称和元素在数组中的**位置编号(下标**,类似于第几班的第几个座位)来使用如 arr[0];
- **下标/索引必须在数组定义的下标范围之内,0~n-1**

- **下标**可以是值为**整型的常量**,如 a[0] a[99]a['a'],也可以是结果为**整型的变量 a[i+j]**
- **数组元素可以像同类的普通变量来进行使用,如 arr[0] = 3; 把 3 赋值给数组 arr 的第一个元素**

3. 一维数组的输入输出

- 输入的元素个数是固定的:

```
#include<iostream>
using namespace std;
int a[10];
int main()
{
    for (int i = 0; i < 10; i++)
    {
        cin >> a[i];
    }
    for(int i=0; i<10; i++)
    {
        cout << a[i] << " ";
    }
    return 0;
}
```

- 输入元素的个数不固定,但是规定不会超过一个最大值,比如规定不会超过 10;

```
#include<iostream>
using namespace std;
const int N = 10;
int a[N];
int main()
{
    int n;
    cin >> n;
    for (int i = 0; i < n; i++)
    {
        cin >> a[i];
    }
    for (int i = 0; i < n; i++)
    {
        cout << a[i] << " ";
    }
    return 0;
}
```

4. 一维数组的初始值

- 当数组定义在主函数之外，初始值为 0,数组定义在主函数之内,初始值随机,;例

```
#include<iostream>
using namespace std;
int b[5];
int main()
{
    int a[5];
    for (int i = 0; i < 5; i++)
    {
        cout << a[i] << " ";
    }
    cout << endl;
    for (int i = 0; i < 5; i++)
    {
        cout << b[i] << " ";
    }
    return 0;
}
```

- 如何初始化为给定的值?

数组的初始化可以在声明时一并完成：

```
类型标识符 数组名[常量表达式] = {值1, 值2, …};
```

在初值列表中可以写出全部数组元素的值：

```
int a[5] = {1, 2, 3, 4, 5};
```

也可以写出部分。例如，以下方式仅对数组的前2个元素进行初始化，其余值为0。

```
int b[5] = {1, 2};
```

对数组元素全部初始化为0，可以简写为{}。

```
int c[5] = {};
```

5. 一维数组遍历

- 数组的遍历就是从头到尾检查每一个元素,输出或者查看是否满足条件
- **将数组类比几班的哪个座位,座位是不能移动的,但是座位上坐的人可以移动的,也不可以凭空插一个座位进去**

6. 一维数组的存储

- 将数组声明在主函数之外,没有存储空间的限制,而且无需格外初始化

7. 一维数组的拷贝

- 如果声明了两个数组, int a[10]; b[10];
- 数组不可以像变量那样进行复制操作 **b=a;是不对的**
- **方法 1.** 而应该使用 for 循环进行复制

```
#include<iostream>
using namespace std;
const int N = 10;
int a[N] = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9};
int b[N];
int main()
{
    for (int i = 0; i < N; i++)
    {
        b[i] = a[i];
    }
    for (int i = 0; i < N; i++)
    {
        cout << b[i] << " ";
    }
    return 0;
}
```

- 方法 2 使用库 cstring 里的 memcpy 函数

如果需要把数组a全部复制到数组b中：

```
memcpy(b, a, sizeof(a));
```

如果要从数组a复制k个元素到数组b：

```
memcpy(b, a, sizeof(int) * k);
```

如果数组a和b都是浮点型：

```
memcpy(b, a, sizeof(double) * k);
```

使用memcpy函数（memory copy）要：

```
#include<cstring>
```