目录

**[一、 循环](#_Toc32400_WPSOffice_Level1)** **[2](#_Toc32400_WPSOffice_Level1)**

[（一） for循环](#_Toc27623_WPSOffice_Level2) [2](#_Toc27623_WPSOffice_Level2)

[（二） While循环](#_Toc2470_WPSOffice_Level2) [3](#_Toc2470_WPSOffice_Level2)

[（三） do-while](#_Toc6468_WPSOffice_Level2) [4](#_Toc6468_WPSOffice_Level2)

**[二、 函数](#_Toc27623_WPSOffice_Level1)** **[5](#_Toc27623_WPSOffice_Level1)**

[1. 函数的类别（我给函数的叫参数，函数给我的叫返回值）](#_Toc7437_WPSOffice_Level2) [5](#_Toc7437_WPSOffice_Level2)

[2. 函数的几点特点：](#_Toc29240_WPSOffice_Level2) [5](#_Toc29240_WPSOffice_Level2)

[3. 例子](#_Toc10195_WPSOffice_Level2) [5](#_Toc10195_WPSOffice_Level2)

**[三、 数组](#_Toc2470_WPSOffice_Level1)** **[6](#_Toc2470_WPSOffice_Level1)**

[（一） 一维数组](#_Toc9112_WPSOffice_Level2) [6](#_Toc9112_WPSOffice_Level2)

[（二） 多维数组](#_Toc26297_WPSOffice_Level2) [7](#_Toc26297_WPSOffice_Level2)

**[四、 关于字符串的内容](#_Toc6468_WPSOffice_Level1)** **[8](#_Toc6468_WPSOffice_Level1)**

# 循环

## for循环

1. 格式：for(初值 ; 判断条件 ; 值改变){循环体}
2. 例子：求从1到10的和

从1到10 的和就是1+2+3+4+5+6+7+8+9+10

#include<stdio.h>

int main(){

int sum = 0;//定义一个变量，用来保存和

int i;//定义一个循环变量

for(i=1; i<=10; i++){

sum + = i;//i从1到10，都加给sum

}

Printf(“%d”,sum);

}

1. 解释一下循环执行过程

1

2 5

for(i=1 ; i<=10 ; i++){

3 4

sum + = i;//i从1到10，都加给sum

}

你会发现345行成一个圈，这就是循环，直到不满足判断条件，就从这个循环跳出来跳到这个位置，然后接着往下执行。

## While循环

1. 格式：while(判断条件){循环体}
2. 例子：求从1到10的和

#include<stdio.h>

int main(){

int sum = 0;//定义一个变量，用来保存和

int i=1;//定义一个循环变量，给个初始值1

while( i<=10){//第一次进来就会判断是否符合条件

sum + = i;//i从1到10，都加给sum

i++；//i每次增加1

}

Printf(“%d”,sum);

}

1. 解释一下循环执行过程

1

while( i<=10 ){

2

sum + = i;//i从1到10，都加给sum

4

3

i++；//i每次增加1

//到底了，返回去

}

234形成一个圈，即一个循环，直到不满足判断条件时退出循环到这个位置，然后继续向下执行

## do-while

1. 格式：do{循环体}while(判断条件)；别忘了这个分号
2. 例子：求从1到10的和

#include<stdio.h>

int main(){

int sum=0;

int i=1;

do{

sum+=i;

i++;

} while(i<=10);

printf("%d", sum);

}

1. 解释一下执行过程：

1

do{

2

sum+=i;

3 5

i++;

4

} while(i<=10);

2345形成一个循环，在循环最后也就是while处判断是否符合条件，符合的话按照5的路线回到do，不符合退出循环到这个位置，然后继续执行。

# 函数

1. 函数的类别（我给函数的叫参数，函数给我的叫返回值）

* 从返回值分类：有返回值的（int、double）;没有返回值的：void
* 从参数来分：有参数的；无参数的

1. 函数的几点特点：

* 函数遇到return就停止
* 没有返回值的函数可以写成void 函数名（参数列表）
* 每一个参数都要标明类型比如：void sum(int a, int b, double c);
* 函数可以写在主函数前面，也可以写在主函数后边，但是，写在主函数后边必须在前边进行声明。生命方法如

1. 例子

#include<bits/stdc++.h>

using namespace std;

int f(int a){

if(a==1){

return 0;

}

int flag = 1;

for(int i=2; i<a; i++){

if(a%i == 0){

flag = 0;

break;

}

}

return flag;

}

int main(){

for(int i=0; i<5; i++){

int a;

cin>>a;

int isSuShu = f(a);

if(isSuShu == 1){

cout<<a<<endl;

}

}

return 0;

}

# 数组

## 一维数组

1. 引言

回顾一下变量，以字符型变量作为例子，定义一个字符变量ch：char ch; 定义一个很好定义，但是如果我让你定义100个呢？你要写一百个char 么？这指定是不现实的。我们说变量是用来装值的，可以理解成花瓶，仓库等装东西的。那么数组就是为了解决一次定义多个变量的东西。可以理解为一排仓库。画一个示意图

变量可以表示为： ，一个仓库

数组就是多个仓库，一串仓库

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

那怎么区分这些仓库，又怎么用么多个变量呢？

1. 数组定义

|  |  |  |
| --- | --- | --- |
| 类型 | 定义变量 | 定义数组(定义100个) |
| int | int a; | int a[100]; |
| long long | long long a; | long long a[100]; |
| float | float a; | float a[100]; |
| double | double a; | double a[100] |

1. 数组引用

a[数字]，比如我要用第二个，就是a[2]，这就可以和普通变量一样使用了。

这里需要强调的是数字第一个是以0开头，不是以1开头。这些数字我们叫他下标。

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16

## 多维数组

1. 引言

回顾一下变量，以字符型变量作为例子，定义一个字符变量ch：char ch; 定义一个很好定义，但是如果我让你定义100个呢？你要写一百个char 么？这指定是不现实的。我们说变量是用来装值的，可以理解成花瓶，仓库等装东西的。那么数组就是为了解决一次定义多个变量的东西。可以理解为一排仓库。画一个示意图

变量可以表示为： ，一个仓库

数组就是多个仓库，一串仓库

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

那怎么区分这些仓库，又怎么用么多个变量呢？

1. 数组定义

|  |  |  |
| --- | --- | --- |
| 类型 | 定义变量 | 定义二维数组 |
| int | int a; | int a[100][100]; |
| long long | long long a; | long long a[100]; |
| float | float a; | float a[100][100]; |
| double | double a; | double a[100][100] |

1. 数组引用

a[横坐标][纵坐标]，比如我要用第二个，就是a[2][3]，这就可以和普通变量一样使用了。

这里需要强调的是数字第一个是以0开头，不是以1开头。这些数字我们叫他下标。

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 1 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 2 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 3 |  |  | a[2][3] |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 4 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15

# 关于字符串的内容

1. 头文件: #include<string.h>
2. strcpy(字符串1, 字符串2) 将字符串2赋值给字符串1  
   str-string-字符串 cpy-copy-复制 ctrl+c
3. strcmp(字符串1,字符串2) 比较两个字符串的大小  
   str-string-字符串 cmp-compare-比较  
   字符串1 > 字符串2 返回正数> 0  
   字符串1==字符串2 返回0  
   字符串1<字符串2 返回负数 <0

# 指针

（\*p）这是一个指针与a做类比 a[4]可以理解为定义了四个变量，

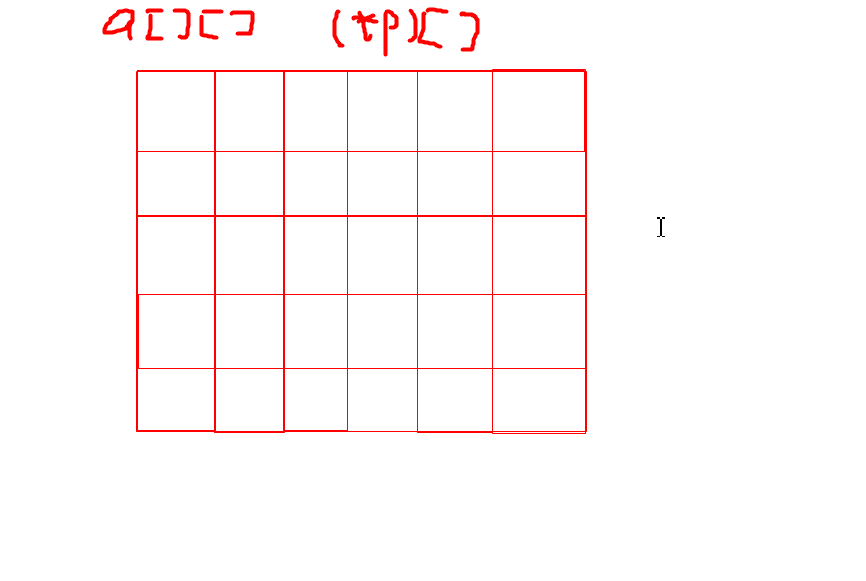
（\*p）[4] 就是四个指针

我们说数组名就是数组的首地址，也就是数组的指针。一个指针指向一个数组。

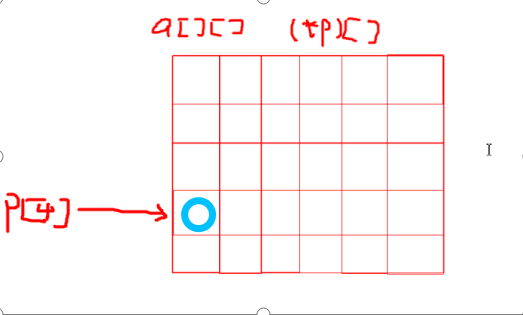
|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |

a

四个指针就指向了四个数组呗，四个数组（我只是说数组，没说数组多长）放一起不就是一个二维数组，有四行么

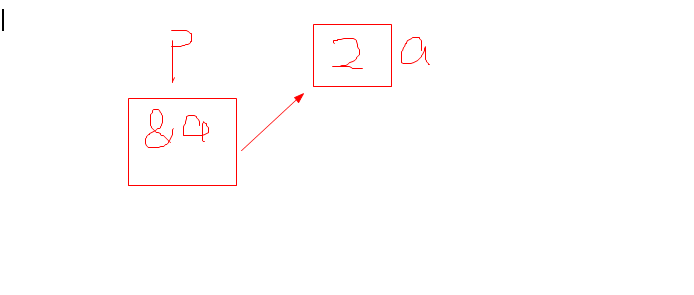
这俩代表的都是二维数组

\*p[4]可以理解\*（p[4]） 只有二维数组才会出现这种表示。如果是一维数组p[4]代表是一个变量，\*一个变量没有意义。所以p[4]是一个指针，指向第四行的指针。



p[4]也就是第四行的首地址，\*是取内容预算符，也就是代表的蓝圈里的值，\*p[4]和p[4][0]是一个意思

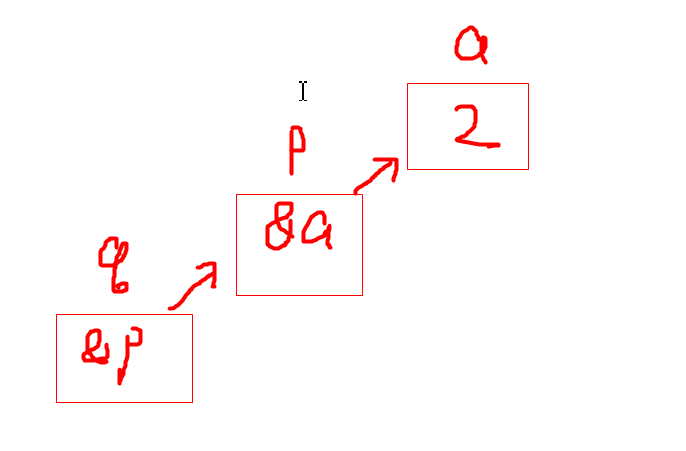
上边说p[4]是第四行的首地址，&p[4][0]也是第四行的首地址。



我们说p是一个指针，指针也是一个变量，只不过这个变量的值是另一变量的地址

a也是一个变量，有指针p指向它，因为p指向了a的地址，所以\*p就是提取a的地址里边的内容，即\*p=2；

那我要是弄一个指针q指向p呢，q就是指向指针的指针



q指向了p，\*q就是提取p里边的值呗即&a。 \*\*p就是连续提取了两次呗。

但是需要注意的是，定义时候的\*是指向的意思，除了定义时候，其他都是提取值得意思。

在形参中的\*\*p就是指向指针p的指针。因为（\*p）是指针啊，形参中是在定义啊，所以是指向，不是提取

总结着说：



int a=2;

int b=3;

int \*p = &a;

int \*\*pp = &p;

\*pp=&b;

怎么理解int \* \* pp；

首先看，前边有int，所以这是一个指针，第一个\*代表着指向一个东西， 再往后看，又看到一个\*pp ，也是一个指针，所以第一指针指向的是一个指针，即指向指针的指针。在给他赋值的时候，制定就得给他一个指针。给他哪里，他就指向了哪里

这个看到int了，知道\*是指向，指向啥样的东西，p前边没有东西，所以说是指向一个普通的变量的

啥类型呢：前边那个类型的

int a[5][5]

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 11 | 12 | 13 | 14 | 15 |
| 21 | 22 | 23 | 24 | 25 |
| 31 |  |  |  |  |
| 41 |  |  |  |  |
| 51 |  |  |  |  |

\*\*a = \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

A 11 B &a[0][0] C a[0] D &(\*a[0])

这个题没有答案，首先看a，我们说这是一个数组名，也就是数组的首地址，指向

\*a是提取地址里边的内容，11 然而\*11没有意义。。。。。

所以没有答案