# C/C++从入门到精通

# 课程介绍

## 讲师介绍

**动脑学院Rock老师：**

十二年Linux平台C++开发经验。

曾就职于京信通信南京研究所。

从事通信行业服务器端开发，无线通信平台的架构。

负责了多个南海军用通信项目。

## 课程目的

1. 大部分初学者，学习C/C++都是从入门到放弃。

1. 大部分初级开发人员只懂得C/C++的皮毛。

函数指针的目的是什么？

C语言的指针陷阱？

模板库的选择？

各个标准模板库的使用陷阱?

怎样避免内存泄露？智能指针的使用陷阱？

怎样使C程序更具有移植性？

怎样使C++程序更加安全、高效？

多线程编程，怎样避免死锁、怎样避免竞态?

原因：

1）学习方法不合适。

2）被误导。

## 适用于

1）零基础。

2）了解C/C++, 但不能熟练掌握。

3）准备跨入IT行业，但不知道学什么的大学生。

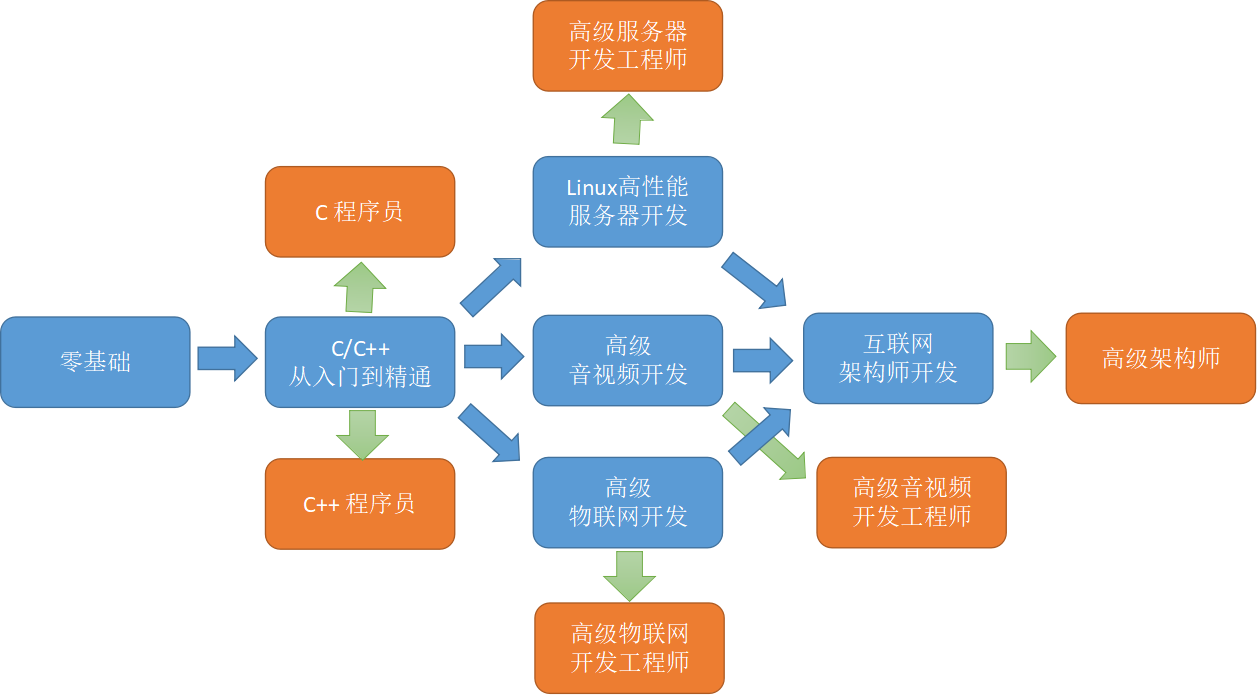
4）准备转入IT行业的其他从业者。

5）准备学习C/C++的其它开发者。

## 不适用于

已具备丰富经验的C++开发人员。

## 学习目标



## 学习内容

《C/C++语言从入门到精通大纲》

## 课程服务

* 一对一讲师指导
* 一对一习题批改
* 企业级项目全程指导
* 简历设计
* 10个以上的面试推荐
* 面试指导
* 笔试指导
* 试用期全程技术指导

## 第一阶段-零基础入门

# C语言的作用、学习方法

## C语言的作用

物联网开发

嵌入式开发

Linux内核开发

Linux系统开发

Linux驱动开发

研究算法、数据结构的必备语言。

学习C++的入门语言。

## C语言的学习方法

### 大道至简

1. 不要刻意记忆语法规则。
2. 以项目为导向，在解决项目问题中学习。
3. 不断试错，在错误中学习。

### 初学者遇到问题的解决办法

1. 自己先思考10分钟。
2. 如果还不能解决，马上问老师。
3. 把问题的解决方案记录下来。(建议用博客）

### 老鸟遇到问题的解决办法

1. 自己研究30分钟以上。
2. 如果还不能解决，百度、谷歌查询类似问题。
3. 重复以上2个步骤。
4. 把问题的解决方案记录下来。(建议用博客）

### 写技术博客的重要性

记录成长轨迹（记录学习上遇到问题，工作上遇到的问题）

扩大个人影响力。（出书、猎头、合作）

每周写一篇，养成习惯。

# 项目1 搭建开发环境

补充：

为什么要搭建开发环境？

编译器的作用？

## Linux平台开发环境的搭建（选修）

### 安装Linux操作系统

建议使用虚拟机vmware方式安装Linux操作系统。

Linux操作系统，可选择：

1. CentOS（建议：Centos 7.0以上）

补充：国内大部分企业的服务器是使用CentOS或（RedHat）

CentOS是Redhat的社区版，用法相同。

1. Ubuntu系统（不建议）

### 确保Linux操作系统能够上网

建议把虚拟机的网卡设置为桥接模式。

检查：

# ping [www.baidu.com](http://www.baidu.com)

或直接在浏览器中打开百度网站(www.baidu.com)

### 在线安装c语言编译器gcc

CentOS系统：

# yum install -y gcc

检验：

# gcc -v

### 在线安装C++语言编译器g++

# yum install -y gcc-c++

检验：

# g++ -v

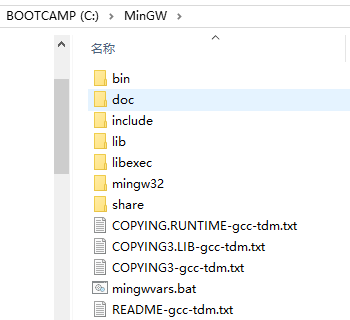
## Windows平台开发环境的搭建

### 方式1：使用MinGW

1. 下载MinGW

百度网盘链接： <https://pan.baidu.com/s/1eTEA2Cq>

1. 把下载的压缩包解压到C:\MinGW目录（其它目录也可）



1. 配置环境变量

把C:\MinGW\bin目录，或其它安装目录下的bin目录，添加到系统的Path环境变量中。

补充：环境变量的作用

4）检验

打开cmd

gcc -v

g++ -v

### 方式2：使用VS

VS （Visual Studio, 最新的是VS2017），是一个大型的集成开发环境（IDE）

初学者，建议使用方式1.

VS太庞大，反而不便于理解程序的本质！

## 开发平台的选择

1. 如果已经了解Linux操作系统的基本使用，建议使用Linux平台
2. 如果不了解Linux操作系统，就直接使用Windows平台，以后再学习Linux操作系统。
3. 零基础的初学者，建议使用Windows平台。

## 编辑器的选择

编辑器的作用：

编写程序（源代码）。

编辑器的选择：

初学者最好使用最简单的文本编辑器，不要使用集成开发环境IDE

Linux平台：vi, vim, 或gedit

Windows平台：记事本，Sublime Text, UltraEdit, notePad, notePad++, source insight

## 开发方式

### 方式1：直接使用IDE

直接使用IDE（例如VS）进行编辑、编译、运行。

### 方式2: 分别使用编辑器和编译器

先用编辑器编写源代码

然后使用编译器，对源代码进行编译，最后再运行。

建议初学者，使用方式2。

# 项目2 交换机后台管理之登录菜单

## 项目需求

用户打开交换机后台管理程序时，需要进行“登录”操作，以确认用户身份的合法性。

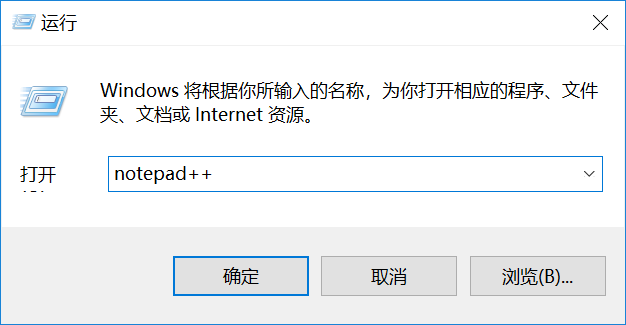
所以，我们需要先实现一个登录菜单，以提示用户执行相关操作。

## 项目实现

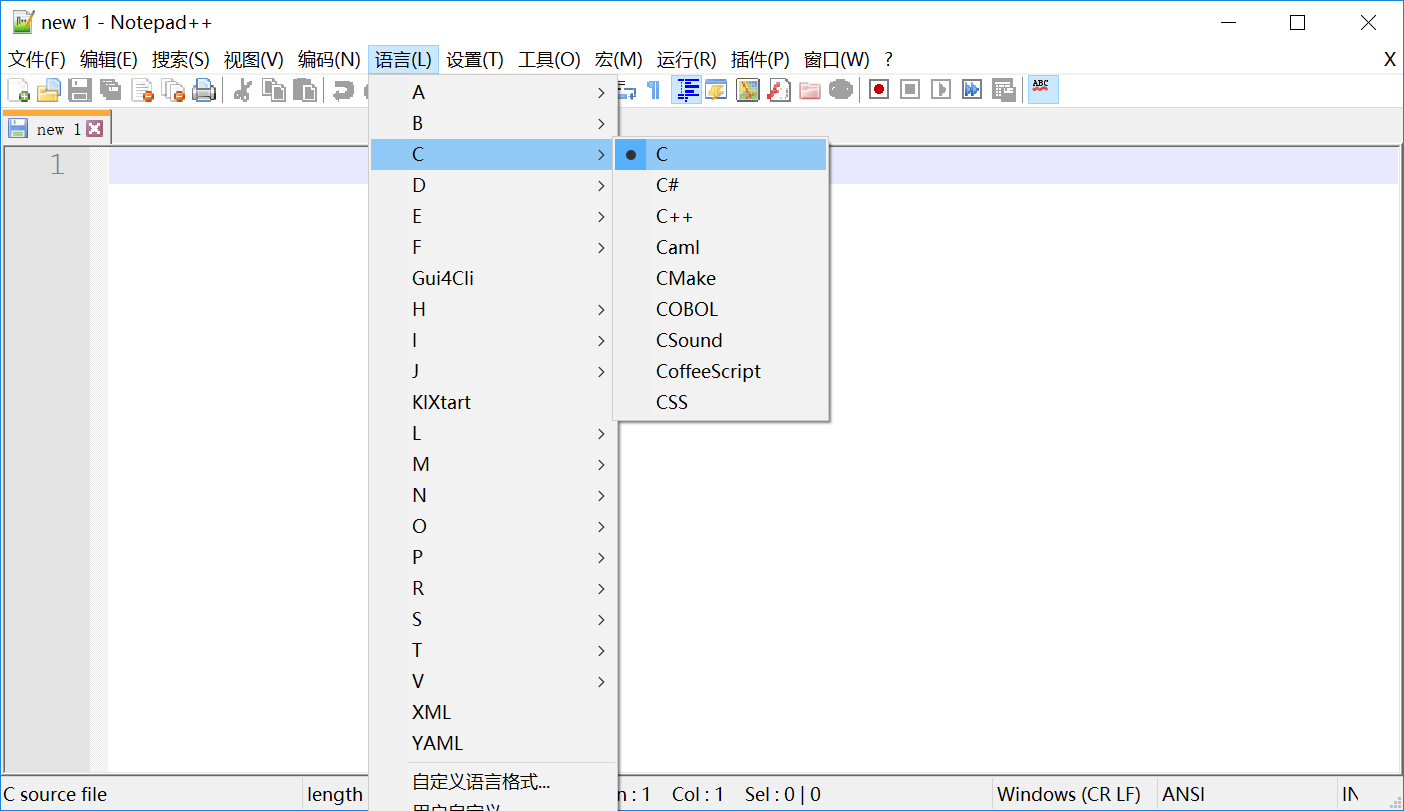
启动命令窗口：



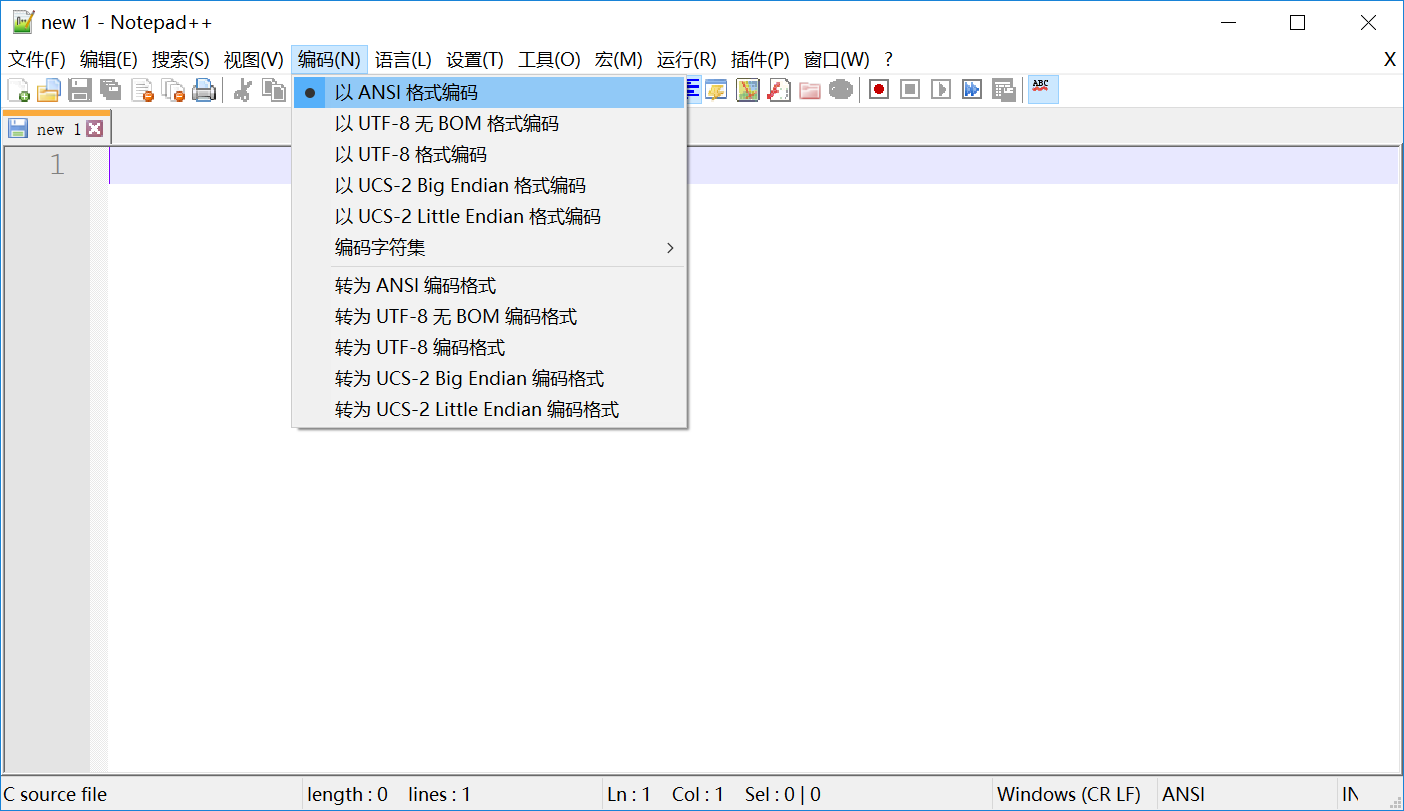
在运行窗口输入notepad++，再单击“确定”。



设置notepad++的语言为C语言：



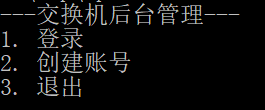
设置notepad++的编码为ANSI格式编码（便于再CMD中显示中文）



main.c

|  |
| --- |
| #include <stdio.h>  int main(void) {  // 打印登录菜单  printf("---交换机后台管理---\n");  printf("1. 登录\n");  printf("2. 创建账号\n");  printf("3. 退出\n");    return 0;  } |

测试效果：



## 项目精讲

### 头文件

#### 为什么要使用头文件

c语言的要求：函数使用前必须先“声明”（或者定义），否则编译器就不识别该函数。

printf函数的声明是在头文件stdio.h中。

#include <stdio.h> 表示把文件stdio.h中的所有内容拷贝到“这里”。

#### 头文件的查找路径

#include <stdio.h>

<>表示，从编译器默认的库路径中去找文件stdio.h

这个默认路径，取决于编译器。不同平台下不同编译器的路径都不相同。

这个默认路径下，已经包含了c标准库所需要的所有头文件。

#include “mytest.h”

“”表示从当前目录下寻找文件mytest.h

如果在当前目录下找不到，再从编译器默认的路径中查找。

### main函数

#### main函数的作用

main函数是程序的唯一入口。

也就是说，程序运行时，首先从main函数开始执行。

一个程序，必须要有一个main函数，而且也只能有一个main函数。

#### main函数的格式

格式1：

int main(void) {

//

}

格式2：

//具体用法在函数的参数部分，再讲解

int main(int argc , char\* argv) {

//

}

#### main函数的返回值

main函数应该用return返回一个int类型数据，也就是说，必须返回一个整数。

一般用法：

程序成功结束，则main函数返回0

程序有异常，则返回一个大于0的整数。

### printf函数

**作用**

用来向标准输出设备（默认是运行这个程序的终端，比如cmd窗口）打印信息。

**实例分析**

printf(“I love you!”);

打印 I love you

printf(“I love you!\nYou love me too!”);



\n表示换行

注意：\是一个“转义字符”，\n把n转义为“换行”

printf(“100\t200\t300”);



\t表示“水平制表符”，常用于对齐。

printf("姓名：%s 年龄: %d\n", "张三丰", 99);



%s表示是字符串（字符串，就是多个字符组成的一个序列）

%d 表示一个整数

printf("圆周率等于 %.3f", 3.1415926);



%f表示输出浮点数（带小数部分的实数）

%.3f表示小数点后保留3位小数，最后一位四舍五入

**注意：printf的其他用法先不要关注，以后需要使用时，再参考《C&C++函数手册.chm》。再次强调，学习C/C++时，只要重点掌握主要用法，很多生僻的用法不用关注。最重要的是编程思维、编程能力。**

### 常见错误

### C程序的编译方法

为什么要编译：

程序员写的代码，属于“高级语言”，计算机不识别。计算机只能识别0和1.

所以，需要把源代码，“转换”成计算机能够识别的文件。

编译方法：

gcc hello.c -o result

对源程序hello.c进行编译，输出的可执行文件是result

（对于Windows平台的编译器gcc, 输出的可执行文件是result.exe，自动添加扩展名.exe）

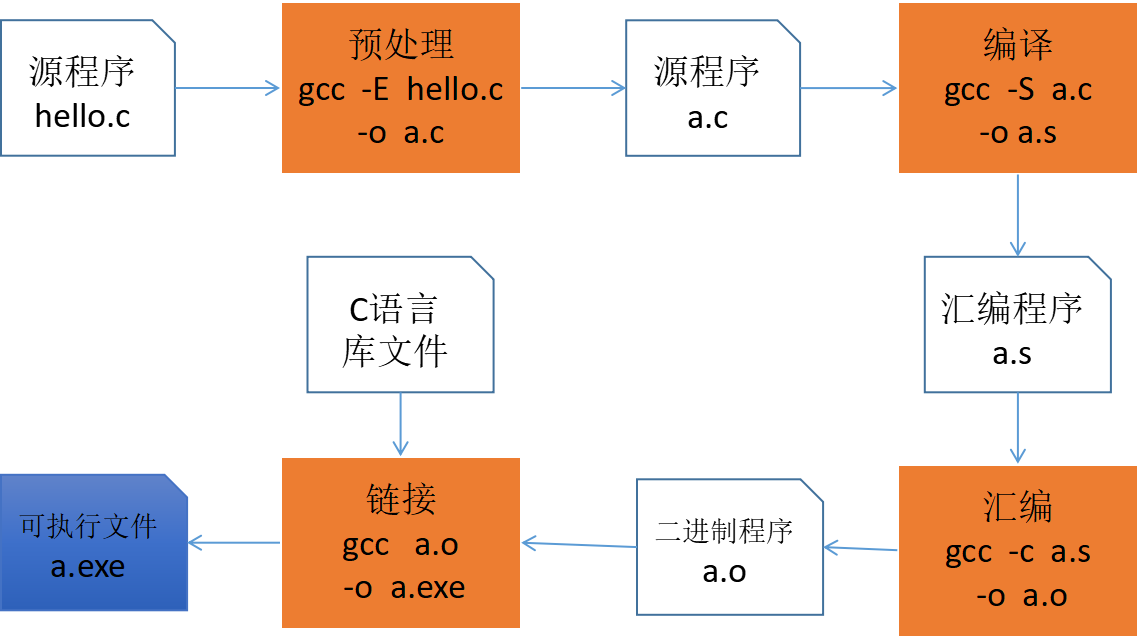
gcc，是编译器，也就是专门用来加工源程序的工具。

gcc hello.c

在windows平台，等效于：gcc hello.c -o a.exe

在linux平台，等效于：gcc hello.c -o a.out

### C程序的编译过程



**预处理**

把程序员写的源代码，进行“预加工”：

1）把#include包含的文件内容拷贝到这里

2）把宏替换成对应的内容（宏的使用，后面再讲）

3）其他预处理

预处理以后，得到的还是源程序！

**编译**

把预处理以后的源程序，加工成“汇编程序”。

汇编程序，是使用“汇编语言”编写的程序。

汇编语言，是一种“低级语言”，直接控制计算机的CPU，内存等。

**汇编**

把汇编程序，加工成二进制程序。

二进制程序，全部由0和1组成。是给计算机“阅读”的程序。

**链接**

把二进制程序，和所需的“库文件”，“组合加工”成计算机可以直接执行的文件。

### C程序的注释

注释是为了让程序更方便阅读。

有两种注释方式

1. 单行注释 //
2. 多行注释 /\* \*/

注意：不要为了注释而注释！

## 项目练习

1. 独立实现该项目，并编译执行。
2. 在登录菜单中，添加1个菜单项“删除账号”
3. 编写一个c程序，能够输出如下信息



# 项目3 交换机后台管理之用户输入

## 项目需求

用户登录时，需要输入用户名和密码。

## 项目实现

|  |
| --- |
| #include <stdio.h>  int main(void) {  // 定义变量，用来表示用户名和密码  char name;  int password;    // 输入用户名和密码  printf("请输入用户名：");  scanf("%c", &name);  printf("请输入密码：");  scanf("%d", &password);    /\*  // 打印登录菜单  printf("---交换机后台管理---\n");  printf("1. 登录\n");  printf("2. 创建账号\n");  printf("3. 退出\n");  \*/  return 0;  } |

## 项目精讲

### C语言的数据类型

在C语言中，任何数据都有一个确定的类型。

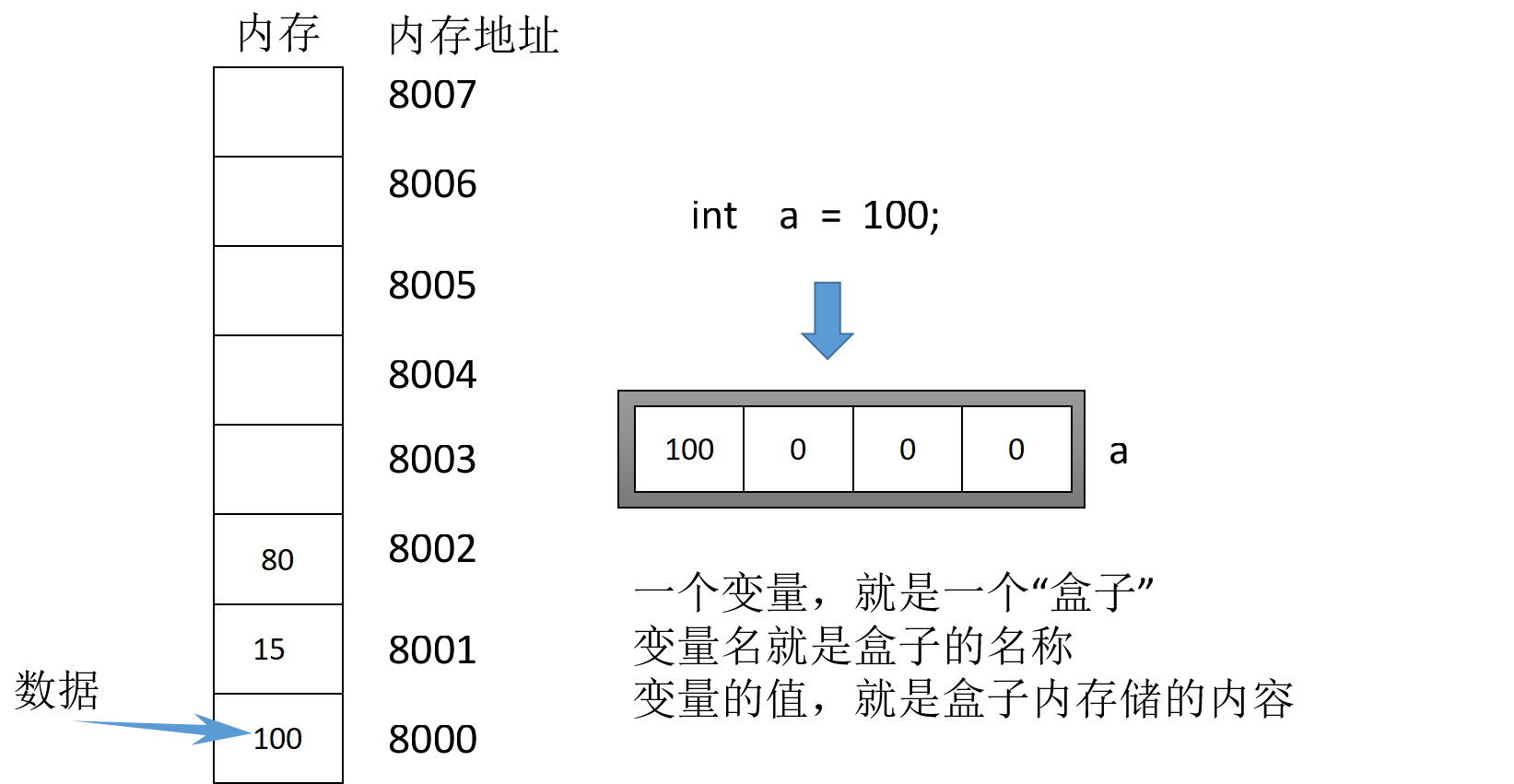
C语言，是一种“强类型”语言。

### 变量

**变量是什么**

变量，不是数学中的变量。

变量，是一个内存中的一块存储空间，即一小块内存。



**为什么要使用变量**

程序在运行时，需要保存很多内容常常变化的数据。

比如，射击类游戏不断变化的“分数”。

**内存的存储单位-“字节”**

内存的记本存储单位，是字节。

一个字节，包含8位二进制位。

**变量名的命名规范**

只能包含3种字符（数字、大/小写字母，下划线）

不能以数字开头（即，只能以字母或下划线开头）

不能和“关键字”同名（c语言内部已经使用的“名称”），比如类型名int

变量名的最大长度，C语言没有规定。

最大长度限制，取决于编译器，一般都在32以上。

变量名，最好“顾名思义”，不用使用汉语拼英！

比如：用name表示姓名，用power表示功率。

变量命令的风格:

int student\_age;

int studentAge;

### 常用的数据类型

**字符类型char**

一个字节。

用来存储小范围的整数（-128 ~ 127），和“字符”（所有ASCII字符，128个)。

char a = 100;

char b = ‘a’;

**整数类型int**

4个字节

用来存储整数，范围：- (2的31次方) ~ 2的31次方-1

**长整形long**

long 也就是 long int

用来存储整数。

在32位系统上，占4个字节，和int相同

在64位系统上，占8个字节。

**长长整形long long**

用来存储整数。

8字节。

**float类型（单精度浮点类型）**

用来存储带小数部分的数据。

4个字节

表示方式：按科学记数法存储，也就是需要存储“尾数”和“指数”

float x = 1.75E5;

//1.75E5就是1.75乘以10的5次方，只需保存尾数（1.75)和指数(5)

float y = 1.123456789;

//精度只能取值到 1.1234568, 在第7位（整数部分不算）是四舍五入后的值。

表示范围：-3.4\*10^38～+3.4\*10^38 （不需记忆）

精度：最长7位有效数字（是指7位10进制位）

**double类型（双精度浮点类型）**

用来存储带小数部分的数据。

8个字节

表示范围：-1.7\*10^308~1.7\*10^308（不需记忆）

精度：最长16位有效数字（是指16位10进制位）

### 变量的定义

实例：

int x ; //定义了一个变量，变量名是x, 属于int类型。

//注意，此时还没有确定的值

int y = 100; //定义了一个int类型变量，变量名是y, 变量的值是100

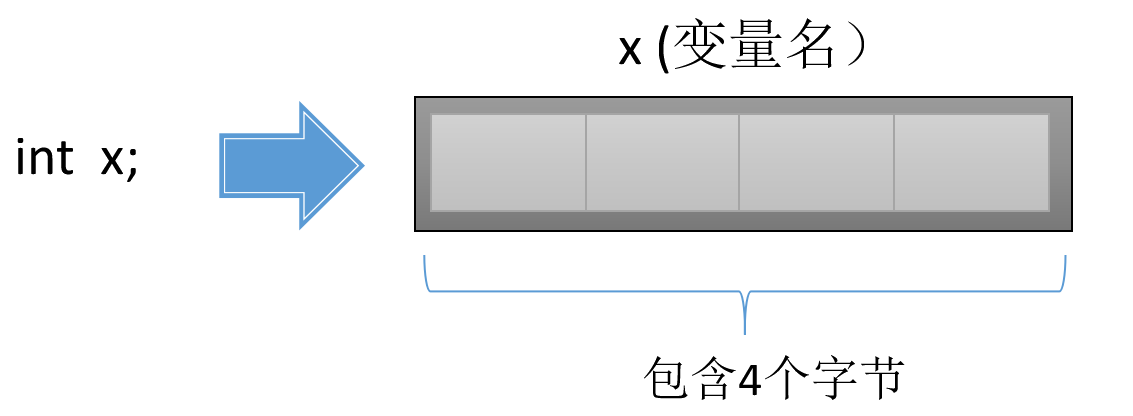
**理解：**

变量保存在内存中。

变量是一个“盒子”

变量名是这个盒子的名称

变量的值，是盒子内存储的物品

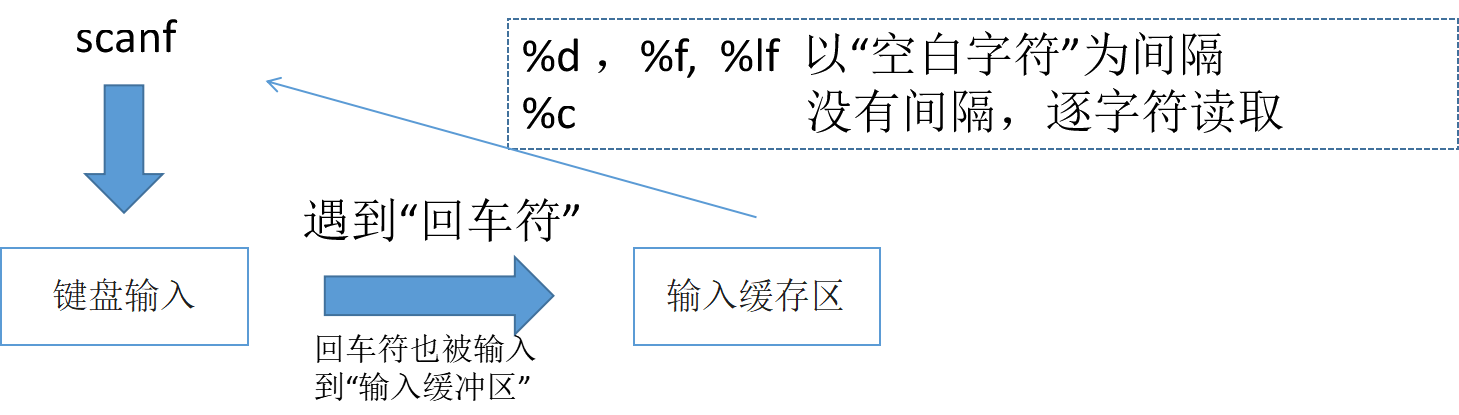


语法：

变量类型 变量名；

### 使用scanf输入数据

**输入机制**



空白字符有：空格，制表符(\t), 回车符

**char变量的输入**

scanf("%c", &c); //输入 100 ，实际只读了字符1

printf("%c\n", c); //输出 1

printf("%d\n", c); //输出49， 字符'1'的ASCII值就是49

**int变量的输入**

int a;

scanf("%d", &a); //输入100

printf("a=%d\n", a); //输出a=100

scanf("%d", &a); //输入a, 不是整数，输入失败！a依旧保持原来的值

printf("a=%d\n", a); //输出a=100

**char变量使用%d输入[ERROR]**

char a, b;

scanf("%d", &a); //输入 256 ，按4个字节存储，第2个字节保存到b变量中！

printf("a=%d,b=%d\n", a, b); //输出a=0,b=1

**float变量的输入**

scanf("%f", &x); //输入3.14

printf("x=%f", x); //输出 x=3.140000

**double变量的输入**

scanf("%lf", &x); //输入3.14

printf("x=%f", x); //输出 x=3.140000

printf("x=%lf", x); //输出 x=3.140000

输入double类型变量的值， 必须使用lf 而不是f, 否则输入失败。

输出double类型变量的值，则可以使用lf或者f, 没有区别。

**scanf注意：**

1. 当使用%s读取字符串串时，遇到空白字符（空格、制表符，回车符）就结束
2. 使用%c读取字符时，任何字符都能读取，但只读一个
3. 使用%d,%f,%lf 读取nt , float, double类型数据时，会自动转换成对应数据，

但是如果遇到其他字符（比如a）,就会读取失败

### char数据类型的其他输入输出函数

#### getchar

char c;

c = getchar(); //输入一个字符

printf("c=%c\n", c);

#### putchar

char c = 'a';

putchar(c); //输出1个字符， printf(“%c”, c);

#### getc

char c;

c = getc(stdin); //输入一个字符， stdin表示“标准输入设备”，默认是键盘

printf("c=%c\n", c);

#### putc

char c = 'b';

putc(c, stdout); //输出1个字符

## 项目练习

1. 独立实现该项目。
2. 让用户输入一个圆的半径，然后输出这个圆的面积和周长。
3. 让用户输入一个小写字母，然后输出对应的大写字母。

## 项目讨论

该项目存在的问题：

1. 用户名，只能输入一个字符，不能接收很长的用户名。
2. 密码很可能不是整数，很可能包含字母和数字。
3. 密码不能隐藏，都回显出来了，不安全。（在讲循环时再解决）

# 项目4 交换机后台管理之用户输入的优化

## 项目需求

用户登录时，用户可能输入很长的用户名。

## 项目实现

|  |
| --- |
| #include <stdio.h>  int main(void) {  // 定义变量，用来表示用户名和密码  //char name;  char name[32];  //int password;  char password[16];    // 输入用户名和密码  printf("请输入用户名：");  //scanf("%c", &name);  scanf("%s", name);  printf("请输入密码：");  //scanf("%d", &password);  scanf("%s", password);    return 0;  } |

## 项目精讲

### 什么是字符串

**什么是字符串**

字符串就是0个或多个“字符”组成的“有序”序列。

**字符串长度**

字符串包含的字符个数。

**字符串结束符**

在c语言中，为了便于存储字符串，要求在最后一个字符的后面存储一个0（一个字节）。

这个0， 称为“字符串结束符”，常用 ‘\0’ 表示。

“China” => ‘C’ ‘h’ ‘i’ ‘n’ ‘a’ ‘\0’

“” => ‘\0’

**字符串常量**

字符串常量，要求用“”扩起来。

printf("name=%s", "Rock"); //%s用来匹配字符串

**字符串的存储**

在c语言中，字符串是以“字符数组”存储的。

### 数组的基本概念

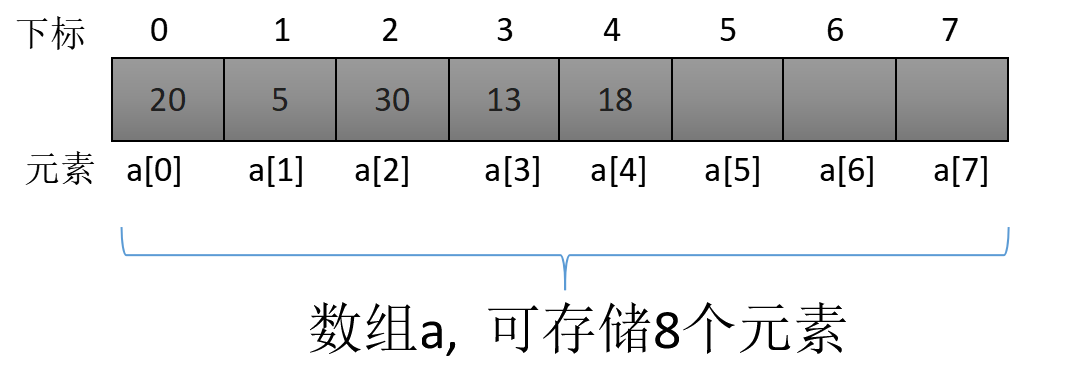
#### 什么是数组

数组，就是多个元素的“组合”。

每个元素的数据类型，必须相同。

每个元素在数组中有一个“下标”，用来表示它在数组中的“序号”，下标从0开始计算。

数组的“容量”是不能改变的。



#### 数组的定义

数组和其他变量一样，需要先定义，再使用。

实例：

int a[8]; //定义了一个数组，数组名是“a”，包含8个元素，每个元素是int类型的变量

a[0] = 20;

a[1] = 5;

printf(“%d”, a[1]);

char num[8];

#### 数组的初始化

在定义数组的同时，设置数组内的元素值。

int a[8] = {20, 5, 30, 13, 18};

printf("%d,%d,%d,%d,%d\n", a[0],a[1],a[2],a[3],a[4]);

int a[8] = {0}; //把数组的所有元素都初始化为0

printf("%d,%d,%d,%d,%d\n", a[0],a[1],a[2],a[3],a[4]);

int a[8] = {1}; //把a[0]初始化为1，其它值都初始化为0

int b[8] = {1, 5}; //把a[0]初始化为1，a[1]初始化为5，其它值都初始化为0

int a[] = {1,2,5}; //定义数组a, 这个数组包含3个元素！

// 根据“初始化列表”，自动计算数组的容量

**常见错误**

int a[2] = {1,2,5}; //错误！初始值太多，大于数组的容量

int a[3];

a = {1, 2, 3}; //不能对数组名直接赋值！

int a[3];

a[3] = 10; //下标越界！下标的取值范围是 0, 1, 2

#### 数组的内元素的访问

通过下标访问对应的元素。

特别注意， 数组的第一个元素的下标是0， 而不是1

int a[10];

a[5] = 200;

printf(“%d\n”, a[5]);

#### 数组的越界

数组的越界， 是指下标超出正常的范围！

例如：

int a[10]; //a[-1] 和 a[10]都是越界！

**越界的后果**

越界非常危险，可能导致数据破坏，或其他不可预期的后果！

**越界的控制**

需要程序员自己手动控制，编译器不做任何检查！因为，C语言完全信任程序员！

### 字符串的存储

字符串，是通过“字符数组”(元素类型为char的数组）来存储的！

demo1

|  |
| --- |
| char name[10];  name[0] = 'R';  name[1] = 'o';  name[2] = 'c';  name[3] = 'k';  name[4] = 0; //字符串结束符0，就是 '\0'  printf("姓名：%s", name); //    name[2] = 0;  printf("姓名：%s", name); // |

demo2

|  |
| --- |
| char name[10] = "Rock"; //相当于char name[10] = {'R', 'o', 'c', 'k', '\0'};  printf("姓名：%s", name); |

demo3

|  |
| --- |
| char name[] = "Rock"; //相当于：name[5] = "Rock"  printf("%d", sizeof(name)); //5 |

### 字符串的输入

#### scanf

从第一个非空白字符开始读取，直到遇到空白字符为止（不包含空白字符）

demo1

|  |
| --- |
| char name[16];  scanf("%s", name); //输入 Rock polo  printf("%s", name); //输出 Rock |

缺点：

1. 不能读取空格、制表符，因为被当作“分隔符”处理了！
2. 可能导致越界。

#### gets

遇到回车符结束，相当于读一行，但是不包括行末的回车符。

demo2

|  |
| --- |
| char name[16];  gets(name); // 输入 Rock polo  printf("%s", name); // 输出 Rock polo |

特点：

1. 能够读取 空格、制表符，但是不读取回车符。
2. 使用方便。

#### fgets

读取一行，直到遇到回车符。

demo1

|  |
| --- |
| char name[8];    fgets(name, 8, stdin);  //最多读8个字符，除去末尾的字符串结束符，实际最多只有7个字符  //输入1234567890 name的值为："1234567\0"  //输入12345 name的值为："12345\n\0" |

特点：

1. 安全！

当输入数据太多时，就只读取（第二个参数 -1）个字符

1. 回车符也被读到字符串（除非输入数据太多）

注意：

fgets函数的第2个参数，常常使用sizeof

demo2

|  |
| --- |
| char name[8];  fgets(name, sizeof(name), stdin); |

sizeof(name)表示, 数组name在内存中占用多少个字节。

### 字符串的输出

printf使用%s

实例：略

### 常用的字符串函数

#### 计算字符串的长度

|  |
| --- |
| #include <stdio.h>  #include <string.h>  int main(void) {  char name[] = "Rock";  printf("len = %d\n", strlen(name)); //len = 4    return 0;  } |

#### 字符串拷贝strcpy

|  |
| --- |
| #include <stdio.h>  #include <string.h>  int main(void) {  char name1[] = "Rock";  char name2[32];    strcpy(name2, name1); //把字符串name1拷贝到字符串name2  printf("name2=%s", name2);    return 0;  } |

strcpy的特点：

把源字符串的“字符串结束符”也一同拷贝到目的字符串中

strcpy的缺点：

可能导致字符串越界！不安全

#### 字符串拷贝strncpy

demo1

|  |
| --- |
| #include <stdio.h>  #include <string.h>  int main(void) {  char str1[10] = "123456789";  char str2[10] = "abcdefghi";    strncpy(str1, str2, 3); //从str2拷贝3个字符到str1  printf("str1=%s", str1); //str1=abc456789    return 0;  } |

demo2

|  |
| --- |
| #include <stdio.h>  #include <string.h>  int main(void) {  char str1[10] = "123456789";  char str2[10] = "ab";    strncpy(str1, str2, 5); //从str2拷贝5个字符到str1  printf("str1=%s", str1); //str1=ab  //str1数组的值：'a','b',0,0,0,'6','7','8','9', 0    return 0;  } |

当strncpy的第3个参数，大于拷贝源（第二个参数）的长度+1时，则把不足部分用0填充！！！

#### 字符串连接strcat

demo

|  |
| --- |
| #include <stdio.h>  #include <string.h>  int main(void) {  char dest1[64];  char dest2[64];    printf("请输入您的省份: ");  gets(dest1);  printf("请输入您的城市: ");  gets(dest2);    strcat(dest1, dest2);    printf("str1: %s\n", dest1);  printf("str2: %s\n", dest2);    return 0;  } |

还有很多其它字符串函数，需要用的时候，再学习。

## 项目练习

1. 独立实现该项目。
2. 要用户输入5个整数，保存到一个数组中。然后计算这个数组的平均值。
3. 要求用户输入一个人的姓名，然后输出这个字符串的长度。

## 项目讨论

该项目存在的问题：

当用户输入的用户名超过31个字符时，导致数组越界。

当用户输入的密码超过15个字符时，导致数组越界。

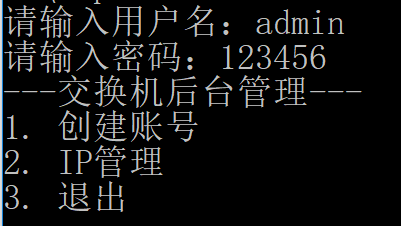
# 项目5交换机后台管理之权限判断

## 项目需求

判断用户名和密码是否正确。

## 项目实现

|  |
| --- |
| #include <stdio.h>  int main(void) {  // 定义变量，用来表示用户名和密码  //char name;  char name[32];  //int password;  char password[16];    // 输入用户名和密码  printf("请输入用户名：");  scanf("%s", name);  printf("请输入密码：");  scanf("%s", password);    if (strcmp(name, "admin") == 0 &&  strcmp(password, "123456") == 0) {  // 打印功能菜单  printf("---交换机后台管理---\n");  printf("1. 创建账号\n");  printf("2. IP管理\n");  printf("3. 退出\n");  } else {  printf("用户名或密码错误!\n");  }    return 0;  } |



## 项目精讲

### 字符串的比较运算

可参考c/c++手册

百度网盘链接：<https://pan.baidu.com/s/1dZJLwE>

#### 使用strcmp函数

#include <string.h>

int strcmp( const char \*str1, const char \*str2 );

比较规则：

按顺先从前往后比较

同序号的字符按“ASCII”码值比较

直到遇到对应字符不等或者字符串结束

返回值：

str1 < str2时， 返回值< 0（有些编译器返回 -1）

str1 > str2时， 返回值> 0（有些编译器返回 1）

str1 等于 str2时， 返回值== 0

demo

|  |
| --- |
| #include <stdio.h>  int main(void) {  char name[32];  int ret;    printf("请输入您的姓名：");  scanf("%s", name);    ret = strcmp(name, "Rock");  printf("ret=%d\n", ret);    return 0;  } |

#### 使用strncmp函数

#include <string.h>

int strncmp( const char \*str1, const char \*str2, size\_t count );

最多比较字符串str1和str2的前count个字符。

demo

|  |
| --- |
| #include <stdio.h>  #include <string.h>  int main(void) {  char name1[32] = "Rock";  char name2[32];  int ret;    fgets(name2, sizeof(name2), stdin); //输入Rock    ret = strcmp(name1, name2);  printf("ret=%d\n", ret);    //  ret = strncmp(name1, name2, strlen(name1));  printf("ret=%d\n", ret);    return 0;  } |

### 其他数据类型的比较运算

char, int, float, double数据的比较都使用：

大于： >

大于或等于: >=

小于： <

小于或等于： <=

不等于： !=

等于： == （注意：不是 = ）

比较运算的结果：（**逻辑值**）

结果为“真”： 1

结果为“假”： 0

|  |
| --- |
| #include <stdio.h>  int main(void) {  int a = 100;  int b = 200;  int ret;    ret = a > b;  printf("ret=%d\n", ret); //ret=0    ret = a < b;  printf("ret=%d\n", ret); //ret=1  return 0;  } |

比较运算的使用场合：

用于“条件判断”

### C语言的布尔类型

#### C语言主要标准

|  |  |
| --- | --- |
| 版本 | 说明 |
| C89 | 两者差别很小，一般都统称为C89  “老版本”  变量的声明必须放在语句的开头 |
| C90 |
| C99 | 支持布尔类型  支持运行时才确定数组的长度 |
| C11 | 最新版本 |

#### C89标准中的逻辑值

使用0和1表示逻辑值

|  |  |
| --- | --- |
| 真 | 1 |
| 假 | 0 |
| 非0值 | 真 |

demo

|  |
| --- |
| #include <stdio.h>  int main(void) {  int a = 100;  int b = 200;  int ret; //或者 char ret;    ret = a > b;  if (ret) {  printf("a > b\n");  } else {  printf("a <= b\n");  }    return 0;  } |

#### C99标准中的逻辑值（兼容C89）

使用bool类型表示逻辑类型

使用 true 表示真

使用 false表示假

注意：需要包含头文件 stdbool.h

|  |
| --- |
| #include <stdio.h>  #include <stdbool.h>  int main(void) {  int a = 100;  int b = 200;  //int ret;  bool ret;    ret = a > b;  if (ret) { //即: if (ret == true)  printf("a > b\n");  } else {  printf("a <= b\n");  }    //true和false是"bool类型的常量"  printf("true=%d\n", true);  printf("false=%d\n", false);    return 0;  } |

注意：大部分C项目使用的是C89标准中的逻辑值表示方式。

### 逻辑运算

#### 逻辑与 &&

|  |  |  |
| --- | --- | --- |
| a | b | a && b |
| 真 | 真 | 真 |
| 假 | 假 | 假 |
| 假 | 真 | 假 |
| 真 | 假 | 假 |

都为真，逻辑与才是真

只要有一个是假， 逻辑与就是假

相当于“而且”

应用场景：

当需要两个条件都满足时，就使用逻辑与

**特别注意：**

条件1 && 条件2

当条件1为真时，才去判断条件2

当条件1为假时，就不再判断条件2

|  |
| --- |
| #include <stdio.h>  int main(void) {  int x = 0;  int a;    printf("请输入一个整数：");  scanf("%d", &a);    if ((a > 5) && ((x=100) > 90)) {  printf("OK\n");  }    printf("x=%d\n", x);    return 0;  } |

#### 逻辑或 ||

|  |  |  |
| --- | --- | --- |
| a | b | a || b |
| 真 | 真 | 真 |
| 假 | 假 | 假 |
| 假 | 真 | 真 |
| 真 | 假 | 真 |

都为假，逻辑与才是真

只要有一个是真， 逻辑与就是真

相当于“或者”

应用场景：

只需要满足任意一个条件时，就使用逻辑或

**特别注意：**

条件1 || 条件2

当条件1为真时，才不再判断条件2

当条件1为假时，才判断条件2

|  |
| --- |
| #include <stdio.h>  int main(void) {  int x = 0;  int a;    printf("请输入一个整数：");  scanf("%d", &a);    if ((a > 5) || ((x=100) > 90)) {  printf("OK\n");  }    printf("x=%d\n", x);    return 0;  } |

#### 逻辑非 !

|  |  |
| --- | --- |
| a | !a |
| 真 | 假 |
| 假 | 真 |

相当于“不”

应用场景：

当需要不满足某条件时，就使用逻辑或

**特别注意：**

逻辑非，只对一个条件进行运算！

是一种“单目运算符”

|  |
| --- |
| #include <stdio.h>  int main(void) {  int age;    printf("请输入您的年龄: ");  scanf("%d", &age);    //特别注意要使用()  //if ( ! age >= 30) 将导致非预期结果, ！会和age结合  if ( !(age >= 30) ) {  printf("您还不到30\n");  } else {  printf("您已过而立之年！\n");  }    return 0;  } |

### 其它运算操作

#### 算术运算

|  |  |  |  |
| --- | --- | --- | --- |
| + | 加 | x = 5 + 3; |  |
| - | 减 | x = 5 - 3; |  |
| \* | 乘 | x = 5 \* 3; |  |
| / | 除 | x = 10 / 3; | 1. 两个整数相除，结果取整数部分 2. 除数不能为0（也不要为0.0） |
| % | 取余 | x = 10 % 3;  //结果为1 | 1. 只有整数和char类型可以做%运算   float和double类型不能做%运算 |

#### 赋值运算

x = 10; //把x的值设置为10, 把10写到变量x中。

x = 10 + a;

左边必须是变量

“优先级”很低，只比 ","（逗号元素符）高。

x = (3 + 5); //先计算"+", 再计算“=”

#### 复合赋值运算

x += 10; // x = x + 10

x -= 10; // x = x - 10

类的还有： \*= , /=, %= 等。

#### 位运算

在后续章节中学习。

#### 自增自减运算

|  |  |  |  |
| --- | --- | --- | --- |
| 后缀自增 | x = i++; | x = i;  i=i+1; | 先取值  再自增（自减） |
| 后缀自减 | x=i--; | x =i;  i=i-1; |
| 前缀自增 | x = ++i; | i=i+1;  x=i; | 先自增（自减）  再取值 |
| 前缀自减 | x = --i; | i=i-1;  x = i; |

注意：

1. 只能对变量做++和--运算，不能对变量和表达式做++和--运算

5++; //ERROR

（3+x)++; //ERRO

1. 建议尽量使用前缀自增（自减），以避免错误。

#### 逗号运算符

优先级最低。

|  |
| --- |
| #include <stdio.h>  int main(void) {  int x;    // 先计算 x = 3+5, 再计算3\*5  x = 3+5, 3\*5, 10/5;  printf("x=%d\n", x); //x=8    //取最后一个表达式的值，作为整个“逗号表达式”的值  x = (3+5, 3\*5, 10/5);  printf("x=%d\n", x); //x=2    return x;  } |

#### 三目运算符

条件 ? 表达式1 ：表达式2

如果条件为真，就取表达式1作为整个表达式的值

如果条件为假，就取表达式2作为整个表达式的值

|  |
| --- |
| #include <stdio.h>  int main(void) {  int year;  int holiday;    printf("请输入您的工作年限: ");  scanf("%d", &year);    holiday = year > 10 ? 20 : 5;  printf("您的年假有%d天\n", holiday);    return 0;  } |

### 类型转换

#### 类型转换的概念

为什么需要“类型转换”

参与运算的两个操作数的数据类型，必须相同！

类型转换的类别：

1. 隐式类型转换

自动完成转换！

1. 算数转换
2. 赋值转换
3. 输出转换
4. 强制类型转化

#### 算数转化

(+,-,\*,/,%)

char , int, long, long long, float, double

#### 赋值转换

|  |
| --- |
| #include <stdio.h>  int main(void) {  int x;  x = 3.14 \* 10; // 31.4 转换为int类型，因为赋值符号的左边变量的类型是int类型    printf("%d\n", x);    return 0;  } |

#### 输出转换

|  |
| --- |
| #include <stdio.h>  int main(void) {  printf("%c\n", 255+50); //305 -> 49 ('1');  printf("%d\n", 255+50);  return 0;  } |

int类型数据， 按照%f格式输出时，将得到错误的输出

float（或double) 类型数据，按照%d格式输出时，将得到错误的输出

#### 强制类型转化

|  |
| --- |
| #include <stdio.h>  int main(void) {  int x = 257 + 100;  printf("%d\n", x);    x = (char)257 + 100;  printf("%d\n", x);    return 0;  } |

### 运算符优先级

一共有15个级别！

不需强制记忆，只需要掌握以下常用的优先级：

最高优先级：( )和[ ]

倒数第二低优先级：赋值和复合赋值(=， +=, -= ...)

最低优先级：逗号表达式

！ > 算术运算符 > 关系运算符 > && > || > 赋值运算符

x = ! 3 + 4 < 5 && 6 > 7 || 8 > 7;

等效于：

x = ((!3 + 4 < 5) && (6 > 7)) || (8 > 7);

### if条件判断语句

demo1

|  |
| --- |
| #include <stdio.h>  int main(void) {  int salary;    printf("请输入你的期望年薪:");  scanf("%d", &salary);    if (salary >= 200000) {  printf("你需要精通C/C++开发\n");  }    printf("OK\n");    return 0;  } |

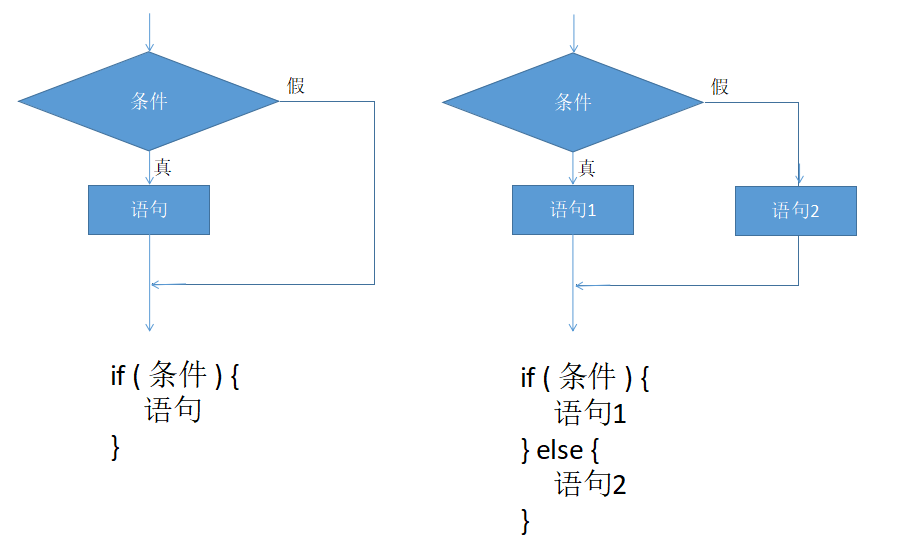
demo2

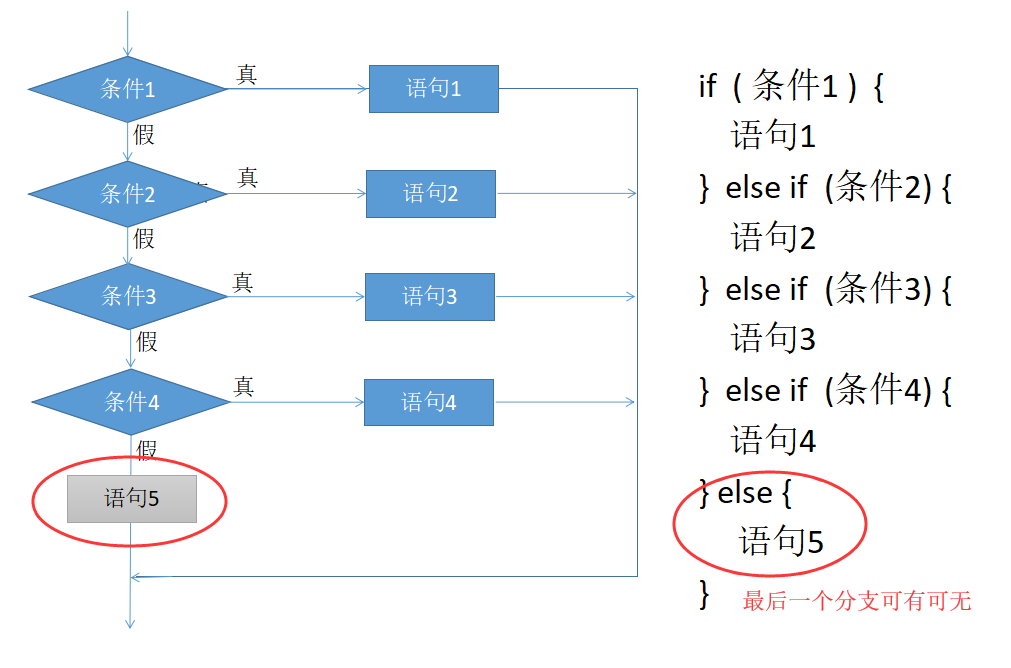
|  |
| --- |
| #include <stdio.h>  int main(void) {  char answer[16];    printf("你有房吗? ");  scanf("%s", answer);  if (strcmp(answer, "yes") == 0) {  printf("OK");  } else {  printf("你是一个好人!\n");  }    return 0;  } |

demo3

|  |
| --- |
| #include <stdio.h>  int main(void) {  char answer[16];    printf("有房吗? ");  scanf("%s", answer);  if (strcmp(answer, "yes") == 0) {  printf("有房，不错\n");  } else if (printf("有车吗? ") &&  scanf("%s", answer) &&  strcmp(answer, "yes")==0) {  printf("有车，还行\n");  } else if (printf("有病吗? ") &&  scanf("%s", answer) &&  strcmp(answer, "no")==0) {  printf("健康就好！\n");  } else {  printf("你是一个好人!\n");  }    return 0;  } |

### 流程图





## 项目练习

1. 独立实现该项目。
2. 让用户输入一个成绩，然后输出这个成绩的等级。

0-59: 不及格

60-79: 及格

80-89: 良好

90-100: 优秀

其它：非法成绩

# 项目6 交换机后台管理之重复输入用户名和密码

## 项目需求

解决项目5中存在的问题：

用户名和密码只能输入一次。如果输入错误，就没有机会重新输入。

## 项目实现

|  |
| --- |
| #include <stdio.h>  int main(void) {  // 定义变量，用来表示用户名和密码  char name[32];  char password[16];    //输入用户名和密码  while (1) {  // 输入用户名和密码  printf("请输入用户名：");  scanf("%s", name);  printf("请输入密码：");  scanf("%s", password);    if (strcmp(name, "admin") == 0 &&  strcmp(password, "123456") == 0) {  break;  } else {  printf("用户名或密码错误!\n");  system("pause");  system("cls");  }  }    system("cls");  // 打印功能菜单  printf("---交换机后台管理---\n");  printf("1. 创建账号\n");  printf("2. IP管理\n");  printf("3. 退出\n");  printf("请选择...");    return 0;  } |

## 项目精讲

### while循环

使用场合：

当需要反复执行某些“过程”时，就可以使用while循环。

使用方法

while (条件) {

语句

}

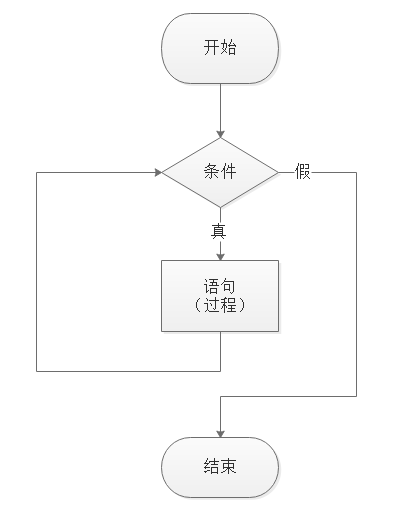
break的使用

死循环

有些场合（比如，游戏引擎的主循环, 就是一个死循环）

有些场合，是要避免死循环。

画流程图



### for循环

从功能上，for循环和while循环是完全等效的！

使用场合

在循环次数已经确定的情况下，使用for循环更方便！

使用方法

for (表达式1； 表达式2；表达式3）{

循环体

}

说明：

表达式1： 为循环做准备

表达式2： 循环条件

表达式3： 改变循环计数

注意：

表达式1、表达式2、表达式3, 这3个表达式的任意一个或多个，都可以省略！

但是其中的“；”不可以省略！

for (; ; ) {

循环体

}

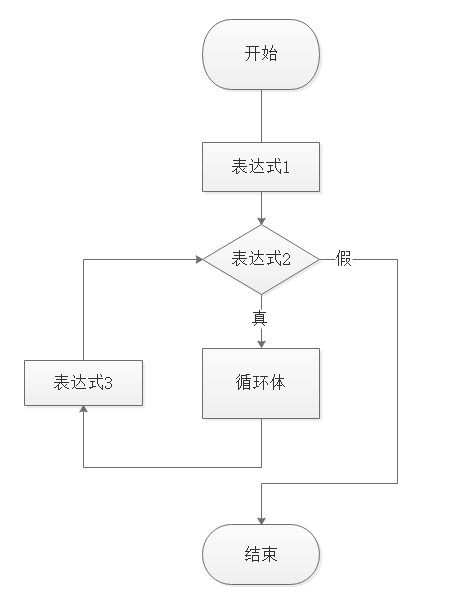
相当于：

while (1) {

循环体

}

流程图



for和while的选择

1. 当已经确定了循环次数时，建议使用for
2. 其他情况，可以使用for ，也可以使用while, 建议使用while

### do-while循环

使用场合:

先执行一次循环体，然后再判断条件，以判定是否继续下一轮循环！

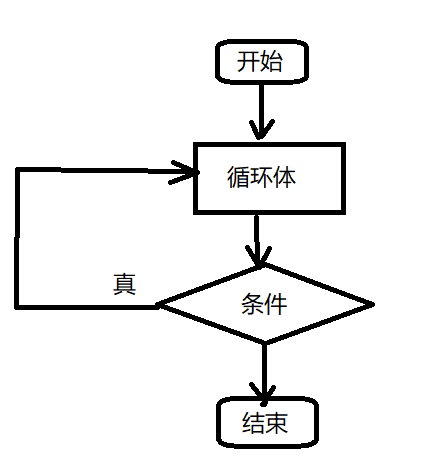
即：至少执行一次循环体！

使用方法

do {

循环体

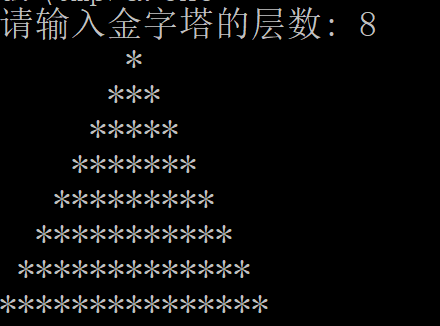
} while （条件）



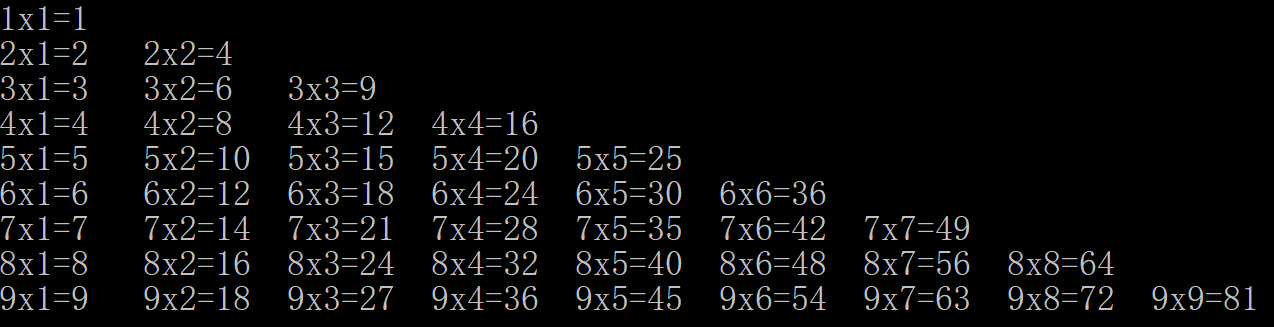
do {}while(0)的用法， 主要用于#define宏定义（后续课程讲解)

## 项目练习

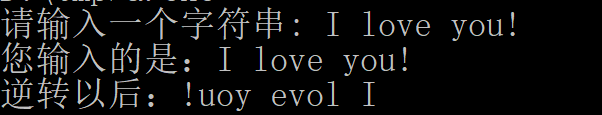
1. 独立完成项目6
2. 打印如下效果，具体的行数要用户输入。



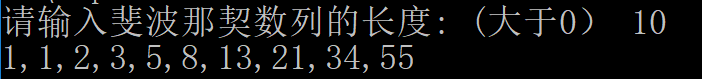
1. 打印乘法口诀表



1. 让用户输入一个字符串，然后把这个字符串“逆转”，并输出。



1. 打印斐波那契数列，具体的个数由用户输入。



# 项目7 换机后台管理之多用户账号登录

## 项目需求

实现多个账号

## 项目实现

|  |
| --- |
| #include <stdio.h>  #include <string.h>  int main(void) {  // 定义变量，用来表示用户名和密码  char name[32];  char password[16];  FILE \*file; //定义了一个文件指针变量，变量名是file  char line[128];  char name\_tmp[32];  char password\_tmp[16];  char \*ret;    //打开文件  file = fopen("users.txt", "r");  if (!file) { //等效于 file == NULL  printf("文件打开失败");  return 1;  }    //输入用户名和密码  while (1) {  // 输入用户名和密码  printf("请输入用户名：");  scanf("%s", name);  printf("请输入密码：");  scanf("%s", password);    /\*  if (strcmp(name, "admin") == 0 &&  strcmp(password, "123456") == 0) {  break;  } else {  printf("用户名或密码错误!\n");  system("pause");  system("cls");  }  \*/    //从文件中读取账号，并进行判断！  while (1) {  //读一行  ret = fgets(line, sizeof(line), file); //line: "admin 123456\n"  if (!ret) {  break;  }  sscanf(line, "%s %s", name\_tmp, password\_tmp);  if (!strcmp(name, name\_tmp) && !strcmp(password, password\_tmp)) {  break;  }  }    if (ret) { //用户名和密码匹配成功  break;  } else {  printf("用户名或密码错误!\n");  system("pause");  system("cls");    fseek(file, 0, SEEK\_SET); //把文件内部的位置指针设置到文件头  }  }    system("cls");  // 打印功能菜单  printf("---交换机后台管理---\n");  printf("1. 创建账号\n");  printf("2. IP管理\n");  printf("3. 退出\n");  printf("请选择...");    return 0;  } |

## 项目精讲

### fopen文件的打开操作

**函数原型**

#include <stdio.h>

FILE \*fopen( const char \*fname, const char \*mode );

参数1：fname 表示文件名（可以含有路径信息）

参数2：打开方式

返回值：FILE\* 文件指针，

如果打开失败，就返回NULL（就是0）

**mode 打开方式**

"r" 以“读”的方式打开一个文本文件（只能读）

"r+" 与"r" 的区别在于，增加了“写”

"rb" 以“读”的方式打开一个二进制文件（只能读）

"rb+" 与"rb"的区别在于，增加了“写”

"w" 以“写”的方式创建一个文本文件，如果这个文件已经存在，就会覆盖原来的文件

"w+" 与"w"的区别在于，增加了“读”

"wb" 以“写“的方式创建一个二进制文件

"wb+" 与"wb"的区别在于，增加了“读”

"a" 以“尾部追加”的方式打开一个文本文件, (只能写）

"a+" 以"a"的区别在于，增加了"读"

"ab" 以“尾部追加”的方式打开一个二进制文件, (只能写）

"ab+" 与"ab"的区别在于，增加了“读”

**小结：**

打开方式，共1到3个字符。

第一个字符是 r、w或a

r 表示“读”，用于打开已经存在的文件

w 表示“创建”, 用于创建一个新文件，并能够“写”

a 表示“尾部追加”，并能够"写"

b, 只能写在第二位，表示打开的是二进制文件

+，只能写在最后，表示增加一个读或写的功能

实例

|  |
| --- |
| #include <stdio.h>  int main(void) {  FILE \*file;    //file = fopen("users.txt", "r");  file = fopen("users1.txt", "r");  if (file != NULL) { //NULL就是0  printf("文件users.txt打开成功!\n");  } else {  printf("文件users.txt打开失败!\n");  }    return 0;  } |

### fclose文件的关闭操作

清理缓冲区，并释放文件指针。

Demo

|  |
| --- |
| #include <stdio.h>  int main(void) {  FILE \*file;    file = fopen("users.txt", "a");  fputs("\nxiaoxiao 123456", file);    **fclose(file);**  return 0;  } |

特别注意：

对文件执行写操作以后，并不会马上写入文件，而只是写入到了这个文件的输出缓冲区中！

只有当这个输出缓冲区满了，或者执行了fflush，或者执行了fclose函数以后，或者程序结束，

才会把输出缓冲区中的内容正真写入文件！

### fgetc文件的读操作

**函数原型：**

#include <stdio.h>

int fgetc( FILE \*stream );

返回值：成功时，返回读到的字符，返回的是int类型（实际值是字符）

失败或读到文件尾，返回EOF (就是-1)

**作用：**

从文件中读取一个字符

**实例：**

|  |
| --- |
| #include <stdio.h>  int main(void) {  FILE \*file;  char c;    file = fopen("users.txt", "r");    while ((c = fgetc(file)) != EOF) { //EOF就是 -1  printf("%c", c);  }    return 0;  } |

### fputc写一个字符到文件fputc

**函数原型：**

#include <stdio.h>

int fputc( int ch, FILE \*stream );

**实例：**

test.c

|  |
| --- |
| #include <stdio.h>  int main(void) {  FILE \*file1;  FILE \*file2;  char c;    file1 = fopen("test.c", "r");  file2 = fopen("test2.c", "w");    while ((c = fgetc(file1)) != EOF) { //EOF就是 -1  **fputc(c, file2);**  }    fclose(file1);  fclose(file2);    return 0;  } |

### fgets 从文件中读取一个字符串

**复习：**

在项目4的“字符串输入”中学习过。

**函数原型：**

#include <stdio.h>

char \* fgets( char \*str, int num, FILE \*stream );

参数:

num： 最多读取num-1个字符，或者遇到文件结束符EOF为止（即“文件读完了”）

返回值; 读取失败时， 返回NULL,

读取成功时，返回str

**实例：**

|  |
| --- |
| #include <stdio.h>  int main(void) {  FILE \*file1;  char tmp[64];    char c;    file1 = fopen("test.c", "r");    while (fgets(tmp, sizeof(tmp), file1) != NULL) {  printf("%s", tmp);  }    fclose(file1);  return 0;  } |

### fputs 写一个字符串到文件中去

**函数原型：**

#include <stdio.h>

int fputs( const char \*str, FILE \*stream );

**实例**

|  |
| --- |
| #include <stdio.h>  int main(void) {  FILE \*file1;  FILE \*file2;  char tmp[64];    char c;    file1 = fopen("test.c", "r");  file2 = fopen("test2.c", "w");    while (fgets(tmp, sizeof(tmp), file1) != NULL) {  fputs(tmp, file2);  }    fclose(file1);  fclose(file2);  return 0;  } |

### fprintf 往文件中写格式化数据

**函数原型：**

#include <stdio.h>

int fprintf( FILE \*stream, const char \*format, ... );

Demo:

|  |
| --- |
| #include <stdio.h>  int main(void) {  FILE \*file1;  char name[32];  int age;  char c;    file1 = fopen("info.txt", "w");    while (1) {  printf("请输入学员姓名：");  scanf("%s", name);  printf("请输入%s的成绩: ", name);  scanf("%d", &age);    **fprintf(file1, "姓名:%s\t\t年龄:%d\n", name, age);**    printf("还需要继续输入吗？ Y/N\n");    //fflush(stdin);  while((c=getchar()) != '\n'); //直到读到回车符为止！    scanf("%c", &c);  if (c == 'Y' || c == 'y') {  continue;  } else {  break;  }  }    fclose(file1);  return 0;  } |

### fscanf 格式化读取文件中数据

**函数原型：**

#include <stdio.h>

int fscanf( FILE \*stream, const char \*format, ... );

返回值：成功时，返回实际读取的数据个数

失败时，返回 EOF （-1）

匹配失败时，返回0

Demo

|  |
| --- |
| #include <stdio.h>  int main(void) {  FILE \*file1;  char name[32];  int age;  int ret;    file1 = fopen("info.txt", "r");    while (1) {  **ret = fscanf(file1, "姓名:%s 年龄:%d\n", &name, &age);**  if (ret == EOF) {  break;  }    printf("%s,%d\n", name, age);  }    fclose(file1);  return 0;  } |

### fwrite 以二进制形式写数据到文件中去

#include <stdio.h>

int fwrite( const void \*buffer, //要写入的数据的其实地址，也就是变量的地址

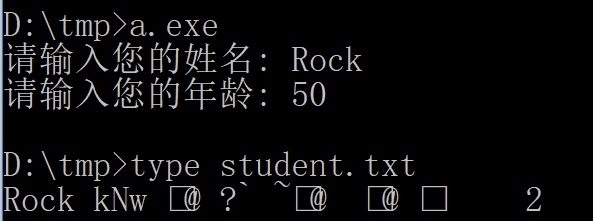
size\_t size, //每“块”数据的大小

size\_t count, //写入几块数据

FILE \*stream );

Demo

|  |
| --- |
| #include <stdio.h>  #include <string.h>  int main(void) {  FILE \*file1;  char name[32];  int age;  int ret;    file1 = fopen("info.txt", "wb");    printf("请输入您的姓名: ");  gets(name);  printf("请输入您的年龄: ");  scanf("%d", &age);    fwrite(name, sizeof(name), sizeof(char), file1);  fwrite(&age, 1, sizeof(int), file1);    fclose(file1);  return 0;  } |



**补充：**

w和wb的区别

wb的demo

|  |
| --- |
| #include <stdio.h>  #include <string.h>  int main(void) {  FILE \*file1;  char info[] = "Hello\nWorld";  int age;  int ret;    file1 = fopen("test.txt", "**wb**");    fwrite(info, sizeof(char), strlen(info), file1);    fclose(file1);  return 0;  } |



w的demo

|  |
| --- |
| #include <stdio.h>  #include <string.h>  int main(void) {  FILE \*file1;  char info[] = "Hello\nWorld"; // \n 保存位 \r\n  int age;  int ret;    file1 = fopen("test.txt", "**w**");    fwrite(info, strlen(info), sizeof(char), file1);    fclose(file1);  return 0;  } |



小结：

在windows平台下，

当使用w方式打开文件时，

如果使用fwrite写入数据时，会把’\n’写入为 ‘\r’’\n’

即把10保存为 13 10

因为，在windows平台下，文本文件中的回车符\n，会保存为 \r\n

( \n的ASCII码为10， \r的ASCII码为13）

当使用wb方式打开文件时，

如果使用fwrite写入数据时，遇到’\n’仍只写入为 ‘\n’

### fread 以二进制形式读取文件中的数据

函数原型：

#include <stdio.h>

int fread( void \*buffer, size\_t size, size\_t num, FILE \*stream );

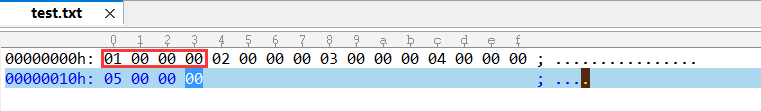
Demo

|  |
| --- |
| #include <stdio.h>  #include <string.h>  int main(void) {  FILE \*file1;  char name[32];  int age;  int ret;    file1 = fopen("student.txt", "rb");      fread(name, sizeof(name), sizeof(char), file1);  fread(&age, 1, sizeof(int), file1);    printf("%s, %d\n", name, age);    fclose(file1);  return 0;  } |

### putw 以二进制形式存贮一个整数

demo

|  |
| --- |
| #include <stdio.h>  #include <string.h>  int main(void) {  FILE \*file1;  int data[] = {1,2,3,4,5};  int i;    file1 = fopen("test.txt", "w");    for (i=0; i<5; i++) {  putw(data[i], file1);  }    fclose(file1);    return 0;  } |



### getw 以二进制形式读取一个整数

**函数原型：**

int getw(FILE \*fp)

返回值：成功时返回读取到的值

失败时返回-1。

Demo

|  |
| --- |
| #include <stdio.h>  int main(void) {  FILE \*file;  int value;    file = fopen("test.data", "rb");  if (!file) {  printf("文件打开失败!\n");  return 1;  }    while (1) {  value = getw(file);  if (value == -1 && feof(file)) {  break;  }    printf("%d ", value);  }    fclose(file);    return 0;  } |

### 文件状态检查函数

#### feof 文件结束

函数原型：

#include <stdio.h>

int feof( FILE \*stream );

返回值：如果指定的程序，已经到达文件末尾位置，就返回非零值（真）。

|  |
| --- |
| #include <stdio.h>  int main(void) {  FILE \*file;  char c;    file = fopen("test.c", "r");    //while ((c = fgetc(file)) != EOF) { //EOF就是 -1  while (!feof(file)) {  c = fgetc(file);  printf("%c", c);  }    return 0;  } |

#### ferror 文件读/写出错

|  |
| --- |
| #include <stdio.h>  int main(void) {  FILE \*file;  char c;  int ret;    file = fopen("test.c", "r");    fputc('A', file);    if (ferror(file)) {  perror("文件file发生错误");  }      return 0;  } |

执行结果：



把 “r” 改为 “r+” 就不会发生错误了。

#### clearerr 清除文件错误标志

函数原型：

#include <stdio.h>

void clearerr( FILE \*stream );

Demo

|  |
| --- |
| #include <stdio.h>  int main(void) {  FILE \*file;  char c;  int ret;    file = fopen("test.c", "r");    fputc('A', file);  if (ferror(file)) {  perror("文件file发生错误");  }    //如果不清除文件错误，以后读写文件时, 即使没有发生错误，ferror仍将返回非零值（认为还有错）  clearerr(file);    c = fgetc(file);  printf("c=%c\n", c);  if (ferror(file)) {  perror("文件file发生错误");  }    return 0;  } |

#### ftell 获取文件指针的当前位置

**函数原型：**

#include <stdio.h>

long ftell( FILE \*stream );

Demo

|  |
| --- |
| #include <stdio.h>  int main(void) {  FILE \*file;  char c;  int ret;  long offset;    file = fopen("test.c", "r");    offset = **ftell**(file);  printf("当前位置是： %ld\n", offset);    fgetc(file);  offset = **ftell**(file);  printf("当前位置是： %ld\n", offset);    fclose(file);    return 0;  } |

### 文件定位函数

注意：文件始终只能从当前的位置向文件尾方向读写！

#### fseek 随机定位

**函数原型：**

#include <stdio.h>

int fseek( FILE \*stream, long offset, int origin );

参数2：

偏移量，可正可负。

<0 向文件头方向偏移

>0 向文件尾方向偏移

参数3：

SEEK\_SET 从文件的开始位置定位， 此时参数2必须大于0

SEEK\_CUR 从文件的结束位置定位

SEEK\_END 从文件的结束位置定位， 此时参数2必须小与0

**Demo**

|  |
| --- |
| #include <stdio.h>  int main(void) {  FILE \*file;  char c;  char buff[256];  int i;  file = fopen("test.c", "r");    //读取文件最后10个字符  fseek(file, -10, SEEK\_END);  while (!feof(file)) {  c = fgetc(file);  printf("%c", c);  }    //读取文件的第一行  fseek(file, 0, SEEK\_SET);  fgets(buff, sizeof(buff), file);  printf("\n第一行：%s\n", buff);    //读取当前位置的前10个字符  fseek(file, -10, SEEK\_CUR);  printf("\n这10个字符是：");  for (i=0; i<10; i++) {  c = fgetc(file);  printf("%c", c);  }    close(file);  return 0;  } |

#### rewind 反绕

把文件的位置指针定位到开始位置。

rewind(file)

等效于：

fseek(file, 0, SEEK\_SET)

## 项目练习

1. 练习1

独立实现项目7.

1. 编写一个程序，统计该程序本身一共有多少个字符，有多少行，并打印输出。

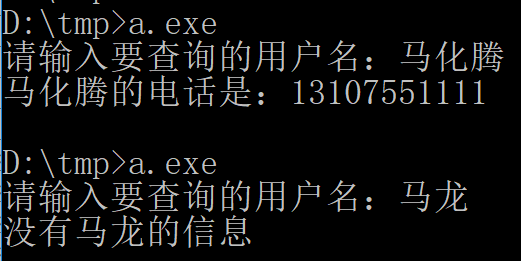
|  |
| --- |
| #include <stdio.h>  // 统计这个程序本身，有多少个字符，有多少行代码  int main(void) {  FILE \*file ;  char c;  int count\_char = 0; //字符总数  int count\_line = 0; //行数    file = fopen("test.c", "r");  if (!file ) {  printf("文件打开失败!\n");  return 1;  }    while ((c=fgetc(file)) != EOF) {  count\_char++;  if (c == '\n') {  count\_line++;  }  }    count\_line++;    printf("一共有 %d 个字符\n", count\_char);  printf("一共有 %d 行代码\n", count\_line);    return 0;  } |

1. 已有一个文件，用来保存通讯录，假设已有内容如下：

note.txt

|  |
| --- |
| 张三丰 Tel:13507318888 Addr:武当  刘备 Tel:13802289999 Addr:成都  马云 Tel:13904256666 Addr:杭州  马化腾 Tel:13107551111 Addr:深圳 |

编写一个程序，执行效果如下：



参考：

|  |
| --- |
| #include <stdio.h>  #include <string.h>  int main(void) {  FILE \*file;  char name\_search[64];  char line[256];  char name[64];  char tel[32];  char addr[32];  int found = 0;    file = fopen("note.txt", "r");  if (!file) {  printf("文件打开失败\n");  return 1;  }    printf("请输入要查询的用户名：");  scanf("%s", name\_search);    while (!feof(file)) {  fscanf(file, "%s Tel:%s Addr:%s\n", name, tel, addr);  if (!strcmp(name, name\_search)) {  printf("%s的电话是：%s\n", name\_search, tel);  found = 1;  break;  }  }    if (found == 0) {  printf("没有%s的信息\n", name\_search);  }    return 0;  } |

# 项目8 交换机后台管理系统之菜单选择

## 实现菜单选择功能

|  |
| --- |
| #include <stdio.h>  #include <string.h>  int main(void) {  // 定义变量，用来表示用户名和密码  char name[32];  char password[16];  FILE \*file; //定义了一个文件指针变量，变量名是file  char line[128];  char name\_tmp[32];  char password\_tmp[16];  char \*ret;  char n; //用户选择的菜单编号    //打开文件  file = fopen("users.txt", "r");  if (!file) { //等效于 file == NULL  printf("文件打开失败");  return 1;  }    //输入用户名和密码  while (1) {  system("cls");    // 输入用户名和密码  printf("请输入用户名：");  scanf("%s", name);  printf("请输入密码：");  scanf("%s", password);    //从文件中读取账号，并进行判断！  while (1) {  //读一行  ret = fgets(line, sizeof(line), file); //line: "admin 123456\n"  if (!ret) {  break;  }  sscanf(line, "%s %s", name\_tmp, password\_tmp);  if (!strcmp(name, name\_tmp) && !strcmp(password, password\_tmp)) {  break;  }  }    if (ret) { //用户名和密码匹配成功  break;  } else {  printf("用户名或密码错误!\n");  system("pause");  system("cls");    fseek(file, 0, SEEK\_SET); //把文件内部的位置指针设置到文件头  }  }    while (1) {  system("cls");  // 打印功能菜单  printf("---交换机后台管理---\n");  printf("1. 创建账号\n");  printf("2. IP管理\n");  printf("3. 退出\n");  printf("请选择: ");    fflush(stdin);  scanf("%c", &n);    if (n == '1') {  system("cls");  printf("\n\n---创建账号---\n\n");  printf("待实现...\n\n");  printf("\n\n按任意键返回主菜单");  fflush(stdin);  getchar();  } else if (n == '2') {  system("cls");  printf("\n\n---IP管理---\n\n");  printf("待实现...\n\n");  printf("\n\n按任意键返回主菜单");  fflush(stdin);  getchar();  } else if (n == '3') {  system("cls");  break;  } else {  system("cls");  printf("\n\n输入错误！\n\n");  printf("\n\n按任意键后，请重新输入\n\n");  fflush(stdin);  getchar();  }  }    return 0;  } |

## 项目优化

分析存在的问题：

1. if判断很多
2. 代码臃肿

分析多种优化方案。

|  |
| --- |
| #include <stdio.h>  #include <string.h>  int main(void) {  // 定义变量，用来表示用户名和密码  char name[32];  char password[16];  FILE \*file; //定义了一个文件指针变量，变量名是file  char line[128];  char name\_tmp[32];  char password\_tmp[16];  char \*ret;  char n; //用户选择的菜单编号    //打开文件  file = fopen("users.txt", "r");  if (!file) { //等效于 file == NULL  printf("文件打开失败");  return 1;  }    //输入用户名和密码  while (1) {  system("cls");    // 输入用户名和密码  printf("请输入用户名：");  scanf("%s", name);  printf("请输入密码：");  scanf("%s", password);    //从文件中读取账号，并进行判断！  while (1) {  //读一行  ret = fgets(line, sizeof(line), file); //line: "admin 123456\n"  if (!ret) {  break;  }  sscanf(line, "%s %s", name\_tmp, password\_tmp);  if (!strcmp(name, name\_tmp) && !strcmp(password, password\_tmp)) {  break;  }  }    if (ret) { //用户名和密码匹配成功  break;  } else {  printf("用户名或密码错误!\n");  system("pause");  system("cls");    fseek(file, 0, SEEK\_SET); //把文件内部的位置指针设置到文件头  }  }    while (1) {  system("cls");  // 打印功能菜单  printf("---交换机后台管理---\n");  printf("1. 创建账号\n");  printf("2. IP管理\n");  printf("3. 退出\n");  printf("请选择: ");    fflush(stdin);  scanf("%c", &n);    switch (n) {  case '1':  system("cls");  printf("\n\n---创建账号---\n\n");  printf("待实现...\n\n");  printf("\n\n按任意键返回主菜单");  fflush(stdin);  getchar();  break;  case '2':  system("cls");  printf("\n\n---IP管理---\n\n");  printf("待实现...\n\n");  printf("\n\n按任意键返回主菜单");  fflush(stdin);  getchar();  break;  case '3':  system("cls");  return 0;  default:  system("cls");  printf("\n\n输入错误！\n\n");  printf("\n\n按任意键后，请重新输入\n\n");  fflush(stdin);  getchar();  break;  }  }    return 0;  } |

## 项目精讲

### switch的基本使用

流程图：

switch (x) {

case 表达式1：

语句1

break;

case 表达式2：

语句2

break;

case 表达式3：

语句3

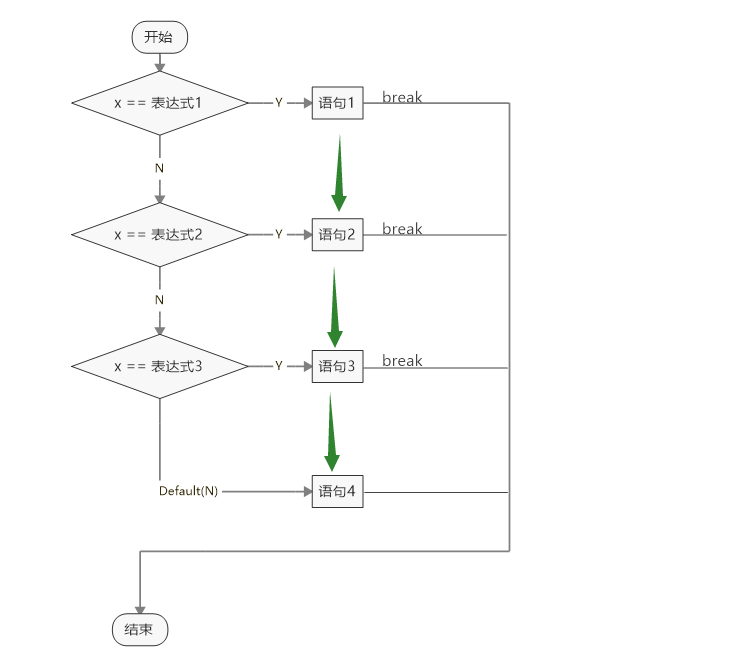
break;

default表达式1：

语句1

break;

}



demo

|  |
| --- |
| #include <stdio.h>  int main(void) {  int x;    x = 1;  switch(x) {  case 1:  printf("1\n");  case 2:  printf("2\n");  case 3:  printf("3\n");  default:  printf("default\n");  }    return 0;  } |

### switch和if的选择

switch： 用于int/char/long/long long 类型的变量，和多个特定常量的判断处理。

（float和double类型不可以）

if: 适用于各种逻辑判断

### switch的注意事项

|  |
| --- |
| #include <stdio.h>  int main(void) {  int c;  scanf("%d", &c);    switch(c) {  case 1:  int x = 0; //错误！  printf("c=1\n");  break;  case 2:  printf("c=2\n");  break;  default:  printf("other\n");  break;  }    return 0;  } |

应该修改为：

|  |
| --- |
| #include <stdio.h>  int main(void) {  int c;  scanf("%d", &c);    switch(c) {  case 1:  {  int x = 0; //合法  ｝  printf("c=1\n");  break;  case 2:  printf("c=2\n");  break;  default:  printf("other\n");  break;  }    return 0;  } |

## 项目练习

### 练习1

独立完成项目8.

### 练习2

编写一个程序，让用户输入一个月份，然后判断这个月有多少天。

假设2月份始终有28天。

分别用if 和switch语句实现。

### 练习3

让用户输入一个成绩，然后判断该成绩的等级。

0-59: 不及格

60-79: 及格

80-89: 良好

90-100: 优秀

其它：非法成绩

分别用if 和switch语句实现。

# 项目9 交换机后台管理系统之函数优化

## 项目需求

项目8的实现，main函数太臃肿，不便于阅读和维护。

## 项目实现

用函数来优化。

|  |
| --- |
| #include <stdio.h>  #include <string.h>  #include <stdlib.h>  FILE \*file;  void init(void) {  //打开文件  file = fopen("users.txt", "r");  if (!file) { //等效于 file == NULL  printf("文件打开失败");  //return 1;  exit(1);  }  }  void login(void) {  char name[32];  char password[16];  char line[128];  char name\_tmp[32];  char password\_tmp[16];  char \*ret;    //输入用户名和密码  while (1) {  system("cls");    // 输入用户名和密码  printf("请输入用户名：");  scanf("%s", name);  printf("请输入密码：");  scanf("%s", password);    //从文件中读取账号，并进行判断！  while (1) {  //读一行  ret = fgets(line, sizeof(line), file); //line: "admin 123456\n"  if (!ret) {  break;  }  sscanf(line, "%s %s", name\_tmp, password\_tmp);  if (!strcmp(name, name\_tmp) && !strcmp(password, password\_tmp)) {  break;  }  }    if (ret) { //用户名和密码匹配成功  break;  } else {  printf("用户名或密码错误!\n");  system("pause");  system("cls");    fseek(file, 0, SEEK\_SET); //把文件内部的位置指针设置到文件头  }  }  }  void create\_user(void) {  system("cls");  printf("\n\n---创建账号---\n\n");  printf("待实现...\n\n");  printf("\n\n按任意键返回主菜单");  fflush(stdin);  getchar();  }  void ip\_admin(void) {  system("cls");  printf("\n\n---IP管理---\n\n");  printf("待实现...\n\n");  printf("\n\n按任意键返回主菜单");  fflush(stdin);  getchar();  }  void logout(void) {  system("cls");  fclose(file);  exit(0);  }  void input\_error(void) {  system("cls");  printf("\n\n输入错误！\n\n");  printf("\n\n按任意键后，请重新输入\n\n");  fflush(stdin);  getchar();  }  void show\_memu(void) {  system("cls");  // 打印功能菜单  printf("---交换机后台管理---\n");  printf("1. 创建账号\n");  printf("2. IP管理\n");  printf("3. 退出\n");  printf("请选择: ");  }  int main(void) {  char n; //用户选择的菜单编号    init(); //初始化  login(); //登录    while (1) {  show\_memu();  fflush(stdin);  scanf("%c", &n);  switch (n) {  case '1':  create\_user();  break;  case '2':  ip\_admin();  break;  case '3':  logout();  break;  default:  input\_error();  break;  }  }    return 0;  } |

## 项目精讲

### 为什么要使用函数

已经有main函数，为什么还要自定义函数？

1. “避免重复制造轮子”，提高开发效率

2）便于维护

### 函数的声明、定义和使用

函数的设计方法：

1. 先确定函数的功能
2. 确定函数的参数

是否需要参数，参数的个数，参数的类型

1. 确定函数的返回值

是否需要返回值，返回值的类型

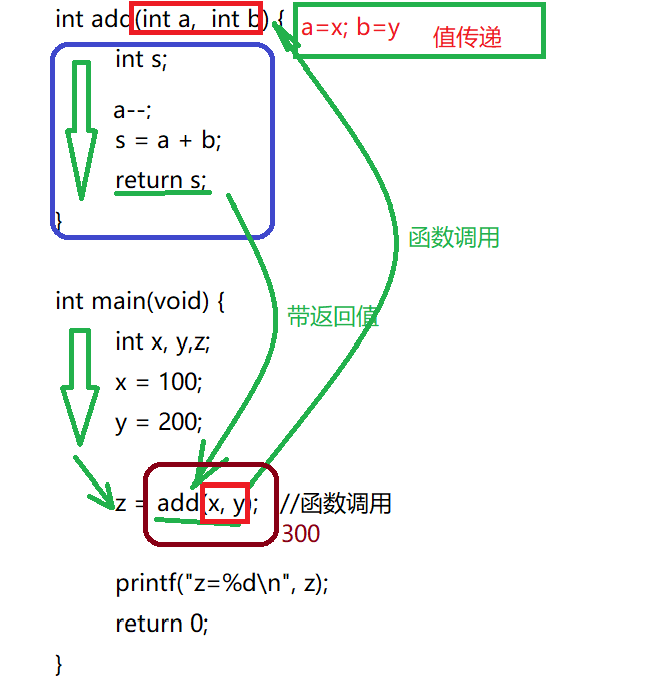
函数的声明

函数的使用

### 函数的值传递

调用函数时，形参被赋值为对应的实参，

实参本身不会受到函数的影响！



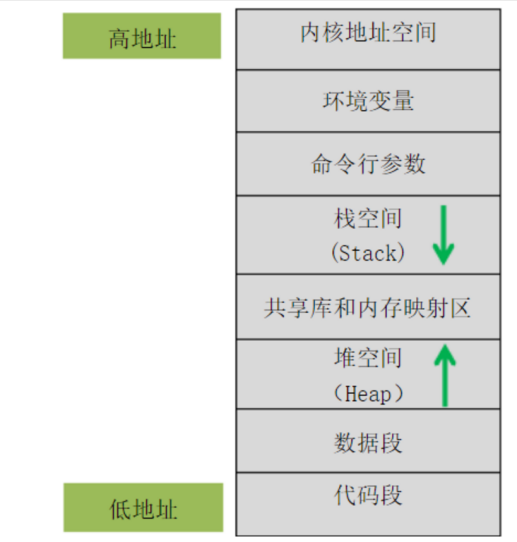
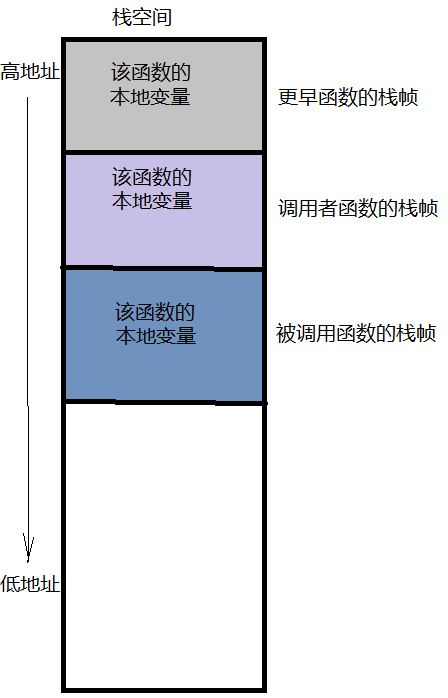
### 函数的栈空间

要避免栈空间溢出。

当调用一个函数时，就会在栈空间，为这个函数，分配一块内存区域，

这块内存区域，专门给这个函数使用。

这块内存区域，就叫做“栈帧”。

demo1

|  |
| --- |
| #include <stdio.h>  #include <string.h>  void test(void) {  //运行时将因为栈帧空间溢出，而崩溃  char buff[1024\*1024\*2];  memset(buff, 0, sizeof(buff));  }  int main(void) {  test();  return 0;  } |

demo2

|  |
| --- |
| #include <stdio.h>  #include <string.h>  void test(int n) {  char buff[1024\*256];  memset(buff, 0, sizeof(buff));    if (n==0) {  return;  }    printf("n=%d\n", n);  test(n-1);  }  int main(void) {  //test(5);  //因为每个栈帧有256K以上, 10个栈帧超出范围  test(10);  return 0;  } |

### 递归函数

定义：在函数的内部，直接或者间接的调用自己。

要点：

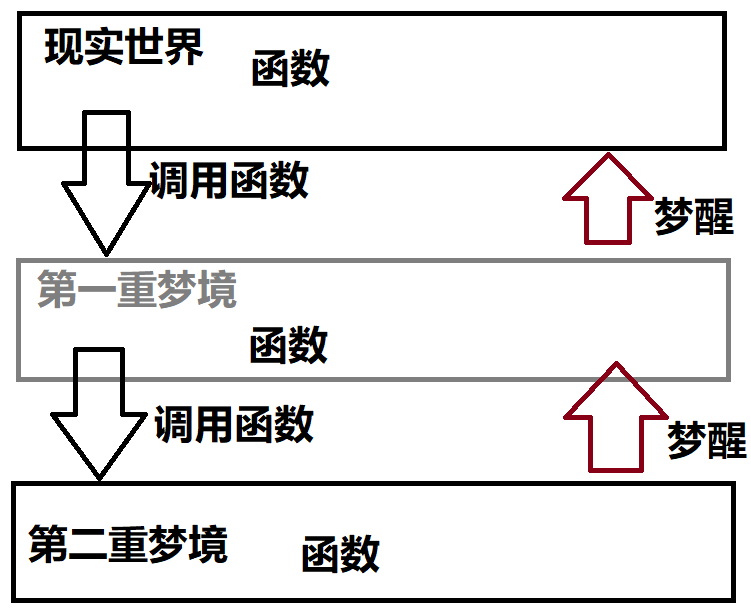
再定义递归函数时，一定要确定一个“结束条件”！！！

使用场合：

处理一些特别复杂的问题，难以直接解决。

但是，可以有办法把这个问题变得更简单（转换成一个更简单的问题）。

盗梦空间



例如：

1. 迷宫问题
2. 汉诺塔问题

斐波那契数列

1. 1， 2， 3， 5， 8， 13， 21， ....

计算第n个数是多少？

f(n)

当n >2时，f(n) = f(n-1) + f(n-2)

当n=1或n=2时， f(n)就是1

int fib(int n) {

int s;

if (n == 1|| n == 2) {

return 1;

}

s = fib(n-1) + fib(n-2);

return s;

}

递归函数的缺点：

性能很低！！！

## 项目练习

### 练习1

独立完成项目9

### 练习2

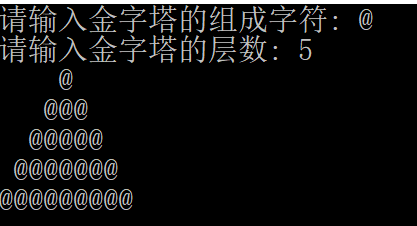
定义一个函数，实现1+2+3+...+n

|  |
| --- |
| #include <stdio.h>  int sum(int n) {  int i;  int s = 0;    for (i=1; i<=n; i++) {  s += i;  }    return s;  }  int main(void) {  int value;    printf("请输入一个整数: ");  scanf("%d", &value);  if (value < 0) {  printf("需要大于0\n");  return 1;  }    printf("%d\n", sum(value));    return 0;  } |

### 打印金字塔

打印指定类型的金字塔，用自定义函数实现。

效果如下：



代码：

|  |
| --- |
| #include <stdio.h>  void show(char c, int n) {  int i;  int j;    for (i=1; i<=n; i++) {  for (j=0; j<n-i; j++) {  printf(" ");  }  for (j=0; j<2\*i-1; j++) {  printf("%c", c);  }  printf("\n");  }  }  int main(void) {  char c;  int n;    printf("请输入金字塔的组成字符: ");  scanf("%c", &c);  if (c == '\n' || c == ' ' || c == '\t') {  printf("请输入一个非空白字符\n");  return 1;  }    printf("请输入金字塔的层数: ");  scanf("%d", &n);  if (n < 1) {  printf("层数需要大于0\n");  return 1;  }    show(c, n);    return 0;  } |

### 用递归函数实现练习2

|  |
| --- |
| #include <stdio.h>  int sum(int n) {  int s;    if (n == 1) {  return 1;  }    s = n + sum(n-1);    return s;  }  int main(void) {  int value;    printf("请输入一个整数: ");  scanf("%d", &value);    if (value < 0) {  printf("需要大于0\n");  return 1;  }    printf("%d\n", sum(value));    return 0;  } |

### 用递归函数实现汉诺塔

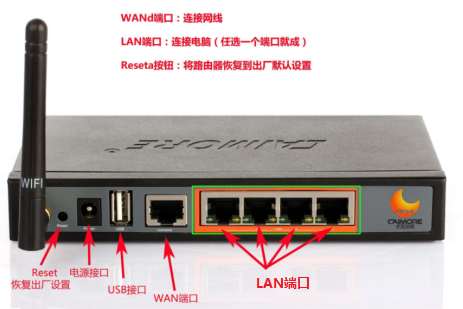
|  |
| --- |
| #include <stdio.h>  void hanoi(int n, char pillar\_start[], char pillar\_mid[], char pillar\_end[]) {  if (n == 1) {  printf("从%s移动到%s\n", pillar\_start, pillar\_end);  return;  }    hanoi(n-1, pillar\_start, pillar\_end, pillar\_mid);  printf("从%s移动到%s\n", pillar\_start, pillar\_end);  hanoi(n-1, pillar\_mid, pillar\_start, pillar\_end);  }  int main(void) {  char name1[] = "A柱";  char name2[] = "B柱";  char name3[] = "C柱";  int n = 3; //盘子数    hanoi(3, name1, name2, name3);    return 0;  } |

# 项目10

## 项目需求

交换机





端口：

1. 端口名称
2. 端口状态
3. 端口的IP地址
4. 端口类型

WAN

LAN

## 项目实现

1. 添加菜单框架

|  |
| --- |
| #include <stdio.h>  #include <string.h>  #include <stdlib.h>  FILE \*file;  void init(void) {  //打开文件  file = fopen("users.txt", "r");  if (!file) { //等效于 file == NULL  printf("文件打开失败");  //return 1;  exit(1);  }  }  void login(void) {  char name[32];  char password[16];  char line[128];  char name\_tmp[32];  char password\_tmp[16];  char \*ret;    //输入用户名和密码  while (1) {  system("cls");    // 输入用户名和密码  printf("请输入用户名：");  scanf("%s", name);  printf("请输入密码：");  scanf("%s", password);    //从文件中读取账号，并进行判断！  while (1) {  //读一行  ret = fgets(line, sizeof(line), file); //line: "admin 123456\n"  if (!ret) {  break;  }  sscanf(line, "%s %s", name\_tmp, password\_tmp);  if (!strcmp(name, name\_tmp) && !strcmp(password, password\_tmp)) {  break;  }  }    if (ret) { //用户名和密码匹配成功  break;  } else {  printf("用户名或密码错误!\n");  system("pause");  system("cls");    fseek(file, 0, SEEK\_SET); //把文件内部的位置指针设置到文件头  }  }  }  void create\_user(void) {  system("cls");  printf("\n\n---创建账号---\n\n");  printf("待实现...\n\n");  printf("\n\n按任意键返回主菜单");  fflush(stdin);  getchar();  }  void ip\_admin(void) {  system("cls");  printf("\n\n---IP管理---\n\n");  printf("待实现...\n\n");  printf("\n\n按任意键返回主菜单");  fflush(stdin);  getchar();  }  void logout(void) {  system("cls");  fclose(file);  exit(0);  }  void input\_error(void) {  system("cls");  printf("\n\n输入错误！\n\n");  printf("\n\n按任意键后，请重新输入\n\n");  fflush(stdin);  getchar();  }  void show\_memu(void) {  system("cls");  // 打印功能菜单  printf("---交换机后台管理---\n");  printf("1. 创建账号\n");  printf("2. IP管理\n");  printf("3. 退出\n");  printf("4. 端口管理\n");  printf("请选择: ");  }  void show\_ports(void) {  system("cls");  printf("---端口状态---\n");  printf("待实现.\n");  system("pause");  }  void set\_ports(void) {  system("cls");  printf("---端口设置---\n");  printf("待实现.\n");  system("pause");  }  void port\_admin(void) {  char n;    while(1) {  system("cls");  printf("1. 查看端口\n");  printf("2. 设置端口\n");  printf("3. 返回主菜单\n");  printf("请选择: ");    fflush(stdin);  scanf("%c", &n);  if (n == '1') {  show\_ports();  } else if (n == '2') {  set\_ports();  } else if (n == '3') {  break;  } else {  input\_error();  }  }  }  int main(void) {  char n; //用户选择的菜单编号    init(); //初始化  login(); //登录    while (1) {  show\_memu();  fflush(stdin);  scanf("%c", &n);  switch (n) {  case '1':  create\_user();  break;  case '2':  ip\_admin();  break;  case '3':  logout();  break;  case '4':  port\_admin();  break;  default:  input\_error();  break;  }  }    return 0;  } |

1. 端口信息的表示

添加类型定义和端口变量

|  |
| --- |
| struct port {  char name[16]; //端口的名称  int status; //1: 激活 0：禁用  char ip[16]; //192.168.1.5  char type[4]; //端口类型 LAN WAN  };  //定义了5个端口变量  struct port port1;  struct port port2;  struct port port3;  struct port port4;  struct port port5; |

1. 实现功能

|  |
| --- |
| void show\_port(struct port port) {  printf("名称[%s]\t状态[%s]\tIP[%s]\t类型[%s]\n",  port.name,  port.status == 0 ? "禁用":"激活",  port.ip,  port.type);  }  void show\_ports(void) {  system("cls");  printf("---端口状态---\n");    printf("PORT1:\t");  show\_port(port1);    printf("PORT2:\t");  show\_port(port2);    printf("PORT3:\t");  show\_port(port3);    printf("PORT4:\t");  show\_port(port4);    system("pause");  }  void set\_port1(void) {  system("cls");  printf("---设置PORT1端口---\n");    printf("请输入端口名称: ");  scanf("%s", port1.name);    printf("请输入端口的状态：[0:禁止] [1:激活] ");  scanf("%d", &port1.status);    printf("请输入端口的类型：[LAN 或 WAN] ");  scanf("%s", port1.type);    printf("请输入端口的IP地址: ");  scanf("%s", port1.ip);    system("pause");  }  void set\_port2(void) {  system("cls");  printf("---设置PORT2端口---\n");    printf("请输入端口名称: ");  scanf("%s", port2.name);    printf("请输入端口的状态：[0:禁止] [1:激活] ");  scanf("%d", &port2.status);    printf("请输入端口的类型：[LAN 或 WAN] ");  scanf("%s", port2.type);    printf("请输入端口的IP地址: ");  scanf("%s", port2.ip);  }  void set\_port3(void) {  system("cls");  printf("---设置PORT3端口---\n");    printf("请输入端口名称: ");  scanf("%s", port3.name);    printf("请输入端口的状态：[0:禁止] [1:激活] ");  scanf("%d", &port3.status);    printf("请输入端口的类型：[LAN 或 WAN] ");  scanf("%s", port3.type);    printf("请输入端口的IP地址: ");  scanf("%s", port3.ip);  }  void set\_port4(void) {  system("cls");  printf("---设置PORT4端口---\n");    printf("请输入端口名称: ");  scanf("%s", port4.name);    printf("请输入端口的状态：[0:禁止] [1:激活] ");  scanf("%d", &port4.status);    printf("请输入端口的类型：[LAN 或 WAN] ");  scanf("%s", port4.type);    printf("请输入端口的IP地址: ");  scanf("%s", port4.ip);  }  void set\_port5(void) {  system("cls");  printf("---设置PORT5端口---\n");    printf("请输入端口名称: ");  scanf("%s", port5.name);    printf("请输入端口的状态：[0:禁止] [1:激活] ");  scanf("%d", &port5.status);    printf("请输入端口的类型：[LAN 或 WAN] ");  scanf("%s", port5.type);    printf("请输入端口的IP地址: ");  scanf("%s", port5.ip);  }  void set\_ports(void) {  char n;    while(1) {  system("cls");  printf("---端口设置---\n");  printf("1. PORT1\n");  printf("2. PORT2\n");  printf("3. PORT3\n");  printf("4. PORT4\n");  printf("5. PORT5\n");  printf("6. 返回\n");  printf("请选择: ");    fflush(stdin);  scanf("%c", &n);  switch (n) {  case '1':  set\_port1();  break;  case '2':  set\_port2();  break;  case '3':  set\_port3();  break;  case '4':  set\_port4();  break;  case '5':  set\_port5();  break;  case '6':  return;  default:  input\_error();  break;  }  }    } |

## 项目精讲

## 为什么要使用“结构”（结构体）

但需要表示一些复制信息时，使用单纯的数据类型很不方便。

比如：学生信息（学号，姓名，班级，电话，年龄）

## 什么是“结构”

结构，就是程序员自定义的一种“数据类型”

是使用多个基本数据类型、或者其他结构，组合而成的一种新的“数据类型”。

## 结构的定义

struct 结构名 {

成员类型 成员名；

成员类型 成员名；

};

实例：

|  |
| --- |
| struct student {  char name[16];  int age;  char tel[12];  }; |

特别注意：

1. 要以struct开头
2. 最后要使用分号
3. 各成员之间用分号隔开

## 结构的初始化

demo

|  |
| --- |
| #include <stdio.h>  struct student {  char name[16];  int age;  };  int main(void) {  struct student s1 = {  "Rock", 38  };    struct student s2 = {  .age = 100,  .name = "张三丰"  };    struct student s3;  s3.age = 40;  strcpy(s3.name, "杨过");    printf("%s, %d\n", s1.name, s1.age);  printf("%s, %d\n", s2.name, s2.age);  printf("%s, %d\n", s3.name, s3.age);  return 0;  } |

## 结构的使用

|  |
| --- |
| // 定义结构体变量  // 注意：完整的类型名称是 struct student  // 而不只是student  struct student s1，s2;  struct student s3;  scanf(“%s”, s1.name);  s1.name = 25;  s2 = s1; //结构体变量之间可以直接赋值 |

使用形式：

结构体变量**.**成员变量

中间用 . 分隔

## 使用结构体作为函数参数

|  |
| --- |
| #include <stdio.h>  #include <string.h>  struct student {  char name[16];  int age;  char tel[12];  };  void work(struct student stu) {  stu.age++;  printf("%s,%d\n", stu.name, stu.age);  }  int main(void) {  struct student s;    strcpy(s.name, "Rock");  s.age = 38;    work(s); //结构体变量s作为函数参数，并不会改变s本身的值  printf("%s,%d\n", s.name, s.age);    return 0;  } |

注意：

一般不建议把结构体直接作为函数参数。

因为结构体的size比较大，直接传递，消耗性能！

解决方案（使用指针）

## 全局变量、局部变量

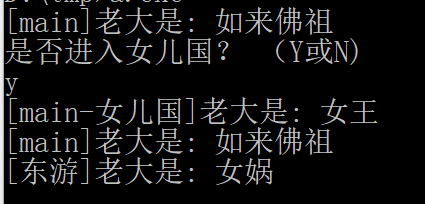
test1.c

|  |
| --- |
| #include <stdio.h>  void east\_travel(void);  char master[16] = "女娲";  void west\_travel(void) {  char master[16] = "唐僧";  printf("[西游]老大是: %s\n", master);  }  int main(void) {  char master[16] = "如来佛祖";    printf("[main]老大是: %s\n", master);    char c;  printf("是否进入女儿国？ （Y或N)\n");  fflush(stdin);  scanf("%c", &c);  if (c=='Y' || c=='y') {  char master[16] = "女王";  printf("[main-女儿国]老大是: %s\n", master);  }    printf("[main]老大是: %s\n", master);    east\_travel();  return 0;  } |

test2.c

|  |
| --- |
| #include <stdio.h>  extern char master[16]; //不能初始化！表示这个全局变量是在其他文件中定义的！  void east\_travel(void) {  printf("[东游]老大是: %s\n", master);  } |

gcc test1.c test2.c



## 项目练习

### 练习1

独立实现项目10

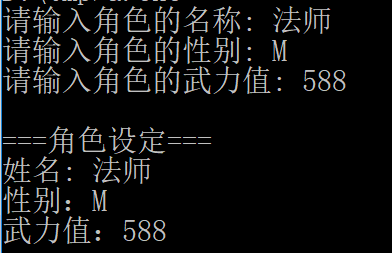
### 练习2

1. 定义一个结构，用来描述一个游戏角色的基本信息。

该角色信息有，名称，性别，武力值

并让用户输入1个角色。

运行效果如下：



参考代码：

|  |
| --- |
| #include <stdio.h>  struct role {  char name[32];  char sex; //'M':男 'W':女  int power;  };  int main(void) {  struct role r1, r2;    printf("请输入角色的名称: ");  scanf("%s", r1.name);    fflush(stdin);  printf("请输入角色的性别: ");  scanf("%c", &r1.sex);    printf("请输入角色的武力值: ");  scanf("%d", &r1.power);    printf("\n===角色设定===\n");  printf("姓名: %s\n", r1.name);  printf("性别：%c\n", r1.sex);  printf("武力值：%d\n", r1.power);    return 0;  } |