第一周的笔记

第一天

mac 系统: unix 系统图形界面的显示

开发环境:图形界面IDE(xcode) 和终端(terminal)

http://192.168.88.253/1408

快捷键:

command + C : 表示复制

command + V : 粘贴

首先 command +c , command + option(alt) +v: 剪切文件

command + F : 搜索

command + N: 打开新的应用程序

command + Space: 切换输入法

command + q: 退出当前应用程序

command + tab: 切换应用程序

Mac 不像windows 样有磁盘分区的概念, 它只有一个根目录

用户目录: 保存用户所有操作的目录

终端操作命令:

ls : 显示当前文件夹下包含的所有文件/文件夹

ls -a : 显示所有文件/文件夹包含隐藏的文件/文件夹

cd : 进入文件

cd + 文件路径(绝对路径/相对路径)

绝对路径: 从根文件开始算起

相对路径: 从当前目录开始算起的路径 用”.”引用

cd 返回用户操作目录

cd .. 返回上一级目录

cd / 返回根目录

mkdir : 创建文件目录

mkdir + dicname:

touch : 创建文件

touch + filename

cat + filename: 显示文件内容

cp: 文件拷贝

cp + src + dst

mv : 给文件或者文件夹修改名字或者拷贝

mv + src +dst

rm : 删除文件或目录

rm + rf + filename/dicname

sudo + 命令 : 加权限操作

vim操作 vim+vimrc +filename

vim: 命令模式 编辑模式 可视模式

i: 从光标前面开始输入文本

o: 从光标的下一行输入文本

O:从光标的上一行输入文本

ESC: 进入命令模式

set number : 显示行标

set nonumber: 不显示行标

编译文件

gcc -o hello hello.c

执行文件

./hello

yy: 复制 xyy复制多行(如: 4yy)

dd:剪切 xdd 剪切多行(如: 4dd)

p: 表示粘贴

u: 撤销

x: 删除一个字符

:wq 保存并退出

:q! 不保存退出

:w 保存

C语言中的注释方法

//

/\* \*/

pwd 文件路径

Xcode

control +F : 往前移动一个字符

control +B: 往后移动一个字符

control +N: 往下移动一行

control + p: 网上移动一行

command + z: 撤销

第二天

(上午) #include <stdio.h>

//----------------------------------进制转换-------------------------

//十进制

//0 1 2 3 4 5 6 7 8 9 10 11 12 .... 99 100 101 ... 999 1000

// 10^(n-1)

//高电平 表示1 低电平表示 0

//二进制: 0 1

//机器语言

//5+6

//101001010100101001

//101001010101010010

//101010010101001010

//汇编语言

//5+6

//movl $5 -8(rbp)

//movl $6 -12(rbp)

//movl -8(rbp) %eax

//addl -12(rbp) %eax

//movl %eax %esi

//汇编器

// 汇编语言 --> 二进制

//C语言

// 5+6

// int a = 5;

// int b = 6;

// int c = a+b;

//C语言的特点: 简洁 可读性好 可移植性强,无平台限制

//C语言 --> 汇编语言 ---> 二进制

//12345 求除10余数,到0结束,倒序取余数

//12345 / 10 -- 5 5\*10^0

//1234 / 10 -- 4 4\*10^1

//123 / 10 -- 3 3\*10^2

//12 / 10 -- 2 2\*10^3

//1 / 10 -- 1 1\*10^4

//0 / 10 -- 0

//

//12345 = 12345 = 1\*10^4 + 2\*10^3 + 3\*10^2 + 4\*10^1 + 5\*10^0= 12345

//0b:二进制的前缀

//123 除2倒序取余数

//123 / 2 --- 1

//61 / 2 --- 1

//30 / 2 --- 0

//15 / 2 --- 1

//7 / 2 --- 1

//3 / 2 --- 1

//1 / 2 --- 1

//0

//123 = 0b1111011 = 1\*2^0 + 1\*2^1 + 0 + 1\*2^3 + 1\*2^4 + 1\*2^5 +1\*2^6 = 1 + 2 + 0 + 8 + 16 + 32 + 64 = 123

//96 234 78 49 278

//运行程序: command + R

/\*int main(int argc, const char \* argv[])

{

int bin[32];

int i =0 ;

int a;

scanf("%d", &a);

while (a) {

bin[i] = a%2;

a/=2;

i++;

}

for (i= i-1; i>=0; i--) {

printf("%d", bin[i]);

}

printf("\n");

return 0;

}\*/

// 0 1 10 11 100 101 110 111 1000 1001 1010 1011 1100 1101 1110 1111

// 0 ~ 15

//逢二进一

//八进制 逢八进一

//前缀加0表示八进制数

//除八倒序取余数

// 0 1 2 3 4 5 6 7

// 10 11 12 13 ...

// 96

// 96 / 8 --- 0

// 12 / 8 --- 4

// 1 / 8 --- 1

// 0

//96 = 0140

//123 456 58 47

//contrl + i: 排序程序

/\*int main(int argc, const char \* argv[])

{

int arr[32];

int i =0 ;

int a;

scanf("%d", &a);

while (a) {

arr[i] = a%8;

a/=8;

i++;

}

for (i= i-1; i>=0; i--) {

printf("%d", arr[i]);

}

printf("\n");

return 0;

}\*/

// 0173

// 0 1 2 3 4 5 6 7

//000 001 010 011 100 101 110 111

// 0173 = 1111011

//"421" 规则

//

//011 010 101 010 100 100 111 101 010 101 010

//032524475252

//二进制转8进制 :

//从低位开始, 每三位为一组, 高位不足三位补0, 然后用一位八进制数代替三位二进制数

//111 110 100 101 010 101 010 011 101 100 101

//076452523545

//逢十六进一

//十进制数转十六进制

//除十六倒序取余数

//0x: 十六进制数的前缀

//十六进制:0 1 2 3 4 5 6 7 8 9 a/A b/B c/C d/D e/E f/F

//e f 10 11 ..... 1f 20 ......a0 .....af b0 .....

//123

//123 / 16 --- b

//7 / 16 ---- 7

//0

// 123 = 0x7b

//345 96 128 77

/\*int main(int argc, const char \* argv[])

{

int arr[32];

int i =0 ;

int a;

scanf("%d", &a);

while (a) {

arr[i] = a%16;

a/=16;

i++;

}

for (i= i-1; i>=0; i--) {

printf("%x", arr[i]);

}

printf("\n");

return 0;

}\*/

//0x14

//0 1 2 3 4 5 6 7

//0000 0001 0010 0011 0100 0101 0110 0111

// 8 9 a b c d e f

// 1000 1001 1010 1011 1100 1101 1110 1111

//"8421"

// 0x14

// 0x14 = 10100 = 2^4 + 2^2 = 20

//十六进制转二进制

//把十六进制中的每一位,分别用四位二进制表示

//0x45 0x96 0xab 0x78

//0x45 = 1000101

//0x96 = 10010110

//0xab = 10101011

//0x78 = 1111000

//0100 1011 0110 1101 0101 0010 1010 0101

//0x4b6d52a5

//二进制转十六进制:

//从低位开始,每四位为一组,高位不足四位补0, 用十六进制的数字代替四位二进制数

//0001 0111 1010 1100 1101 0101 0100 1010 1010 1111 1111

//0x17acd54aaff

(下午) #include <stdio.h>

//

//计算机单位

// 最小单位是位(bit)

// 8bits = 1Byte

// 1Kb = 1024Bytes

// 1Mb = 1024Kb

// 1Gb = 1024Mb

// 1Tb = 1024Gb

// 1Pb = 1024Tb

//厂家

//500G = 500\*1000Mb=500\*1000\*1000Kb

//

// 内存: 存储程序跟数据的地方

// 8bits

// 1111 1111

// 跟计算机约定: 最高位为符号位 最高位为 1 表示负数

// 最高位为 0 表示正数

// 6 : 0000 0110

// -6: 1000 0110

//原码: 最高位为符号位, 其余位表示绝对值大小

// 5 原码: 0000 0101

//-5 原码: 1000 0101

// 6+5`

// 0000 0110

//+ 0000 0101

//-----------------

// 0000 1011 = 11`

// 6-5

// 6+(-5)

// 0000 0110

//+ 1000 0101

//----------------

// 1000 1011 = -11

//

//原码不能参加运算: 因为符号位也参加了运算

//反码: 正数的反码就是它的原码,对于负数,符号位变,其余位按位取反

// -5 原码: 1000 0101 反码: 1111 1010

// -8 原码: 1000 1000 反码: 1111 0111

// -9 原码: 1000 1001 反码: 1111 0110

// -13原码: 1000 1101 反码: 1111 0010

// -17原码: 1001 0001 反码: 1110 1110

// 6+(-5)

// 0000 0110

// + 1111 1010

//-----------------

// 0000 0000(反码) = 0000 0000 = 0

//

//反码转原码: 正数的原码就是它的反码,负数的原码是符号位不变,其余位按位取反

//

// -6 -5

// (-6)+(-5)

// 1111 1001

//+ 1111 1010

//----------------

// 1111 0011(反码) = 10001100 = -12

//反码作减法运算 , 得到的值总是比真实值少1

//

//补码: 对于正数来说,补码就是它的反码,也是它的原码,

//对于负数来说,补码是在反码的基础加1

// -6原码: 1000 0110 反码: 1111 1001 补码: 1111 1010

// -7原码: 1000 0111 反码: 1111 1000 补码: 1111 1001

// -6 +(-7)

// 1111 1010

// 1111 1001

//+--------------

// 1111 0011(补码)

//补码转原码: 对于正数, 补码的原码就是它本身, 对于负数,补码的原码,就是补码的反码 加1

// 1111 0011(补码) => 1000 1100(反码) => 1000 1101(原码)=-13

//

//正数的原码 反码 补码 都是一样的, 都是它的原码

//计算机采用补码的方式存储数据

// 8bits

// 0000 0000 -- 0111 1111 0 ~ 127

// 1000 0000(补码) -- 1111 1111(补码) -128 ~ 0

// +0 0000 0000

// -0 1000 0000 1000 0000

// -0 表示8bits范围内最小的数据

//对于-0来说 最高位的符号位即表示符号位,又表示绝对值大小

//-127 补码 : 1000 0001

// -23 原码: 10010111 反码: 1110 1000 补码: 1110 1001

// -11 原码: 1000 1011 反码: 1111 0100 补码: 1111 0101

// -45 原码: 1010 1101 反码: 1101 0010 补码: 1101 0011

// -34 原码: 1010 0010 反码: 1101 1101 补码; 1101 1110

//

// 数据类型

// char short int long long long

//

//sizeof() 求数据类型在内存空间中占用的大小 算出来的类型是long类型

//long long 类型的大小根据平台而定

// 64bits平台

// char short int long long long

// 1字节 2字节 4字节 8字节 8字节

//

// 32bits平台

// char short int long long long

// 1字节 2字节 4字节 4字节 8字节

//char -128 ~ 127

//int -2^31(-2147483648) ~ 2^31-1(2147483647)

//short -2^15 ~ 2^15-1

//long -2^63 ~ 2^63-1

//long long -2^63 ~ 2^63-1;

/\*int main(int argc, const char \* argv[])

{

printf("char == %ld\n", sizeof(char));//%ld 为一个占位符

printf("int == %ld\n", sizeof(int));//\n表示换行

printf("short == %ld\n", sizeof(short));

printf("long == %ld\n",sizeof(long));

printf("long long == %ld\n",sizeof(long long));

return 0;

}\*/

/\*int main(int argc, const char \*argv[])

{

printf("int == %d ~~ %d\n", 0x80000000,0x7fffffff);

printf("short == %d ~~ %d\n", (short)0x8000, (short)0x7fff);

printf("long == %ld ~~ %ld\n", 0x8000000000000000,0x7fffffffffffffff);

// short a = -12;

// printf("%d\n", a);

return 0;

}\*/

//浮点型数据(实型数据)

//float 单精度浮点型

//double 双精度浮点型

//long double 长双精度浮点型

//3.14

//0.314 \* 10^1(科学计数法)

//0.314e1(指数法)

//1. 符号位 2, 指数位 3, 小数位

//float 精度 6~7

//double 精度 15 ~16

//long double

/\*int main(int argc, const char \*argv[])

{

printf("float == %ld\n",sizeof(float) );

printf("double == %ld\n", sizeof(double));

printf("long double == %ld\n", sizeof(long double));

return 0;

}\*/

//int main(int argc, const char \*argv[])

//{

// printf("%f\n",3.1444466665435353535);

// printf("%.20lf\n",3.145667888998900000908766);

// return 0;

//}

//常量: 常量是计算机中最基本的元素 整型常量 浮点型常量 字符常量 字符串常量 枚举常量

//如: 12 3.14 'a' "qianfeng"

//

//变量: int a = 12 ; a = 13;

//

/\*int main(int argc, const char \*argv[])

{

int a=0;// "=" 赋值符号 "==" 表示等于

printf("a = %d\n", a);

a= 12;//赋值语句

printf("a= %d\n", a);

return 0;

}\*/

int main(int argc, const char \* argv[])

{

int arr[32];

int i =0 ;

int a;

scanf("%d", &a);

while (a) {

arr[i] = a%8;

a/=8;

i++;

}

for (i= i-1; i>=0; i--) {

printf("%o", arr[i]);

}

printf("\n");

return 0;

}

第三题

(上午) #include <stdio.h>

//常量: 常量是计算机中的罪基本元素, 字符常量,字符串常量,整型常量,浮点型常量,枚举常量

//字符常量: 'a' '}' '\n' '\0' '\r'

//ASCII码(美国信息标准交换码)

//'A' 65

//'a' 97

//'0' 48

//字符常量特点: 单引号括起来

//转义字符: '\n' '\0' '\r' '\\' '\'' '\"'

//字符串常量: 如 "qianfeng" "hello world"

//字符串常量特点:

// 1. 用双引号包含起来

// 2. 每一个元素都是一个字符

// 3. 字符串中的每个元素占用1个字节

// 4. 字符串以'\0'表示字符串的结束

//

//'a' "a"

//1.一个是单引号,一个双引号

//2.'a'表示字符,"a"表示字符串

//3.'a' 占用4字节内存空间 "a"占用2字节内存空间

//4. "a"有\0表示结束

//整型: 12 678

//浮点型常量: 3.14 45.5678

//变量: 内存中一段命名的存储空间

//int a

//变量定义: 数据类型 + 变量名

//变量名必须为一个标识符

//标识符: 以字母或下划线开头后面跟若干个字母数字下划线

//1. 必须以字母或者下划线开头

//2. 标识符不能跟C语言中的关键字相同

//3. \_开头的标识符通常给编译器使用

//command + / 注释/取消注释

//int main(int argc, const char \* argv[])

//{

// printf("%ld\n" , sizeof('a'));

// return 0;

//}

/\*int main(int argc, const char \*argv[])

{

//char ch = 'a';//隐式转换

char ch1,ch2;

printf("请输入两个字符\n");

scanf("%c",&ch1);

getchar();//读取一个字符

scanf("%c",&ch2);

printf("%c %c\n",ch1,ch2);

// printf("%c\n",ch);//%c表示char类型数据

//printf("ch = %ld a = %ld\n", sizeof(ch), sizeof('a'));

return 0;

}\*/

/\*int main(int argc, const char \*argv[])

{

char str[100];

char str1[100];

scanf("%s",str);

scanf("%s",str1);

printf("%s\n",str);

printf("%s\n", str1);

return 0;

}\*/

/\*int main(int argc, const char \*argv[])

{

// char str[100];

char str1[100];

scanf("%[abcdefg]",str1);

printf("%s\n", str1);

// scanf("%[^\n]",str);

// printf("%s\n",str);

return 0;

}\*/

/\*int main(int argc, const char \*argv[])

{

char str[100];

gets(str);

printf("%s\n", str);

return 0;

}\*/

//int main(int argc , const char \*argv[])

//{

// //%d %ld %lld

// int a = 12;

// long b = 345345;

// long long c = 234567892345678;

// printf("a = %d b = %ld c= %lld\n", a, b, c);

//

// return 0;

//}

//%u %lu %llu

//unsigned

//unsigned char

//unsigned int

//unsigned short

//unsigned long

//unsigned long long

//无符号整型数据

/\*int main(int argc, const char \*argv[])

{

unsigned char a=0xff;

unsigned short b=0xffff;

unsigned int c=0xffffffff;

unsigned long d=0xffffffffffffffff;

printf("char max == %u\n", a);

printf("short max == %u\n",b);

printf("int max == %u\n",c);

printf("long max == %lu\n", d);

return 0;

}\*/

int main(int argc, const char \*argv[])

{

//%f %lf %Lf

float f;

double f1 = 34567.4545;

long double f2 = 234567.23456789;

scanf("%f", &f);

printf("%f\n", f);

printf("%lf\n", f1);

printf("%.10Lf\n", f2);

return 0;

}

(下午) #include <stdio.h>

//运算符 算术运算符 关系运算符 逻辑运算符 位操作运算符 赋值运算符 条件运算符 逗号运算符 指针运算符 求字节运算符(sizeof) 特殊运算符

//算术运算符:

//+ - \* / %(取模)

//自增自减: ++ --

//关系运算符: > >= == <= !=

//逻辑运算符: && || !

//三目运算符: ? :

//赋值运算符: = += -= \*= /= %=

//取地址符: &

//逗号运算符:,

//表达式: 由常量 变量 以及运算符组成的式子

//单个的常量变量也是表达式

//如: a+b a\*b (a+b,a\*b) a&&b a>b !a

//a+b;//表达式语句

//+ - \* / %(取模)

/\*int main(int argc, const char \* argv[])

{

int a, b,c;

a = 10;

b = 20;

printf("a+b == %d\n", c=a+b);

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

printf("a\*b == %d\n", a\*b);

printf("a-b == %d\n", a-b);

printf("a%%b == %d\n", a%b);//第一个%表示转义

return 0;

}\*/

//自增自减: ++ --

/\*int main(int argc, const char \*argv[])

{

int a, b, c, d;

a = 10;

b = 1;

c = 5;

d = 20;

printf("a = %d\n", a++);//a++ <=> a+=1 a=a+1

printf("a = %d\n", a);

printf("b = %d\n", ++b);

printf("b = %d\n", b);

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

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

printf("d = %d\n", --d);

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

return 0;

}\*/

/\*int main(int argc, const char \*argv[])

{

int a= 20;

printf("%d %d %d %d\n", a++, a--, ++a, --a);//a的值是未定义的

return 0;

}\*/

//关系运算符: > >= == <= !=

//表达式成立, 值为1, 表达式不成立, 值为0

/\*int main(int argc, const char \*argv[])

{

int a= 20;

int b= 20;

printf("(a>b) == %d\n", a>b);//0 表达式不成立, 表达式的值为假

printf("(a<b) == %d\n", a<b);//1 表示表达式为真

printf("(a==b) == %d\n", a==b);

printf("(a>=b) == %d\n", a>=b);

printf("(a<=b) == %d\n", a<=b);

return 0;

}\*/

//逻辑运算符: && || !

// 真值表

// 表达式1 表达式2 ...表达式n 表达式

// 假 假 假

// 真 假 假

// 假 真 假

// 真 真 真

// 有一个表达式为假, 则整个表达式的值为假

/\*int main(int argc, const char \*argv[])

{

int a,b, c, d;

a= 30 ;

b = 20;

c =50;

d = 40;

printf("%d\n", (a>b)&&(c>d));

return 0;

}\*/

// ||

// 表达式1 表达式2,.... 表达式N 表达式

// 假 假 假

// 真 假 真

// 假 真 真

// 真 真 真

// 有一个表达式为真, 则整个表达式的值为真

/\*int main(int argc, const char \*argv[])

{

int a, b, c,d;

a= 50 ; b= 20 ;c = 60 ;d = 40;

printf("%d\n", (a>b)||(c>d));

return 0;

}\*/

//!

// 表达式1 表达式

// 真 假

// 假 真

/\*int main(int argc, const char \*argv[])

{

int a = 0;

printf("%d\n", !a);

return 0;

}\*/

//三目运算符: ? :

//表达式1? 表达式2: 表达式3

//表达式1的值为真,则整个表达式的值为表达式2的值

//表达式1的值为假,则整个表达式的值为表达式3的值

/\*int main(int argc, const char \*argv[])

{

int a , b, c;

a = 10;

b = 20;

c = 30;

printf("%d\n", a<b? a: b);

printf("%d\n", a>b? a : b>c? b:c);//右结合性

printf("%d\n", a>b? a :(b>c?b :c));

}\*/

//赋值运算符: = += -= \*= /= %=

/\*int main(int argc, const char \*argv[])

{

int a;

a = 10;

printf("%d\n", a+=10);// a= a+10 20

printf("%d\n", a-=10);// a= a-10 10

printf("%d\n", a\*=10);// a = a\*10 100

printf("%d\n", a/=10);// a = a/10 10

printf("%d\n", a%=10);// a = a%10 0

return 0;

}\*/

//取地址符: &

//逗号运算符:,

//表达式1,表达式2,表达式3,表达式4,.......,表达式n

//表达式N的值为表达式的值

/\*int main(int argc, const char \*argv[])

{

int a, b, c;

printf("%d", (a=10,b=20,c=30));

return 0;

}\*/

//运算符优先级

//括号>单目运算符>算术运算符>关系运算符>逻辑运算符>三目运算符>赋值运算符>逗号运算符

//\* / % > + -

//&& > ||

//加括号能够改变优先级

/\*int main(int argc, const char \*argv[])

{

int a, b,c,d;

a = 10;

b = 20;

c = 30;

d = 40;

printf("%d\n", a+b\*c/d);

return 0;

}\*/

// 短整型向长整形看齐

// 整形向浮点型看齐

// 单精度向双精度看齐

//

//int main(int argc, const char \*arv[])

//{

// char a = 0x12;

// short c = 0x13;

// long d = 0x78;

//

// int b = 0x00000087;

// printf("%d\n",a+b);//隐式转换

// printf("%d\n",a+c);

// printf("%ld\n", a+d);

// return 0;

//}

//位扩展

//对于正数高位补0 对于负数 高位补1

// 1111 1111 1111 1111 1111 1111 1001 0010

//+0000 0000 0000 0000 0000 0000 1000 0111

//

/\*int main(int argc, const char \*argv[])

{

int a= 20;

float f= 3.14;

double f1 = 4.345673456;

printf("%d\n",(int)(a+f));//强制转换

printf("%f\n",a+f);

printf("%f\n", f+f1);

return 0;

}\*/

//<http://192.168.88.73/1408>

//

//int a= 511 0x1ff

// char b =a; b = 0xff = -1

//

// float a = 3.14

// int b = (int)a%2;

第四天

(上午) #include <stdio.h>

//单分支结构

//if(表达式) 表达式成立(真),则执行括号中的代码块,否则不执行

//{

// 代码块;

//}

//

/\*int main(int argc, const char \* argv[])

{

int a;

scanf("%d", &a);

if (a== 10) {

printf("hello world\n");

}

return 0;

}\*/

/\*#include <stdlib.h>

int main(int argc, const char \*argv[])

{

int score;

printf("请输入学生的分数 : ");

scanf("%d", &score);

if (score < 0 || score >100) {

printf("输入错误\n");

exit(0);

//return -1;

}

if (score>=0 && score <60) {

printf("不及格\n");

}

if (score>=60 && score <70) {

printf("该学生的成绩凑合\n");

}

if (score >=70 && score <80) {

printf("该学生的成绩正常\n");

}

if (score >= 80 && score < 90) {

printf("该学生的成绩良好\n");

}

if (score >= 90 && score <= 100) {

printf("该学生的成绩优秀\n");

}

return 0;

}\*/

//双分支结构

//if(表达式)表达式为真, 执行语句列表1,为假执行语句列表2

//{

// 语句列表1;

//}

//else

//{

// 语句列表2;

//}

/\*int main(int argc, const char \*argv[])

{

int a;

scanf("%d", &a);

if (a>=60 && a<=100)

{

printf("及格\n");

}

else{

printf("不及格\n");

}

}\*/

/\*int main(int argc, const char \*argv[])

{

int year;

printf("请输入一个年份: ");

scanf("%d", &year);

if ((year%4 == 0 && year%100!=0) || (year%400==0))

{

printf("%d年是闰年\n", year);

}

else

{

printf("%d年是平年\n" ,year);

} return 0;

}\*/

//if(表达式1)

//{

// 语句列表1;

//}

//else if(表达式2)

//{

// 语句列表2;

//}

//else if(表达式3)

//{

// 语句列表3;

//}

//...

//else if(表达式n)

//{

// 语句列表n;

//}

//else

//{

// 语句列表;

//}

/\*int main(int argc, const char \*argv[])

{

int month;

printf("请输入月份: ");

scanf("%d", &month);

if (month<1 || month >12) {

printf("输入月份错误\n");

return -1;

}

if (month >=2 && month <= 4) {

printf("春季\n");

}

else if (month >= 5 && month <= 7)

{

printf("夏季\n");

}

else if (month >= 8 && month <= 10)

{

printf("秋季\n");

}

else

{

printf("冬季\n");

}

return 0;

}\*/

//if(表达式)

//{

// if(表达式)

// {

//

// }

// else

// {

//

// }

//}

//else

//{

// if(表达式)

// {

//

// }

// else

// {

//

// }

//}

/\*int main(int argc, const char \*arv[])

{

int sex, choice;

printf("(1)酒心巧克力 (2)扇贝巧克力 (3)果仁巧克力\n");

printf("请选择你喜欢的巧克力(1, 2, 3): ");

scanf("%d", &choice);

printf("请输入您的性别(1: 男生 2: 女生): ");

scanf("%d", &sex);

if (sex ==1) {

if (choice == 1) {

printf("您是一个恋爱高手, 很会讨女孩子喜欢\n");

}

else if (choice ==2)

{

printf("您是一个胆小的人, 总是担心自己不够好\n");

}

else if(choice ==3)

{

printf("您对女孩子一心一意, 但是她总是感觉不到\n");

}

else{

printf("您输入了一个错误选项\n");

}

}

else if(sex ==2)

{

if (choice == 1) {

printf("在恋爱中, 您往往处于劣势\n");

}

else if (choice ==2)

{

printf("您是一个浪漫的人, 恋爱中能够很好的调节气氛\n");

}

else if(choice ==3)

{

printf("您是一个女强人, 恋爱中总是忽略对方的感受\n");

}

else{

printf("您输入了一个错误选项\n");

}

}

else

{

printf("输入错误\n");

}

return 0;

}\*/

/\*#include <math.h>

int main(int argc, const char \*argv[])

{

float num;

printf("请输入您的月消费金额 : ");

scanf("%f", &num);

if (num >= 0) {

if (num == 0) {

printf("零消费\n");

}

else if (num >= 1 && num <= 500)

{

printf("低消费\n");

}

else if(num >500 && num <= 1000)

{

printf("中等消费\n");

}

else

{

printf("高等消费\n");

}

}

else

{

printf("您不仅没有花钱, 还挣钱了 %f\n", fabs(num));

}

return 0;

}\*/

//跳出switch

/\*switch(表达式)

{

case 常量表达式1:

语句列表1;

break;

case 常量表达式2:

语句列表2;

break;

case 常量表达式3:

语句列表3;

break;

case 常量表达式4:

语句列表4;

break;

...

case 常量表达式n:

语句列表n;

break;

default:

语句列表;

break;

}\*/

/\*int main(int argc, const char \*argv[])

{

float a, b;

char ch;

scanf("%f%c%f", &a, &ch, &b);

switch (ch) {

case '+':

printf("a+b == %f\n", a+b);

break;

case '-':

printf("a-b == %f\n" ,a-b);

break;

case '\*':

printf("a\*b == %f\n", a\*b);

break;

case '/':

printf("a/b == %f\n", a/b);

break;

default:

break;

}

return 0;

}\*/

/\*int main(int argc, const char \*argv[])

{

int a;

scanf("%d", &a);

switch (a) {

case 1:

printf("hello one\n");

case 2:

printf("hello two\n");

case 3:

printf("hello three\n");

case 4:

printf("hello four\n");

case 5:

printf("hello five\n");

default:

break;

}

return 0;

}\*/

//<http://192.168.88.54/1408/>

/\*int main(int argc, const char \*argv[])

{

int month;

scanf("%d", &month);

switch (month) {

case 2:

case 3:

case 4:

printf("春天\n");

break;

case 5:

case 6:

case 7:

printf("夏天\n");

break;

case 8:

case 9:

case 10:

printf("秋天\n");

break;

case 11:

case 12:

case 1:

printf("冬天\n");

default:

break;

}

return 0;

}\*/

/\*int main(int argc, const char \*argv[])

{

int a=10;

int b=20;

a=a+b;

b=a-b;

a=a-b;

printf("%d %d\n" , a, b);

return 0;

}\*/

(下午) #include <stdio.h>

//goto

//标号 + :

//通常配合条件语句使用

//只能在同一个函数内使用goto 语句

//void func(void)

//{

//lable:

// printf("hello world\n");

//

//}

/\*int main(int argc, const char \* argv[])

{

int a;

scanf("%d", &a);

dece:

printf("hello world\n");

if (a>10) {

goto dece;

}

return 0;

}\*/

/\*int main(int argc, const char \*argv[])

{

int sum = 0, i=1;

number:

sum+=i;

i++;

if (i<=100)

{

goto number;

}

printf("%d\n", sum);

}\*/

/\*while(表达式)//判断表达式的值, 表达式的值为真,执行语句体, 为假就不执行

{

语句体;

}\*/

/\*int main(int argc, const char \*argv[])

{

int i=1, sum=0;

while (i<=100) {

sum += i;

i++;

}

printf("%d\n", sum);

}\*/

/\*int main(int argc, const char \*argv[])

{

int cnt = 0;

while (getchar() != '\n') {

cnt++;

}

printf("%d\n", cnt);

return 0;

}\*/

/\*do{

语句块;

}while();\*/

/\*int main(int argc, const char \*argv[])

{

int sum=0, i =1;

do{

if (i%2) {

sum += i;

}

i++;

}while (i<=100);

printf("%d\n", sum);

return 0;

}\*/

//int main(int argc, const char \*argv[])

//{

// int i = 0;

// while (i< 10) {

// printf("i=%d hello world\n", i);

// i++;

// }

//}

/\*int main(int argc, const char \*argv[])

{

int i = 0;

do{

printf("i = %d hello world\n", i);

//i++;

}while (i++<10);

return 0;

}\*/

/\*

for(表达式1;表达式2;表达式3)

{

语句块;

}

\*/

/\*int main(int argc, const char \*argv[])

{

int i;

for (i=0; i<10; i++)

{

printf("i = %d hello world\n", i);

}

return 0;

}\*/

/\*int main(int argc, const char \*argv[])

{

int i =0;

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

printf("hello world\n");

}

return 0;

}\*/

//循环中用break跳出一层循环

/\*int main(int argc, const char \*argv[])

{

int i= 0;

for (; ; i++) {

if (i<10) {

printf("hello world\n");

}

else

break;

}

return 0;

}\*/

/\*int main(int argc, const char \*argv[])

{

int i= 0;

for (; ; ) {

if (i<10) {

printf("hello world\n");

}

else

break;

i++;

}

return 0;

}\*/

/\*int main(int argc ,const char \*argv[])

{

int a;

int i;

scanf("%d", &a);

printf("%d=", a);

for (i=2; a!=1; i++) {

if (a%i==0) {

printf("%d ", i);

a/=i;

i--;

}

}

return 0;

}\*/

/\*int main(int argc, const char \*argv[])

{

int i, j;

for (i = 0; i<6; i++) {

for (j= 0; j<20; j++) {

printf("\*");

}

printf("\n");

}

return 0;

}\*/

/\*int main(int argc, const char \*argv[])

{

int i, j;

for (i=1; i<=9; i++) {

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

printf("%d\*%d=%d ", i,j,i\*j);

}

printf("\n");

}

return 0;

}\*/

int main(int argc, const char \*argv[])

{

int i, j;

for (i= 1; i<=4; i++) {

for (j=1; j<=5; j++) {

printf("%d ", i\*j);

}

printf("\n");

}

return 0;

}

第五天

(上午) #include <stdio.h>

//break;

//用在switch 中跳出switch语句, 执行后面的程序

//用在循环中, 跳出当前循环,只跳出一层循环

//contiue :

/\*int main(int argc, const char \* argv[])

{

int i;

for (i=0; i<10; i++) {

if (i==5) {

continue;

}

printf("i = %d\n", i);

}

return 0;

}\*/

//1.求1~100之间的素数(质数)

/\*int main(int argc, const char \*argv[])

{

int i,j;

for (i=1; i<=100; i++) {

for (j=2; j<i; j++) {

if (i%j==0) {

break;

}

}

if (!(j==i)) {

continue;

}

printf("%d ",i);

}

return 0;

}\*/

//2.输入一个大写字母，如F

//比如：

//输入:F

//输出：

//A

//ABA

//ABCBA

//ABCDCBA

//ABCDEDCBA

//ABCDEFEDCBA

//

//输入:C

//输出:

//A

//ABA

//ABCBA

/\*int main(int argc, const char \*argv[])

{

int i, j;

char ch;

scanf("%c", &ch);

for (i=0; i<ch-'A'+1; i++) {

for (j=0; j<=i; j++) {

printf("%c",'A'+j);

}

for (j=i; j>0; j--) {

printf("%c", 'A'+j-1);

}

printf("\n");

}

return 0;

}\*/

//

/\*int main(int argc, const char \*argv[])

{

char ch;

int cnt=0;

//while (scanf("%c",&ch),(ch !='\n'))

while (ch=getchar(),(ch !='\n'))

{

if ((ch>='A'&&ch<='Z')||(ch>='a' && ch <= 'z')) {

cnt++;

}

}

printf("%d\n", cnt);

return 0;

}\*/

//4. 输入年月日, 判断是一年中的第多少天

/\*int main(int argc, const char \*argv[])

{

int i;

int year,month,day;

int totalday=0;

int tmp;

scanf("%d,%d,%d",&year,&month,&day);

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

switch (i) {

case 1:

case 3:

case 5:

case 7:

case 8:

case 10:

case 12:

tmp=31;

break;

case 2:

if ((year % 4==0 && year%100!=0)||(year%400==0)) {

tmp=29;

}

else

tmp=28;

break;

case 4:

case 6:

case 9:

case 11:

tmp=30;

break;

default:

return -1;

}

totalday+=tmp;

}

totalday +=day;

printf("%d\n",totalday);

return 0;

}\*/

//流程控制（for求阶乘的和 ） 输入n(int类型)，打印1！+2！+3！+4！+5！+。。。+n!的值

/\*

比如：

输入：2

输出：

3

输入：3

输出:

\*/

/\*int main(int argc, const char \*argv[])

{

int i;

int Sn=0;

int tmp=1;

int n;

scanf("%d",&n);

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

tmp \*=i;

Sn +=tmp;

}

printf("%d\n", Sn);

return 0;

}\*/

//7.

//流程控制（数值序列2） 输出数字序列 2/1，3/2，5/3，8/5，13/8，21/13...，输出个数由键盘输入。注意输入使用scanf输入

/\*

比如：

输入 3输出为

2/1

3/2

5/3

输入 4 输出为

2/1

3/2

5/3

8/5

\*/

/\*int main(int argc, const char \*argv[])

{

int n;

int i;

int a=1, b = 1;

scanf("%d", &n);

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

printf("%d/%d ",a+b, a);

int tmp;

tmp = a+b;

b = a;

a=tmp;

}

printf("\n");

return 0;

}\*/

//8.

//辗转相除法求最大公约数

//

/\*int main(int argc, const char \*argv[])

{

int a, b;

scanf("%d%d", &a, &b);

int tmp;

while ((tmp=a%b)) {

//int tmp =a%b;

a=b;

b=tmp;

}

printf("%d\n",b);

return 0;

}\*/

//9.一球从 100 米高度自由落下,每次落地后反跳回原高度的一半;再落下,求它在第10 次落地时,共经过多少米

/\*int main(int argc, const char \*argv[])

{

float s=0.00;

int i;

float t=100.00;

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

s+=t;

t/=2;

}

s+=100;

printf("%f\n",s);

return 0;

}\*/

//3.输入一个大写字符，如F

//比如：

//输入:F

//输出：

//FEDCBA

//EDCBAB

//DCBABC

//CBABCD

//BABCDE

//ABCDEF

//

//输入 B

//输出:

//BA

//AB

//int main(int argc, const char \*argv[])

//{

// char ch;

// int i, j;

// scanf("%c", &ch);

// for (i=0; i<ch-'A'+1; i++) {

// for (j=i; j<ch-'A'+1; j++) {

// printf("%c",ch-j);

// }

// for (j=0; j<i; j++) {

// printf("%c",'A'+1+j);

// }

// printf("\n");

// }

// return 0;

//}

//5.

//流程控制（for循环） 输入一个大写字母，如 F，输出

//\

//比如：

//\

//输入：F

//\

//输出：

//F

//EFE

//DEFED

//CDEFEDC

//BCDEFEDCB

//ABCDEFEDCBA

/\*int main(int argc, const char \*argv[])

{

int i;

int year,month,day;

int totalday=0;

int tmp;

scanf("%d %d %d",&year,&month,&day);

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

switch (i) {

case 1:

case 3:

case 5:

case 7:

case 8:

case 10:

case 12:

tmp=31;

break;

case 2:

if ((year % 4==0 && year%100!=0)||(year%400==0)) {

tmp=29;

}

else

tmp=28;

break;

case 4:

case 6:

case 9:

case 11:

tmp=30;

break;

default:

return -1;

}

totalday+=tmp;

}

totalday +=day;

printf("%d\n",totalday);

return 0;

}\*/

#include"stdio.h"

int main(int argc, const char \*argv[])

{

char ch;

scanf("%c",&ch);

for (int i=0; i<=ch-'A'; i++) {

int j;

for (j=0; j<ch-'A'-i; j++) {

printf(" ");

}

for (j=0; j<=i; j++) {

printf("%c",'A'+j);

}

for (j=0; j<i; j++) {

printf("%c",'A'+i-j-1);

}

printf("\n");

}

}

(下午)

#include <stdio.h>

//#include "file.h"

//函数

//函数又叫方法,是指实现某项功能或完成某项任务的代码块

/\*void show(void)

{

printf(" \* \n");

printf(" \*\*\* \n");

printf(" \*\*\*\*\* \n");

printf("\*\*\*\*\*\*\*\n");

}\*/

//函数的主体从大括号开始,从大括号结束

//函数组成

//main函数,是给系统调用的函数

//函数组成: 返回值, 函数名, 传入参数

//如: 实现两个整数相加,返回它们的和

/\*void show(void)

{

printf(" \* \n");

printf(" \*\*\* \n");

printf(" \*\*\*\*\* \n");

printf("\*\*\*\*\*\*\*\n");

}

int add(int x, int y)

{

return x+y;

}

int main(int argc, const char \* argv[])

{

int val;

show();

val = add(3, 5);

printf("%d\n",val);

return 0;

}\*/

//函数组成: 函数名 函数接口 函数体 返回值

//函数调用

//带参数的函数

//函数的返回值就是函数运行的结果

//形式参数也叫参变量,实际上就是变量

//实际参数是实际存在的数值

//形参相当于函数中定义的变量,调用函数传递参数的过程相当于定义形参的变量用实参的值来初始化

//我们把在函数内部定义的变量称为局部变量

void print\_time1(int hour, int minute)

{

printf("%02d:%02d\n", hour, minute);

printf("%p %p\n", &hour, &minute);

}

void print\_time2(int hour, int minute)

{

print\_time1(hour, minute);

printf("%p %p\n", &hour, &minute);

}

int main(int argc, const char \*argv[])

{

int hour, minute;

hour = 12;

minute = 43;

printf("%p %p\n", &hour, &minute);

print\_time2(hour,minute);

return 0;

}

//局部变量就是在函数体内部声明的变量,它只在函数体内部有效,也就是说只能在本函数内部使用它,在本函数外是不能使用的,只有在函数被调用的使用,变量才会在内存上分配空间,调用结束,内存释放;

//测试局部变量每次分配内存位置不确定

//全局变量是在函数体外部定义的变量(包括main函数),它不属于哪个函数,而是属于源程序,因此全局变量可以为程序中得所有函数调用,它的有效范围从定义开始到源程序结束;

//全局变量的生命很长,它在程序的整个执行过程中都占用内存,当程序结束时,该变量的"寿命"才结束,系统这才释放它所占用的内存

//全局变量的缺点:值容易被修改

//作用域,局部变量的作用域是从变量的定义开始到右大括号结束,全局变量的作用域是从变量的定义开始到源程序的结束

//函数的声明及定义

//函数的声明就是告诉编译器知道有什么样的函数存在,可以在后面找到它,假如没有声明,那么就有可能会出现找不到函数的错误

//库函数的使用: 条用库函数时 , 需要包含库函数的头文件

//函数的递归调用

//一个函数在它的函数体内调用它本身,称之为递归函数

//C语言中允许递归调用, 递归调用中,主调函数又是被调函数

//

//为了防止递归无终止进行,必须在函数体内有终止递归调用的条件

//常用的办法就是加条件判断语句,满足某种条件后就不在递归调用,逐层返回

//数学上有这样的定义 n的阶乘等于n乘以n-1的阶乘,n-1的阶乘等于(n-1)\*(n-2)!

//这样下去永远也没有完,所以需要定义一个基础条件

//基础条件: 0的阶乘为1

//n! = n\*(n-1)!

//0! = 1

//因此 3! = 3\*2! = 3\*2\*1!= 3\*2\*1\*0!= 3\*2\*1 = 6;

//正因为有了基础条件,才没有无休止的进行下去

//那么我们用程序怎么样去实现呢

//

//一个过程或函数在其定义或说明中又直接或间接调用自身的一种方法，它通常把一个大型复杂的问题层层转化为一个与原问题相似的规模较小的问题来求解，递归策略只需少量的程序就可描述出解题过程所需要的多次重复计算，大大地减少了程序的代码量。递归的能力在于用有限的语句来定义对象的无限集合。用递归思想写出的程序往往十分简洁易懂。

//一般来说，递归需要有边界条件、递归前进段和递归返回段。当边界条件不满足时，递归前进；当边界条件满足时，递归返回。

//注意：

//(1) 递归就是在过程或函数里调用自身;

//(2) 在使用递增归策略时，必须有一个明确的递归结束条件，称为递归出口，否则将无限进行下去（死锁）。

//

//递归算法一般用于解决三类问题：

//(1)数据的定义是按递归定义的。(Fibonacci函数)

//(2)问题解法按递归算法实现。(回溯)

//(3)数据的结构形式是按递归定义的。(树的遍历，图的搜索)

//

//递归的缺点：

//递归算法解题的运行效率较低。在递归调用的过程当中系统为每一层的返回点、局部量等开辟了栈来存储。递归次数过多容易造成栈溢出等。

int factorial(int n)

{

if (n==0)

{

return 1;

}

else

{

int recurse = factorial(n-1);

int result = recurse \* n;

return result;

}

}

//斐波那契数列

//1 1 2 3 5 8 13 21 34 55 89 144 ....

int fibonacci(int n)

{

if (n==0||n==1) {

return 1;

}

else {

int val = fibonacci(n-1) + fibonacci(n-2);

return val;

}

}

//1、编写递归函数求两个正整数a和b的最大公约数（GCD，Greatest Common Divisor），使用Euclid算法：

//

//如果a除以b能整除，则最大公约数是b。

//

//否则，最大公约数等于b和a%b的最大公约数。

//

//Euclid算法是很容易证明的，请读者自己证明一下为什么这么算就能算出最大公约数。最后，修改你的程序使之适用于所有整数，而不仅仅是正整数。

int euclid(int a, int b)

{

if(a%b==0)

{

return b;

}

else

{

return euclid(b, a%b);

}

}

//2、 趣味问题——年龄。有5个人坐在一起，问第五个人多少岁？他说比第4个人大2岁。问第4个人岁数，他说比第3个人大2岁。问第三个人，又说比第2人大两岁。问第2个人，说比第一个人大两岁。最后问第一个人，他说是10岁。请问第五个人多大？用递归算法实现。

int age(int n)

{

if (n==1) {

return 10;

}

else{

return age(n-1)+2;

}

}

/\*int main(int argc, const char \*argv[])

{

//printf("%d\n", factorial(3));

// for (int i= 0; i<20; i++) {

// printf("%d ",fibonacci(i));

// }

printf("%d\n",euclid(319, 377));

printf("age = %d\n", age(5));

return 0;

}\*/