1、书接上回，运算符的使用

1.1 sizeof运算符

```
1     sizeof运算符用来计算数据类型的占用内存空间的大小，单位为字节。
2
3 格式：
4     sizeof(数据类型名);
5
6     64位系统：返回long int类型，使用格式化字符-》%ld
7
8     32位系统：返回int类型，使用格式化字符-》%d
```

```c
#include <stdio.h>

int main(int argc, const char *argv[])
{
//#define  X86_64
#ifdef  X86_64
    printf("unsigned char size = %ld\n", sizeof(unsigned char));
    printf("unsigned short size = %ld\n", sizeof(unsigned short));
    printf("unsigned int size = %ld\n", sizeof(unsigned int));
    printf("unsigned long size = %ld\n", sizeof(unsigned long));
    printf("unsigned long long size = %ld\n", sizeof(unsigned long long));
```

```
12      printf("float size = %ld\n",
    sizeof(float));
13      printf("double size = %ld\n",
    sizeof(double));
14      printf("long double size = %ld\n",
    sizeof(long double));
15  #else
16      printf("unsigned char size = %d\n",
    sizeof(unsigned char));
17      printf("unsigned short size =
    %d\n", sizeof(unsigned short));
18      printf("unsigned int size = %d\n",
    sizeof(unsigned int));
19      printf("unsigned long size = %d\n",
    sizeof(unsigned long));
20      printf("unsigned long long size =
    %d\n", sizeof(unsigned long long));
21      printf("float size = %d\n",
    sizeof(float));
22      printf("double size = %d\n",
    sizeof(double));
23      printf("long double size = %d\n",
    sizeof(long double));
24
25  #endif
26      return 0;
27  }
28
```

```
linux@ubuntu:~/DC23032/day05$ gcc 01sizeof.c
linux@ubuntu:~/DC23032/day05$
linux@ubuntu:~/DC23032/day05$ ./a.out
unsigned char size = 1
unsigned short size = 2
unsigned int size = 4
unsigned long size = 8
unsigned long long size = 8
float size = 4
double size = 8
long double size = 16
```

64位系统输出结果

1.2 逗号运算符

```
格式:
    (表达式1, 表达式2, 表达式3,....);
    最后一个表达式的结果被返回。
```

```c
#include <stdio.h>

int main(int argc, const char *argv[])
{
    int a = 10;
    int b = 20;
    int c = 30;

    int d = (a++, ++b, --c); // a = 11 b = 21 c = 29 d = 29
    // 逗号运算符里边的所有的表达式都被执行了,
    // 只是最后一个表达式的结果被返回。
```

```
12      // 在使用逗号运算符时，必须在最外边添加
    (), 不可以省略
13
14      printf("a = %d, b = %d, c = %d, d =
    %d\n",
15             a, b, c, d);
16
17      // int f = ++a,++b; // 错误
18      return 0;
19  }
20
```

1.3 运算符的优先级

```
1   运算符优先级从高到低：  单算移关与，异或逻条赋，
    逗号
2       单 ：+(正数) -(负数) ++ -- ~(按位取反)
    !(非)
3       算 ：  + - * / %
4       移 : << >>
5       关 : > < >= <= != ==
6       与 : &(按位与)
7       异 : ^(按位异或)
8       或 : |(按位或)
9       逻 : && ||
10      条 : ?:
11      赋 : = += -= *= /= %= &= |= ^= <<=
    >>=
12      逗号 : ,
13
```

同级运算符结合的问题：
    单条赋：从右到左结合，
        比如：a = b = 1; 从右到左结合，先把1赋值给b,再把b赋值给a
    剩余的都是从左到右结合
        比如：a && b && c; 从左到右结合，先进行a && b，结果在&&c
可以参考《运算符优先级.docx》文档，已经上传有道云笔记中。

2、常用的输入输出函数

学习任何一个函数，都要先掌握函数的功能，参数，返回值。
系统提供的函数都有可以使用man查看对应的帮助手册。
man man ---> 查看man的帮助手册
        1   可执行程序或 shell 命令
        2   系统调用(内核提供的函数)
        3   库调用(程序库中的函数)
        4   特殊文件(通常位于 /dev)
        5   文件格式和规范，如 /etc/passwd
        6   游戏
        7   杂项(包括宏包和规范，如 man(7), groff(7))
        8   系统管理命令(通常只针对 root 用户)
        9   内核例程 [非标准

```
13
14 man 3 库函数的名字，比如 man 3 putchar
```

2.1 putchar/getchar函数

2.1.1 putchar函数

```
1  #include <stdio.h>
2  int putchar(int c);
3   |       |       |---> 函数的形参，调用函数时需要传递一个实参。
4   |       |-------> 函数的名字
5   |---> 函数返回值的类型，可以使用对应类型的变量接收函数的返回值
6
7  功能：输出一个字符到终端
8  参数：
9      @ c : 要输出的字符
10         1> 可以是一个字符常量
11         2> 可以是一个字符类型的变量
12         3> 可以是一个字符对应的ASCII码值
13         4> 可以是一个整数，输出整数对应ASCII码表中的字符，
14                 如果超过ascii码表的最大值，输出一个乱码。
15         5> 可以是一个经过运算的表达式，表达式的结果是一个整数。
16 返回值：
17     成功返回写到终端上的字符的对应的整数值，
```

```
18      失败返回错误码EOF(-1);
19
```

```c
#include <stdio.h>
int main(int argc, const char *argv[])
{
    // 1. 可以传递字符常量
    int retVal = putchar('A');
    // 有可能失败，因此需要进行出错处理
    if (retVal == -1)
    {
        printf("输出字符失败\n");
        return -1;
    }
    else
    {
        printf("输出字符成功: %c --- %d\n", retVal, retVal);
    }

    // 2. 可以传递字符类型的变量
    char ch = 'a';
    putchar(ch);
    putchar('\n');      // 输出一个换行符

    // 3. 可以传递一个整数值
    putchar(97);    // a 的ascii码值为97
    putchar(10);    // /n 的ascii码值为10


    // 4. 可以是一个结果为整型的表达式
```

```
28      putchar('A'+33);
29      putchar(5+5);
30
31      return 0;
32  }
33
```

2.2.2 getchar函数

```
1  #include <stdio.h>
2  int getchar(void);
3  功能：从终端获取一个字符
4  参数：
5      无
6  返回值：
7      成功：返回接收字符对应的ascii码值
8      失败：返回错误码EOF(-1)
9
```

```
 1  #include <stdio.h>
 2  int main(int argc, const char *argv[])
 3  {
 4      char a, b, c;
 5      // 使用getchar连续接收3个字符
 6      a = getchar();
 7      b = getchar();
 8      c = getchar();
 9
10      printf("a = %c, b = %c, c = %c\n",
    a, b, c);
```

```
11
12      return 0;
13  }
14
```

```
 1  #include <stdio.h>
 2  int main(int argc, const char *argv[])
 3  {
 4      char a, b, c;
 5      // 使用getchar连续接收3个字符
 6      a = getchar();
 7      getchar(); // 处理垃圾字符，只能处理一
    个垃圾字符
 8      b = getchar();
 9      getchar();
10      c = getchar();
11
12      printf("a = %c, b = %c, c = %c\n",
    a, b, c);
13
14      return 0;
15  }
16
```

2.2 puts/gets函数

2.2.1 puts函数

```
#include <stdio.h>
int puts(const char *s);
功能：向终端输出一个字符串
参数：
    @ s ：输出字符串的首地址
        1> 可以是字符串常量
        2> 可以是字符数组的名字
        3> 可以是指向一个字符串的字符指针
返回值：
    成功：返回一个非负的整数，发送成功的字符串的个数(字符串长度+1)
    失败：返回错误码EOF(-1)

注：puts函数在发送字符完成之后，会自动补发一个'\n'换行符
```

```c
#include <stdio.h>

int main(int argc, const char *argv[])
{
    // c语言的字符串的结尾为'\0',输出字符串时遇到'\0'停止输出

    // 1. 可以传递一个字符串常量
    // 字符串发生结束之后自动补发一个'\n'字符
    int retVal = puts("hello world!!!");
```

```c
    // EOF系统提供的错误码，本质就是一个宏定义
    // #define  EOF  -1
    if ( retVal != EOF)
    {
        printf("发生字符串成功: %d\n", retVal);
    }
    else
    {
        printf("发生字符串失败\n");
        return EOF;
    }

    puts("");  // 发生一个换行符

    // 2. 可以传递一个字符数组
    // 数组名就表示字符数组的首地址
    char name[20] = "zhangsan";
    puts(name);

    // 3. 可以传递一个指向字符串的字符指针
    char *str_p = "nihao"; // 字符指针指向一个字符串串常量
    // str_p中存放的就是字符串常量的首地址
    puts(str_p);


    // 输出字符串时遇到'\0'结束
    puts("hello\0world");  // 输出hello
    return 0;
```

```
}

```

2.2.2 gets函数

```
#include <stdio.h>

char *gets(char *s);
功能：从标准输入获取一个字符串
参数：
    @ s ：将接收的字符串存到哪个地址中
返回值：
    成功：返回s，及接收字符串的缓冲区的首地址
    失败：返回NULL

注：gets可以接收终端所有输入的字符包括空格，当按下回车时，输入结束，
    最终会在字符串的结尾补一个'\0'.
    即使输入的字符串的长度大于接收字符串缓冲区的长度，也可以接收，
    并不会进行越界的检查，但你在次打印字符串时，就会越界访问内存空间，
    就会出现段错误。
```

```
#include <stdio.h>

int main(int argc, const char *argv[])
{
    // 将接收的字符串存放一个字符数组的缓冲区
```

```
6       char buffer[20];    //缓冲区最多接收19个字符，最后存一个'\0'
7
8       printf("please input a string, char number < 20 > ");
9
10      // 从终端获取字符串，有多少接收多少，并不会进行越界的检查
11      // 如果越界之后，访问非法空间时，执行程序可能会报段错误
12      // 一旦出现段错误一般都是访问了非法的内存空间。
13      // 在实际的开发中一般解bug遇到最多的错误就是段错误，
14      // 这种错误在程序编译时不会报错，执行时才会报错。
15      char *retVal = gets(buffer);
16      if (retVal == NULL)
17      {
18          puts("从标准输入获取字符串失败");
19          return -1;
20      }
21
22      puts(buffer);
23
24      return 0;
25  }
26
```

2.3 printf/scanf函数

2.3.1 printf函数

```
#include <stdio.h>

int printf(const char *format, ...);

功能：向终端输出一个格式化字符串
参数：
    @ format ： 参数控制格式的字符串
    @ ... ： 可变参数， 可变参数有几个，由format中的格式字符决定，
            可变参数会依次替换format中的格式化占位符，
            要求参数的类型和格式化字符要一一对应。
返回值：
    成功返回正数，失败返回负数
```

格式化字符：

附加格式说明符

修饰符	功能
m	输出数据域宽，数据长度<m，左补空格；否则按实际输出
.n	对实数，指定小数点后位数（四舍五入）
	对字符串，指定实际输出位数
-	输出数据在域内左对齐（缺省右对齐)
+	指定在有符号数的正数前显示正号(+)
0	输出数值时指定左面不使用的空位置自动填0
#	在八进制和十六进制数前显示前导0，0x
l	在d, o, x, u前，指定输出精度为long型
l	在e, f, g前，指定输出精度为double型

```c
#include <stdio.h>

int main(int argc, const char *argv[])
{
    int x = 0x12345678;
    printf("Dct x = %d\n", x);
    printf("Dct x = %u\n", x);
    printf("Oct x = %o\n", x);
    printf("Oct x = %#o\n", x);
    printf("Hex x = %x\n", x);
    printf("Hex x = %#x\n", x);

    printf("char c = %c\n", 'A');
    printf("string str = %s\n", "hello world!");
```

```
15
16      double Pi = 3.141500;
17      printf("Pi = %lf\n", Pi); // 默认输
    出小数点后6位
18      printf("Pi = %le\n", Pi);
19
20      printf("Pi= %g\n", Pi);  // 3.1415
21
22      printf("Pi = %10lf\n", Pi);
23      // 占10位宽度，默认右对齐，不足补空格，
24      // 超过10位宽度原样输出
25
26      printf("Pi = %010lf\n", Pi); //
27      // 占10位宽度，  默认右对齐，不足补0，
28      // 超过10位宽度原样输出
29
30      printf("Pi = %10.3lf\n", Pi);
31      // 浮点数，.3保留小数的后三位，不足3位补0
32      printf("Pi = %-10.5lf\n", Pi);
33      // - :  左对齐
34
35      return 0;
36  }
37
```

2.3.2 scanf函数

```
#include <stdio.h>

int scanf(const char *format, ...);
功能：从终端接收数据
参数：
    @ format ： 格式控制字符串
    @ ... ： 可变参数，需要提供的是变量的地址，
            只有提供的是地址时，才可以将通过终端读取的数据存到变量对应的空间中。
返回值：
    成功返回正数，失败返回负数。
```

格式输入函数

目前，scanf支持的格式字符很多，如下，

格式符号	作用
i, d	十进制整数
x, X	十六进制无符号整数
o	八进制无符号整数
u	无符号十进制整数
c	单一字符
s	字符串
e	指数形式浮点小数
f	小数形式浮点小数

```
#include <stdio.h>
```

```c

int main(int argc, const char *argv[])
{
    char c;
    // 变量名前加&符号，表示对变量取地址
    printf("please input char type data > ");
    scanf("%c", &c);
    printf("c = %c\n", c);

    short s;
    printf("please input short type data > ");
    scanf("%hd", &s);
    printf("s = %hd\n", s);


    int i;
    printf("please input int type data > ");
    scanf("%d", &i);
    printf("i = %d\n", i);


    float f;
    printf("please input float type data > ");
    scanf("%e", &f);
    printf("f = %e\n", f);

    char name[20];
```

```
    printf("please input string type data > ");
    scanf("%s", name);
    printf("name = %s\n", name);
    return 0;
}

```

```
#include <stdio.h>

int main(int argc, const char *argv[])
{
#if  0
    // 1. 可以同时接收多个参数
    int a, b;
    printf("please input two number(int,int) > ");
    // 当输入多个参数时，多个参数的输入可以以空格，tab键，回车分隔，
    // 系统会自动的吃掉空格,tab键和回车符。
    scanf("%d%d", &a, &b);
    printf("a = %d, b = %d\n", a, b);
#endif
#if  0
    // 2. scanf格式化字符串中除了格式化字符以外，
    // 还有很多其他的字符，需要按照格式化字符串原样进行输入，
    // 否则结果出错
    int year,mouth,day;
```

```
19      printf("please input year, mouth, day > ");
20      scanf("year=%d,mouth=%d,day=%d", &year,&mouth,&day);
21      printf("%d:%d:%d\n", year,mouth,day);
22  #endif
23  #if 0
24      // 3. scanf默认不可以获取带有空格的字符串,
25      // 默认遇到一个空格之后, 输入结束
26      char name[20];
27      // printf("please input your name > ");
28      // scanf("%s", name);
29      // printf("name = %s\n", name);
30
31      // 方式1: 使用gets可以接收带空格的字符串
32      // printf("please input your name > ");
33      // gets(name);
34      // printf("name = %s\n", name);
35
36      // 方式2 :  使用[^\n]符号, 除了\n以外的所有字符都接收
37      printf("please input your name > ");
38      scanf("%[^\n]s", name);
39      printf("name = %s\n", name);
40  #endif
41
```

```c
#if 0
    // 以下几种情况，当输入数据之后，系统无法自动吃掉
    // 空格，tab键，回车符，scanf函数处于阻塞无法退出，
    // 当按下任意一个字符之后，在回车就会退出。
    // 因此在使用scanf函数时，格式化字符串的结果不要有
    // 空格，\t \n \r
    int a, b, c;
    //scanf("%d\n", &a);
    scanf("%d\t", &b);
    // scanf("%d ", &c);
#endif

#if 0
    // 将\n \t \r，空格写到格式化字符的前边，
    // 系统会自动的吃掉\n，\t,\r,空格。
    int a, b, c;
    //scanf("\n%d", &a);
    scanf("\t%d", &b);
    // scanf(" %d", &c);
#endif
    return 0;
}

```

scanf接收字符时，处理垃圾字符的问题。

```c
#include <stdio.h>

int main(int argc, const char *argv[])
{
    int a, b;
    char oper;
    printf("请输入一个表达式(1 # r) > ");
    // 输入三个参数时，三个参数以空格分割，空格也是一个字符
    // 被%c接收之后，赋值给oper，对于程序来说，空格就是一个
    // 垃圾字符，需要处理垃圾字符
    // scanf("%d%c%d", &a, &oper, &b);

    // 方式1：使用%*c 吃掉一个垃圾字符
#if 0
    // scanf("%d%*c%c%*c%d", &a, &oper, &b);
    scanf("%d", &a);
    scanf("%*c");        // 吃掉垃圾字符
    scanf("%c", &oper);
    scanf("%*c");
    scanf("%d", &b);
#endif
    // 方式2：使用getchar()函数接收一个垃圾字符
    scanf("%d", &a);
    getchar();    // 吃掉垃圾字符
    scanf("%c", &oper);
    getchar();
    scanf("%d", &b);
```

```
28
29      // 在接收字符类型的变量时需要思考是否需要吃掉垃圾字符
30
31      if (oper == '+')
32      {
33          printf("a + b = %d\n", a + b);
34      }
35
36      if (oper == '-')
37      {
38          printf("a - b = %d\n", a - b);
39      }
40      return 0;
41  }
42
```

3、if...else分支语句

3.1 语法格式

```
1  1. if
2      if (表达式)
3      {
4          C语句块;
5      }
6
7      执行过程：“表达式”成立，执行“C语句块”，
```

```
        “表达式”不成立，跳过“C语句块”，向下继续执行

2. if...else...
    if (表达式)
    {
        C语句块1;
    }
    else
    {
        C语句块2;
    }

    执行过程：“表达式”成立，执行“C语句块1”，
        “表达式”不成立，执行“C语句块2”，执行完成之后在向下执行

3. if...else if ... else...
    if (表达式1)
    {
        C语句块1;
    }
    else if (表达式2)
    {
        C语句块2;
    }
    else if (表达式3)
    {
        C语句块3;
    }
    ...此处省略很多else if...
```

```
37
38      else            /* else 分支可以省略不写*/
39      {
40          C语句块n;
41      }
42
43      执行过程：判断表达式1是否成立，如果成立执行C语句块1，
44          执行完C语句块1之后，后边的所有的判断都不在进行判断，即使成立。
45          如果表达式1不成立，则判断表达式2，如果成立执行C语句块2，
46          执行完C语句块2之后，后边的所有的判断都不在进行判断，即使成立。
47          其他的都是一样的依次类推。
48          只有前边所有的表达式都不成立时，才会执行else分支对应的C语句块n.
49
50      表达式：只要是一个可以表示真假的表达式都可以。
51          一般为逻辑表达式，判断表达式
```

3.2 案例

```
1 1. 从终端输入一个字符，判断字符为大写字符，小写字符，数字。
2 #include <stdio.h>
3
4 int main(int argc, const char *argv[])
5 {
```

```
 6      char ch;
 7      printf("please input a char > ");
 8      scanf("%c", &ch);
 9
10      if (ch >= 'a' && ch <= 'z')
11      {
12          printf("%c是一个小写字符\n", ch);
13      }
14      else if (ch >= 'A' && ch <= 'Z')
15      {
16          printf("%c是一个大写字符\n", ch);
17      }
18      else if (ch >= '0' && ch <= '9')
19      {
20          printf("%c是一个数字\n", ch);
21      }
22      else
23      {
24          printf("%c既不是大小写字符，也不是
    数字\n",ch);
25      }
26      return 0;
27  }
28  ----------------------------------
29    #include <stdio.h>
30
31  int main(int argc, const char *argv[])
32  {
33      char ch;
34      printf("please input a char > ");
35      scanf("%c", &ch);
```

```
36      if ((ch < 'a' || ch > 'z') && (ch < 'A' || ch > 'Z') && (ch < '0' || ch > '9'))
37      {
38          printf("%c既不是大小写字符，也不是数字\n",ch);
39          return -1;
40      }
41
42
43      if (ch >= 'a' && ch <= 'z')
44      {
45          printf("%c是一个小写字符\n", ch);
46      }
47      else if (ch >= 'A' && ch <= 'Z')
48      {
49          printf("%c是一个大写字符\n", ch);
50      }
51      else
52      {
53          printf("%c是一个数字\n", ch);
54      }
55      return 0;
56  }
57
58
```

```
1  从终端输入薪资，选择不同的交通工具。
2  #include <stdio.h>
3  int main(int argc, const char *argv[])
4  {
```

```c
    int salary;
    printf("请输入你的薪资 > ");
    scanf("%d", &salary);


    if (salary < 0)
    {
        printf("输入的薪资不合理请重新输入\n");
        return -1;
    }

    if (salary <= 10000)
    {
        printf("地铁\n");
    }
    else if (salary <= 30000)
    {
        printf("出租\n");
    }
    else if (salary <= 50000)
    {
        printf("自驾\n");
    }
    else if (salary <= 100000)
    {
        printf("专职司机\n");
    }
    else
    {
        printf("自由选择\n");
```

```
    }


    return 0;
}

```

```
练习题1:
    在终端输入一个整数，用来表示学生的成绩
    输出学生成绩对应的等级
        [90,100]    A
        [80, 90)    B
        [70, 80)    C
        [60, 70)    D
        [0,60)      E
#include <stdio.h>
int main(int argc, const char *argv[])
{
    int score;
    printf("请输入成绩 > ");
    scanf("%d", &score);
    if ( score < 0 || score > 100)
    {
        printf("输入成绩不合理请重新输入\n");
        return -1;
    }

    if (score >= 90 && score <=100)
    {
        printf("等级A\n");
```

```
24      }
25      else if (score >= 80 && score < 90)
26      {
27          printf("等级B\n");
28      }
29      else if (score >= 70 && score < 80)
30      {
31          printf("等级C\n");
32      }
33      else if (score >= 60 && score < 70)
34      {
35          printf("等级D\n");
36      }
37      else
38      {
39          printf("等级E\n");
40      }
41
42      if (score >= 90 /*&& score <=100*/)
43      {
44          printf("等级A\n");
45      }
46      else if (score >= 80 /* && score <
    90 */)
47      {
48          printf("等级B\n");
49      }
50      else if (score >= 70 /* && score <
    80 */ )
51      {
52          printf("等级C\n");
```

```
    }
    else if (score >= 60 /* && score < 70 */)
    {
        printf("等级D\n");
    }
    else
    {
        printf("等级E\n");
    }

    // 如果if...else分支中只有1条c语句，可以省略{}
    if (score >= 90)
        printf("等级A\n");
    else if (score >= 80 && score < 90)
        printf("等级B\n");
    else if (score >= 70 && score < 80)
        printf("等级C\n");
    else if (score >= 60 && score < 70)
        printf("等级D\n");
    else
        printf("等级E\n");


    return 0;
}

```

```
练习题2：
```

```c
    在终端输入一个整数，用来表示年份，输出这一年是平年还是闰年
    闰年：能被4整除且不能被100整除，或者能被400整除

#include <stdio.h>
int main(int argc, const char *argv[])
{
    int year;
    printf("请输入年份 > ");
    scanf("%d", &year);

    if (year <= 0)
    {
        printf("输入的年份不合理\n");
        return -1;
    }

    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;
}

```

```
练习题3：
    在终端输入三个整数，分别表示三角形的三边长
    输出能否构成三角形
    如果能构成三角形，再输出能构成什么类型的三角形：等腰、等边、直角、普通。

    使用if嵌套。

#include <stdio.h>
int main(int argc, const char *argv[])
{
    int a, b, c;
    puts("请输入三角形的三个边长(a,b,c) > ");
    scanf("%d%d%d", &a, &b, &c);

    if (a <= 0 || b <= 0 || c <= 0)
    {
        puts("输入的边长至少有一个小于等于0");
        return -1;
    }

    if (a + b > c && a + c > b && b + c > a)
    {
        puts("可以构成三角形");
        if (a == b && b == c /* && a == c */ )
        {
            puts("等边三角形");
```

```
27          }
28          else if (a == b || b == c || a
    == c)
29          {
30              puts("等腰三角形");
31          }
32          else if (a*a + b*b == c*c ||
    a*a + c*c == b*b || b*b + c*c == a*a)
33          {
34              puts("直角三角形");
35          }
36          else
37          {
38              puts("普通三角形");
39
40          }
41      }
42      else
43      {
44          puts("不可以构成三角形");
45      }
46      return 0;
47  }
48
```

```
1  下周授课内容
2      分支语句
3      循环语句
4      数组
5      函数
```