一、数据类型

基本类型：int、char、int、float、double

构造类型：数组(一维数组、二维数组、指针数组、函数指针数组、结构体数组……)、结构体、共用体

指针：一维指针、二维指针、数组指针、函数指针、结构体指针

空类型：void，该类型也叫缺省型，用于描述值为空集，主要用于说明不返回值的函数或指向任一类型的指针等

1、32位计算机，各种数据类型所占字节数是：

char = 1；short = 2；int = 4；long = 4；float = 4；double = 8(实型数据的存储方式不同于其他基本数据类型)

2、bool类型

bool类型数据的值，是真或者非真，-即：0/1(非0即为真)

定义时需加上：#include<stdbool.h>

用的时候不需要专门声明。

3、char型数据

字符型数据，就是占用1个字节，8位，赋值范围要注意

unsigned char　　(无符号)

signed char　　(有符号)

如果给char型数据赋值超过范围，则会溢出。溢出原则，“保留低位，舍弃高位”

4、字符常量

char a = 'a'和char a = 97等价，字符常量'a'的ASCII码是97

5、字符串常量

定义：用双引号括起来的字符，就是字符串常量。

char *p = “hello”；

“helllo” “你好吗，世界！”

每个字符串常量后面都会有一个“******\0”

\0　用来表示字符串结束

6、define与typedef(注意两者格式的区别)

typedef是关键字，为已有类型取别名(如typedef unsigned int unit)

define是宏定义，仅仅进行简单的文本替换(#define TYPE 100)

举例：#define unit_p int*

unit_p a,b表示的是int *a,b;(b为int型)

typedef int * unit_p

unit_p a,b;表示的是int *a;int *b;

```
#include <stdio.h>
#define N 3.0e-23
#define M 950
int main(int argc, const char *argv[])
{
    double a;
    scanf("%lf",&a);
    printf("%.2e\n",a * M / N);
    return 0;
}
```

7、sizeof(关键字)

这个是计算数据占用字节数的运算符，只针对数据类型，不针对变量。 返回值：是括号内数据占用的字节数。

比如：int a = 5；则sizeof(a)和sizeof(int) = 4；sizeof(short int)=2；32位系统中，指针是占4个字节

利用指针判断所使用系统是多少位的系统

int *p = NULL;printf("%d\n",sizeof(p));32位系统中，指针是占4个字节；64位系统中，指针是占8个字节

8、带参宏定义define

对于带参宏定义不仅应在参数两侧加括号，还应在整个符号串外加括号，才能保证大部分情况下不出错。而有些时候就算已经这样做了，还是不能得到我们想要的结果。

```
#include <stdio.h>
#define SQ(y) ((y)*(y))
```

```
int main()
{
	int i = 1;
	while(i <= 5){
		printf("%d ",SQ(i ++));
	}
	return 0;
}
```

结果：1 9 25

```
#include <stdio.h>
int SQ(int y)
{
	return (y * y);
}
int main()
{
	int i = 1;
	while(i <= 5){
		printf("%d ",SQ(i ++));
	}
	return 0;
}
```

结果：1 4 9 16 25

以上两段程序说明了带参宏定义和函数之间是有很大差别的

编译器是在程序的预编译阶段对宏进行处理的，主要是用预编译指令来替换源文件中的宏。

(1)宏会在编译器在对源代码进行编译的时候进行简单替换，不会进行任何逻辑检测，即简单代码复制而已。

(2)宏进行定义时不会考虑参数的类型。

(3)参数宏的使用会使具有同一作用的代码块在目标文件中存在多个副本，即会增长目标文件的大小。

(4)参数宏的运行速度会比函数快，因为不需要参数压栈/出栈操作。

(5)参数宏在定义时要多加小心，多加括号。

(6)函数只在目标文件中存在一处，比较节省程序空间。

(7)函数的调用会牵扯到参数的传递，压栈/出栈操作，速度相对较慢。

(8)函数的参数存在传值和传地址(指针)的问题，参数宏不存在。

□ 带参的宏与函数区别

	带参宏	函数
处理时间	编译时	程序运行时
参数类型	无类型问题	定义实参,形参类型
处理过程	不分配内存 简单的字符置换	分配内存 先求实参值,再代入形参
程序长度	变长	不变
运行速度	不占运行时间	调用和返回占时间

二、存储类型

存储类型：auto、register、static、extern

全局变量：存放在静态区

局部变量：静态区、栈区、堆区

1、局部变量

关键字：auto，它的作用域：就是它所在函数的花括号{}内部。

定义：定义在函数内部，随着函数的执行而分配栈区，随着函数的调用产生，随着函数的退出而消失。

当一个自动变量没有被初始化，它的值是默认的随机值。

局部变量定义在一个函数内部，那么它只在这个函数内部有效。

2、全局变量

不能用auto来修饰，不是自动变量

定义：定义在函数外部，一般是定义在文件头的位置。

如果初始化了全局变量，那么它会被分配到data段，data段的数据会被原样保存到磁盘中。

```
#include<stdio.h>
int s;//全局变量未初始化，会被分配到BSS段，该段在程序加载初期会被自动清零。所以没初始化前，它的默认值是0。
int sd = 1024;//如果初始化了全局变量，那么它会被分配到data段，data段的数据会被原样保存到磁盘中。
//全局变量在程序加载时就被初始化
//全局变量初始化后在内存中分配了固定的地址；
int main()
{
    int a = 1; //局部变量在程序调用的时候才会被初始化；
                //局部变量调用时，被分配地址是随着每次调用、释放变化的。
    return (0);
 }
```

3、变量作用域

局部变量的名称作用域在一个花括号之间

作用域小的名称，将覆盖掉作用域大的名称

一个工程里不能有两个重名的全局变量

4、static静态存储类型

用static声明一个静态变量，那么这个静态变量和全局变量同等对待，放到静态区，在程序被加载时被初始化，之后不再执行初始化；静态局部变量和全局变量，只是作用域和名称不一样，其它的属性都一样(参照上一文档的图)

5、用static声明一个全局变量的用法

extern声明引用一个外部变量。比如：两个程序文件，a文件可以通过extern声明引用b文件里面的变量，在编译的时候要将两个程序文件同时编译。

static int a=0；用static修饰一个全局变量，那么这个全局变量的作用域仅限于本文件；修饰一个函数的话，效果相同。

总结：用static修饰一个局部变量，那么这个局部变量的作用相当于全局变量；

用static修饰一个全局变量，那么这个全局变量仅限于它所在文件使用，而不能通过extern声明而被另外的文件使用；

用static修饰一个函数，那么这个函数的作用域仅限于它所在的文件，而不能通过extern声明而被另外的文件使用；

extern的作用：可以声明函数，也可以声明变量。它的作用是，通过extern声明后，即使本文件中没有该变量和文件，那么也可以通过声明从别的文件中进行引用或者调用。

6、大端序和小端序

小端序：低地址存低字节；大端序：低地址存高字节

int a = 0X12345678(78为低字节)

char *q = (char *)&a;q相较于q+1为低地址

可以打印比较q和q+1的值，判断所使用系统是大端序还是小端序