1969年贝尔实验室-美国
的ken和dmr发明了C和Unix
信任程序员
C语言是编写操作系统的不二之选
C语言是目前运行速度最快的语言
C语言最适合被用来表达算法

需求分析
设计阶段
编写程序
调试程序

关键字表

auto ：声明自动变量　short ：声明短整型变量或函数　int：声明整型变量或函数 long ：声明长整型变量或函数　float：声明浮点型变量或函数　double ：声明双精度变量或函数　char ：声明字符型变量或函数　struct：声明结构体变量或函数　union：声明共用数据类型 enum ：声明枚举类型　typedef：用以给数据类型取别名　const ：声明只读变量　unsigned：声明无符号类型变量或函数　signed：声明有符号类型变量或函数　extern：声明变量是在其他文件正声明　register：声明寄存器变量　static ：声明静态变量　volatile：说明变量在程序执行中可被隐含地改变　void ：声明函数无返回值或无参数，声明无类型指针

if:条件语句　else ：条件语句否定分支（与 if 连用）　switch ：用于开关语句　case：开关语句分支　for：一种循环语句　do ：循环语句的循环体　while ：循环语句的循环条件　goto：无条件跳转语句

continue：结束当前循环，开始下一轮循环　break：跳出当前循环

default：开关语句中的“其他”分支　sizeof：计算数据类型长度

return ：子程序返回语句（可以带参数，也可不带参数）循环条件

常量和变量

简单的屏幕输出

各种各样的数据类型和数据格式

以及各种类型及其的打印输出的格式

实数只是一个实数的近似数，并不是真正的实数，而是无限接近的数

主函数，函数，语句，代码块

变量名称的命名规则

注释的两种写法，及其的不可嵌套的性质，说的是注释，不是行注释

计算变量或者数据类型所占的内存空间的大小

内存　　掉电即失　　线性地址表　　按照字节进行编地址

一个字节 = 8个二进制位

sizeof是一个关键字，不是函数名

1.sizeof在计算变量所占空间大小时括号可以省略；sizeof在计算数据类型的大小时不能省略括号（sizeof(类型)）。

关键字，编译期确定结果

3.sizeof后面为什么一直有括号

sizeof struct student；这一句话将会，编译失败，因为 sizeof 将 struct 作为操作数，而并非将整个 struct student 作为操作数，

sizeof a是正确的

sizeof int却是不正确的

```
#include <stdio.h>
#include <stdlib.h>
int main()
{
        int a =0;
    int b,c,d;
    b =2;
    c = sizeof int;sizeof在计算数据类型的大小时不能省略括号（sizeof(类型)）。
    d = sizeof a;
    printf("%d,%d,%d,%d",a,b,c,d);

        system("pause");
        return 0;
```

}

变量的赋值和赋值运算符

赋值运算符，用于赋值呀

赋值表达式 = 赋值运算符+两侧的操作数

只能将右边的数赋值给左边的数

考虑优先级和结合性（左结合还是右结合），**其中圆括号的优先性最高**

从右向左依次赋值属于《多重赋值》，多重赋值在声明的时候是不允许进行的

报错，在声明阶段进行了多重赋值，报错，必然。

```
#include <stdio.h>
#include <stdlib.h>
int main()
{
        int a =0;
    int b = c = d = 2;
    printf("%d,%d,%d,%d",a,b,c,d)

        system("pause");
        return 0;
}
```

成功，正确的多重赋值

```
#include <stdio.h>
#include <stdlib.h>
int main()
{
        int a =0;
    int b,c,d;
    b = c = d = 2;
    printf("%d,%d,%d,%d",a,b,c,d);
        system("pause");
        return 0;
```

}

本章拓展内容：

有符号整数和无符号整数：

有符号的整数：C语言将其解释为符号位
无符号的整数：C语言将其解释为数据位

负数在计算机中存储的方式是补码；
反码：求反；
补码：求反，加一
正数的补码，反码都是相同的。

计算机中将负数存为补码的原因？
方便将减法转换为加法运算来处理。
舍掉超出的位数

实型数据在内存中的存储格式：

定点小数是指小数点的位置是固定的，小数点位于符号位和第一个数值位之间，表示一个纯小数

拓展材料：关键字sizeof及其十大特性

Sizeof是c/c++中的关键字，它是一个运算符，其作用是取得一个对象（数据类型或数据对象）的长度（即占用内存的大小，以byte为单位）。其中类型包括基本数据类型（不包括void）、用户自定义类型（结构体、类）、函数类型。数据对象是指用前面提到的类型定义的普通变量和指针变量（包含void指针）。
十大特性：
特性0:sizeof是运算符，不是函数；
特性1:sizeof不能求void类型的长度；
特性2:sizeof能求void类型的指针的长度；

其实指针也是变量，不过这个变量很特殊，存放其它变量地址的变量，目前32位计算机平台下的指针变量的大小为4。

特性3:sizeof能求得静态分配内存的数组的长度；

注意：用sizeof求字符串的长度时要加上末尾的‘/0’。

例：

```
void fun(int array[10])
{
	int n = sizeof(array);
}
```

编辑程序：

```
#include<iostream>
using namespace std;

int fun(int array[10])
{
	int n = sizeof(array);
	return n;
}
int main()
{
	int ret;
	int array[10] = { 0 };
	ret = fun(array);
	cout << ret << endl;
	system("pause");
	return 0;
}
```

运行结果：

```
4
请按任意键继续. . .
```

在fun类n的值为4，而不是40，这是因为在函数参数传递时，数组被转化成指针了，假如直接传递整个数组的话，那么必然涉及到数组元素的拷贝（实参到形参的拷贝），当数组非常大时，会导致函数执行效率极低，而只传递数组的地址（即指针），只需要拷贝4byte。

特性4:sizeof不能求动态分配的内存的大小；

特性5:sizeof不能对不完整的数组求长度，否则会编译出错；

特性6:当表达式作为sizeof的操作数时，它返回的是表达式计算结果的类型大小，但是它不对表达式求值；

例：程序:

```
#include<iostream>
using namespace std;
int main()
{
        char ch = 1;
        int num = 1;
        int n1 = sizeof(ch + num);
        int n2 = sizeof(ch = ch + num);
        cout << "n1=" << n1 << endl;
        cout << "n2=" << n2 << endl;
        system("pause");
        return 0;
}
```

结果：

```
n1=4
n2=1
请按任意键继续. . .
```

由于默认类型转换的原因，表达式ch + num的计算结果类型为int，因此n1=4；而ch=ch + num的计算结果为char，虽然在计算ch + num时结果为int，当把结果赋给ch时又进行了类型转换，因此表达式的最终类型还是char，故n2=1。

特性7:sizeof可以对函数调用求大小，并且求得的大小等于返回类型（函数类型）的大小，但是不执行函数体；

特性8:sizeof求得的结构体（及其对象）的大小并不等于各个数据成员对象的大小之和；

规则：

a.结构体的大小等于结构体内最大成员大小的整数倍；

b.结构体内的成员的首地址相对于结构体的首地址的偏移量是其类型的整数倍，比如说double型成员相对于结构体的首地址的地址偏移量应该是8的倍数；

c.为了满足规则1和2，编译器会在结构体成员之后进行字节填充；

例：程序：

```
#include<iostream>
using namespace std;
int main()
{
```

```
        struct A
        {
                int num1 = 0;
                int num2 = 0;
                double num3 = 0;
        };

        struct B
        {
                int n1 = 0;
                double n2 = 0;
                int n3 = 0;
        };
        cout << "A=" << sizeof(A) << endl;
        cout << "B=" << sizeof(B) << endl;
        system("pause");
        return 0;
}
```

结果：

A=16

B=24

请按任意键继续. . .

sizeof(A)：4+4+8=16

sizeof(B)：4（n1占用地址空间：0,1,2,3）+4（不够8的倍数填充4个地址空间，4,5,6,7）+8（n2占用地址空间：8-15）+4（n3占用地址空间：16-19）+4（不够8的倍数填充4个,20,21,22,23）=24。

特性9:sizeof不能用于求结构体的位域成员的大小，但是可以求得包含位域成员的结构体的大小