C语言概述

1. **为什么学习c语言？**

C语言的发展和起源

C的特点：优点：代码量小 速度快 功能强大 缺点：危险性高 可开发周期长 可移植性不强

C的应用领域

C的重要性

1. **怎样学习c语言？**

途径：多思考 多上机

目标：能看懂程序，能调试程序，自学能力要强

老师上课犯得错误要记下来 自己犯得错误也要记下来。

资料：谭浩强《c语言程序设计》清华

1. **学习目标**

熟练掌握c语言的语法规则

掌握简单的算法

为以后学习c++,数据结构以及c#，java打下基础。

**4**．**Visual的使用**

后面有\*号就是新修改的没有保存的

有错误按f4

Close没有整个关闭

要点击close workspace 才是彻底关闭，或者要运行第二个程序的时候。

以cpp结尾是最重要的，其他的都是中间文件，没有用。

实数默认double类型

C语言编程预备知识

1. **cpu 内存条 硬盘 显卡 主板 显示器 之间的关系**

先把数据调用到内存条里面去，然后cpu在调用。将某些数据变成图像和声音，把数据发送给声卡， 主板提供一个传输的设备，让他们有效的组织在一起。

1. **helloworld是如何运行的**

编译 链接两个步骤生成.exe后缀名的程序，感叹号点了之后，是软件请求操作系统去执行这个文件，操作系统调用cpu，再在显示器上显示，都是靠cpu来运行的。

软件是不可以直接访问链接的，要借助系统。

1. **什么是数据类型**

先把数据保存到计算机里面，编程的第一步是数据存储，存储之前先分类。

基本类型数据：

整数（整形int4个字节，短整型short int2个字节，长整形long int8个字节），

浮点数（单精度浮点数float—存储范围比较小占的字节少4个，

双精度浮点数double存储范围大占的字节多8个，含有小数位的），

字符char一个字节，

复合类型数据（基本类型拼凑在一起）

结构体

枚举重点

共用体（已经被淘汰）

1. **什么是变量**

I=3， 3最终被存储在内存中，程序终止之后所占的空间被释放掉，运行时请求操作系统，操作系统给你分配内存，那个地址鱼i有关系了。

变量的本质是内存中的一段存储空间。

1. **Cpu 内存条 vc6++ 操作系统之间的关系**

这个软件请求操作系统，操作系统调用cpu给他在内存上分配位置。

1. **变量为什么必须初始化**

初始化就是赋值的意思。看11

1. **如何定义变量**

数据类型 变量名=要赋的值 等价于 数据类型 变量名 ；变量名=要赋的值。

Int i=3 等价于int i，i= 3

Int I ，j 等价于 int I ， int j

Int i，j=3等价于int I ；int j；j=3

1. **什么是进制**

十进制逢十进一

1. **常量在c语言中是如何定义的**

整数

十进制 ：传统的写法

十六进制：前面加0x

八进制：前面加数字0

浮点数

传统的写法

Float=3.22

科学记数法

Float=3.22e3 其实是3200 还有e-2

字符

单个字符用单引号括起来

字符串用双引号括起来。

“a”正确 因为里面默认添加了\0

1. **不同类型数据之间的相互转换**

Int i=45

Long j=102435

I=j

这样会丢失数据

后来看看补码

1. **什么是字节**

字节就是存储数据的单位，并且是硬件或者cpu所能访问的最小单位。

1.5g 1.5\*1024\*1024\*1024\*8 个0或者1

1. **代码规范化**

书：高质量；注释多一些

为了让自己让别人看的更懂，有利于后来重新看一下；使程序更不容易出错

1. **常量是以什么样的代码存储在计算机中的**

就是编码的问题

整数是以补码的形式转换为二进制的形式存储在计算机中的

实数是以IEEE754标准转换为二进制代码存储在计算机中的

字符的本质与整数相同，asc码转换为整数，然后再看整数用哪个二进制代码表示的。

1. **什么是ascii码**

双引号里面默认添加 \0

定义只能定义一次。复制可以多次。详见字符代码

ASCII不是一个值。而是一种规定，规定了不同的字符使用那个整数值去表示。

‘A’=65 ‘B’=66 ‘a’=97 ‘b’=98 ‘0’=48

1. **软件运行与内存管理的关系:**

1）软件在运行前，需要向操作系统申请空间，在内存空间充足时，操作系统将分配一段内存空间并将外存中的软件拷贝一份存入该内存空间中，并启动该软件的运行。

2）在软件运行期间，该软件所占空间不再分配给其他软件。

3）当软件运行完毕之后，操作系统将回收该内存空间（注意：操作系统并不清空该没存空间中遗留下来的数据，）以便再给其他软件使用。

综上所述，一个软件分配的空间中可能存着以前其他软件使用过的残留数据，这些数据被称之为垃圾数据，所以，我们为一个数组，一个变量，分配好存储空间知乎都要对其初始化。

基本的输入输出函数的用法

**1.Printf（）将变量的内容输出到显示器上**

四种用法

1. printf（“字符串/n”）
2. printf（“输出控制符”，输出参数）
3. print（“输出控制符1输出控制符2.。”,输出参数1，输出参数2）
4. printf（“输出控制符 非输出控制符“，输出参数）

**2.输出控制符**

%d int

%ld long int

%c char

%f float

% lf double

%x（或者%X%#x）是十进制int 或者long int或者 short int

%o 同上

%s 字符串

**3.非输出控制符（本身是什么就输出什么）**

\n i=什么什么的

**4.为甚么需要输出控制符**

01组成的代码可以表示数据也可以表示指令

如果01组成的代码表示的是数据的话，那么同样的01代码组合以不同的输出格式输出就会有不同的输出结果。

**5.scanf（）输入 ，通过键盘将数据输入到变量之中**

两种用法

用法一：scanf（“输入控制符“，输入参数）

功能： 将从键盘输入的字符转化为输入控制符所规定格式的数据，然后存入以输入参数的值为地址的变量之中。

用法二：scanf（“非输入控制符 输入控制符“。输入参数）

功能： 将从键盘输入的字符转换为输入控制符所规定格式的数据，然后存入以输入参数的值为地址的变量之中

注意：输入的时候要输入 非输入控制符，要不然会报错。

1)使用scanf之前最好先使用printf提示用户以什么样的方式来输入

2)Scanf中尽量不要使用非输入控制符，尤其不要用\n控制符

3）应该编写代码对用户的非法输入做适当的处理

While （（ch=getchar()）!=’\n’）

continue

运算符

**1.算术运算符**

+ - \* /（除）%（取余）

**2.关系运算符**

> >= < <= !=（不等于） ==（等于）

**3.逻辑运算符**

！（非） &&（并且） ||（或）

**4.赋值运算符**

= += \*= /= -=

**5.优先级别**

算术>关系>逻辑>赋值

**6.附录一些琐碎的知识**

还有自增自减 三目运算符 逗号表达式

**7.注意：除 与 取余**

1）.

除法的运算结果和运算对象的数据类型有关，两个数都是整数，则商就是整数，如果商有小数，则去掉小数部分；被除数和除数只要有一个或者两个都是浮点数，则商也是浮点数。

例如： 16/5==3 16/5.0==3.2000 -13/4==-4

-13/-3==4 3/5==0 5/3==1

最典型的例题是求s=1+1/2+1/3+1/4+1/5.。。。

2）.

取余%的对象必须是整数，结果是整除后的余数。其余数的符号与被除数相同

例如：13%3==1 13%-3==1 -13%3==-1

-13523==-13 3%5==3

**8.逻辑运算符**

！（） &&（并且） ||（或）

！真 假

！假 真

真&&真 真

真&&假 假

假&&真 假

假&&假 假

&&左边的表达式为假，右边的表达式不会执行

不含有分号的是表达式，含有分号的是语句

真||真 真

真||假 真

假||真 真

假||假 假

左边的表达式为真时，右边的表达式不会执行。

C语言对真假的处理

非零就是真

零就是假

真是用1表示

假是用0表示

流程控制（很重要）

**1.什么是流程控制？**

程序代码执行的顺序

**2.流程控制的分类**

**1）顺序**

**2）选择**

定义：某些代码可能执行，也可能不执行，有选择地执行某些代码。

分类：（If 此下问题参考代码）。

1. if最简单用法
2. if的范围问题
3. if else的用法
4. if else if else的用法
5. c语言对真假的处理
6. if举例—求分数的等级
7. if的常见问题解析
8. else后面有表达式 有分号

if（表达式1）

A

Else if（表达式2）

B

Else if （表达式3）

C

Else （表达式4）；

D

这样写语法不会出错，但是逻辑上有问题

Else（表达式4）；

等价于

Else

（表达式4）；//是一个无实际意义的语句。

D;

1. else后面有表达式的

if（表达式1）

A

Else if（表达式2）

B

Else if （表达式3）

C

Else （表达式4）

D

这样写是不对的，正确的写法是：

要么去掉表达式4

要么在else后面加if

1. 没有else结尾的

if（表达式1）

A

Else if（表达式2）

B

Else if （表达式3）

C

没有语法上面的错误，但是逻辑上不好。

1. if（表达式1）

A

Else（表达式2）

B

当表达式1和2都成立的时候,只执行A.

1. 加分号的问题

If （表达式1）//注意这里不可以加分号 加了分号之后变成一个语句 然后下面的else就会出错。

A；

Else

B;

1. 空语句的问题

If （3>2）;

等价于

If （3>2）

;//这是一个空语句

**3）循环**

定义：某些代码会被重复执行。

分类：

**for**

1. 格式

For （1；2；3）

语句A

流程12A3

1. 执行的流程（重点）

单个for的使用

多个for循环的嵌套使用

For（1；2；3）

For（4；5；6）

A

一个语句

B

流程1245A65

为什么6完了执行5？

如果6完了执行3 意味着下面这个语句执行完毕，则是5不成立的时候，所以6完了执行5.内部循环结束在执行外部循环。3完了执行2.2成立又执行4，又执行下一个循环。执行到B 时程序就结束。

**while**

1. 执行的顺序

格式

While（表达式）

语句；

1. 与for的相互比较

For可以和他相互交换

一般用for，他的逻辑性更强

1. 举例：

回文数

斐波那契数

1. While什么时候使用，什么时候使用for。

没法儿说，用多了自然会知道。

**Do.. while**

格式：

Do{

……..

}while （表达式）；

是先执行在判断。

主要用于人机交互，dowhile不等价于for和while，因为他一定会执行一次。

例子：一元二次方程。

**3.学习c语言为甚麽很重要？**

有助于培养自学能力（百分百掌握之后）

有助于学习c++ java，因为思想完全是一模一样的。

**4.如何看懂一个程序？**

（对于一个问题想了半天也不会就看答案，然后看懂他写的程序。

看懂了照着敲或者改一下或者看的特别懂再敲，也有可能是错的。

敲过一遍，出错加深对这个程序的理解。

多想想为什么是这样啊 或者为什么有个括号 为什么有个那个。）

**5.看懂程序的三步**

1. 流程

先执行哪一个后执行哪一个

1. 每个语句的功能
2. 试数

就是各种试一试

实验那个语句。

**6.对一些小算法的程序**

尝试自己去编程解决它

如果解决不了就看答案

关键是把答案看懂（这个要花很大的精力，也是我们学习的重点）

看懂之后，尝试自己去修改程序，并且知道修改之后的不同输出结果的含义

照着答案去敲

调试错误

不看答案，自己去敲。

如果程序实在无法理解，就把他背会。

**7.一些小算法**

判断一个数字是否是素数

判断一个数字是否是回文数

求一个十进制数字的二进制形式

求一个数字的每一位是奇数的数字并且取出来的新数字。

求一个数字倒过来的数字。

**8.强制转换**

格式：

（数据类型）（表达式）

功能：

把表达式的值强制转换为前面做执行的数据类型

例子：

（int）（4.5+2.3）结果是6

（float）（5）结果是5.0000

**9.为什么出现自增？**

代码更加精炼，自增的速度更快。

要明白的几个问题

编程要尽量屏蔽掉前自增和后自增的区别，i++和++i单独成一个语句，不要把它作为一个完整复合语句的一部分使用，就是不用和其他式子连在一起。Int m = i++ + ++I 这样写就是不规范的。

**10.自增自减**

**分类：**

前自增 --++i

后自增 --i++

**前自增和后自增的异同：**

相同点：最终都是i的值加一

不同点：前自增整体表达式的值是i加一之后的值；后自增整体表达式的值i加一之前的值。

**11.浮点数的存储带来的问题**

Float和double不能精确的保存一个小数，好像是因为补码有什么数据位什么的。

例子

Float i=99.9；

Printf（“%f\n”,i）;

最终在vc6++中的结果为：99.900002

**12.三目运算符**

A?B:C

等价于

If（A）

B;

Else

C;

**13.逗号表达式**

格式：

（A,B,C,D）

功能：

从左到右执行，最终表达式是D的值。

**14.switch语句**

第一：所有case后面的常量表达式为便于描述我们姑且称之为标签，这些标签都只能是（1）枚举常量（2）数值常量（3）字符常量（4）常变量（5）宏名中的一种。注意普通变量，枚举变量是不能作为标签使用的。

第二：Switch后面括号里的表达式允许是任何类型

但是vc++中只允许为int或者char类型。

第三：执行完一个case语句之后，流程控制就转移到下一个case字句继续执行，case常量表达式只是起语句标号的作用，并不是在该处进行条件判断，在执行switch语句时，根据switch中表达式的值找到与之匹配的case语句，就从此case句开始执行下去，不再进行判断。

第四：switch是选择不是循环，但是其中可以出现break语句。但是不能出现continue。

**15.break的用法**

break如果用于循环是用来终止循环

break如果用来switch，也是用来终止。

break不能用于if，除非if是循环内部的一个语句。

在多层循环中，break终止距离她近的for

在switch多层嵌套中

**16.continue的用法。（小break）**

功能：用于跳过本次循环余下的语句，转去判断是否需要执行下次循环。

例子1：

For （1；2；3）

{

A;

B;

Continue；//如果执行该语句，则执行完该语句之后，会执行语句3，cd不会执行。

C;

D;

}

例子2：

While（表达式）

{

A;

B;

Continue；//如果执行该语句，则执行完该语句之后，会执行表达式，cd不会执行。

C;

D;

数组

**1.为什么需要数组？**

为了解决大量同类型数据的存储和使用问题。

为了模拟现实世界。

**2.数组的分类**

**一维数组（int a【25】直线）**

1）定义：为n个变量连续分配存储空间；所有的变量数据类型必须相同；所有变量所占的字节大小必须相等。

2）有关一维数组的操作：

初始化：

完全初始化

Int a 【5】 = {1,2,3,4,5}；

不完全初始化，未被初始化的元素为0.

Int a【5】 = {1,2,3}

不初始化：

所有元素是垃圾值。

清零：

Int a 【5】= {0}；

错误的写法：

Int a 【5】；

a【5】= {1,2,3,4,5}；

只有在定义数组的是同时才可以整体赋值，其他情况下整体赋值是错误的。

时刻记住那是**长度**。

Int a 【5】= {1,2,3,4,5}

Int b【5】；

如果把a数组中的值全部复制给b

错误的写法：

B=a 因为一维数组名不代表数组中的所有的元素；一维数组名代表数组中第一个元素的地址。

正确的写法：

For （i=0；i<5；++i）

B【i】=a【i】；

赋值，排序，求最大值最小值，倒置，查找，插入，删除

**二维数组int a 【8】【6】平面**

1）Inta【3】【4】

总共是12个元素，可以当做3行4列看待，这12个元素的名字依次是 A[0][0] a[0][1]还有a[1][0] 一直到a[2][3]

A[i][j]表示第i+1行第j+1列的元素

2）Int a[m][n]；最后一个就是a[m-1][n-1]

初始化：

Inta[3][4] = {1,2,3,4,5,6,7,8,9,10,11,12}

Int a = {

{1,2,3,4}

{5,6,7,8}

{9,10,11,12}

}

3）输出二维数组内容：像平常那样输出。

4）其他操作：、

对二维数组排序；判断矩阵是否对称；求每一行的最大值；矩阵的相乘。

**三维数组int a 【3】【4】【5】立体的**

**多维数组：**

是否存在多维数组：不存在。因为内存是线性一维的，

N维数组，可以当做每个元素是n-1维数组的数组。

比如：

inta[3][4],该数组是含有三个元素的一维数组，只不过每个元素都可以在分成4个小元素。

Int[3][4][5]是含有3个元素的数组，只不过每个元素都是4行5列的二维数组。

函数《重点》

1. **函数**

c语言的基本单位。跟其他语言相比较得出的。

**2.为什么需要函数？**

避免了重复性操作，有利于程序的模块化，把大问题分解成小问题，因为每一个功能都可以用一个独立函数去实现。

**3.什么叫做函数？**

逻辑上：能够完成特定功能的独立的代码单元。

物理上：能够接受数据（也可以不接受）；能够对接收的数据进行处理；能够将数据处理的结果返回（当然也可以不返回任何值）。

总结： 函数是一个工具，是为了解决大量类似问题而设计的，可以当做一个黑匣子（内部处理数据）。

**4.如何定义函数？**

函数的返回值（void） 函数的名字 （函数的形参列表）

{

函数的执行体

}

**1）函数定义的本质**：是详细描述函数之所以能够实现某个特定功能的具体方法（就是怎么着详细写能实现呗）。

**2）Return表达式；的含义：**

1. 终止被调函数，像主调函数返回表达式的值
2. Break是用来终止循环和switch的，return是用来终止函数的。
3. 如果表达式为空，则只终止函数，不像主调函数返回任何值。

例子：

Void f()

{

Return;//这里不用加任何值，也不能加任何值，含义是终止函数不返回任何值。

}

Int f（）

{

Return 10；含义是终止函数并且像主调函数返回他的值10.

}

函数名字前面不是void就会返回一个值，返回的值就是括号里面的。

**3）函数返回值的类型也称为函数的类型**

因为如果 函数名前的返回值类型 和 函数执行体中的return表达式； 中表达式的类型不同的话，则最终函数返回值的类型 以函数名前的返回值类型。例子看函数的类型.cpp

**5.函数的分类**

有参函数 和 无参函数

有返回值函数 和 无返回值函数

库函数 和 用户自定函数

普通函数 和 主函数（main函数）

一个程序必须有且只能有一个主函数，程序是从主函数进入的。

主函数可以调用普通函数，普通函数不可以调用主函数。（因为main函数不能实现递归）

普通函数之间可以随便调用

主函数是程序的入口和出口。

值传递函数 和 地址传递函数

**6.函数调用和定义的顺序**

如果函数调用在前面，就是不知道调用的谁所以会出错。所以要加函数前置声明，

**7.函数前置声明的作用：**

1. 告诉编辑器即将可能出现的若干个字母代表的是一个函数
2. 告诉编译器即将可能出现的若干个字母所代表的函数的形参和返回值的具体情况。
3. 函数声明是一个语句，末尾必须加分=分号
4. 对库函数的声明是通过#include<库函数所在文件的名字.h>来实现的。<stdio.h>std是标准的意思，i是输入，o是输出，h是头文件。

**8.形参和实参**

这两个必须个数相同，位置一一对应，数据类型必须相互兼容。

**9.如何在软件开发中合理的设计函数来解决实际问题？**

一个函数的功能尽量独立单一，这样可重性高；多学习多模仿牛人的代码；

函数是c语言的基本单位。类是Java，c#,c++的基本单位。

**10.常用的系统函数**

Double sprt （double x）； 求得x的平方根

Int ads （int x）； 求得x的绝对值

Double fabs （double x）； 求得x的绝对值。

推荐书：turboc2.0实用大全

**11.专题**：

递归：看郝斌老师的数据结构视频。

变量的作用域和存储方式：

**1.按作用域分：**

全局变量：在所有函数外部定义的变量叫做全局变量；

适用范围：从定义位置开始到整个程序结束。

局部变量：在一个函数内部定义的变量或者函数的形参。

void f(int i)i虽然是形参。但是在f内部使用的。

{

int j =20;

}j也是在内部使用的。

适用范围：只能在本函数内部使用。

全局变量和局部变量命名冲突的关系：

全局变量和局部变量的名字一样的时候，局部变量会屏蔽全局变量，最终显示的结果是局部变量的值。（就好像手机流量先使用本地流量在使用全国流量）

**2.按变量的存储方式**

静态变量

自动变量

寄存器变量

指针（地址）

1. **优点重要性：**

表示一些复杂的数据结构，快速的传递数据减少了内存的耗用；是函数返回一个以上的值；能直接访问硬件；能够方便的处理字符串；是理解面向对象语言的基础。

**2.总结:**

是c语言的灵魂

**3.定义**

地址：内存单元的编号；从0开始的非负整数；范围：4G（0—4G-1）

指针：指针就是地址，地址就是指针；指针变量就是存放内存单元编号的变量，或者说指针变量就是存放地址的变量；指针和指针变量是两个不同的概念；指针的本质就是一个操作受限的非负整数。指针可以相减。不可以相加。指针是水，指针变量是杯；但是要注意：通常我们叙述时把指针变量称作指针。就是这个变量有好多种类型。

**4.分类**：

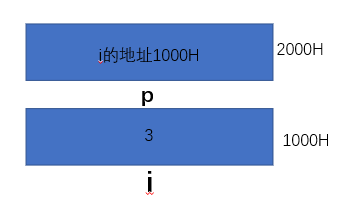
（1）**基本类型指针**

Int \*p;

I = 3;

\*p=I;

就是以下这种关系。



**5.\*的含义：**

1.乘法；

2.定义指针变量。Int\*p；//定义了一个名字叫做p的变量，int \*表示p只能存放int变量的地址。

3.指针运算符：该运算符放在已经定义好的指针变量的前边，取地址符的逆运算。

**6.如何通过被调函数修改主调函数普通变量的值**

实参必须为该普通变量的地址

形参必须为指针变量

在被调函数中通过

\*形参名 = ……

的方式就可以修改主调函数相关变量的值

**7.指针和数组**

**1）一维数组名：**

一维数组是个指针常量，他存放的是一维数组第一个元素的地址。

**2）下标和指针的关系：**

如果p是个指针变量，则p【i】永远等于\*（p+i）

**3）确定一个一维数组需要几个参数？**

（一个函数要对一个一维数组进行处理，则需要那些参数？）

数组第一个元素的地址，数组的长度。

**4）指针变量的运算。**

指针变量只能相减。如果两个指针变量指向的是同一块连续单元中的不同存储单元，则这个指针可以相减。

**5）一个指针变量到底占几个字节？**

预备知识：

Sizeof（数据类型）

功能：返回值就是该数据类型所占的字节数。

例子：

Sizerof(int) = 4 sizerof(char) =1

Sizerof(功能名)

功能：返回值是该变量的字节数。

**6）问题**

假设p指向char类型变量（1个字节），假设q指向int类型变量（4个字节），假设r指向double类型变量（8个字节，），p q r本身所占的字节是否一样？

回答：是一样的

总结：一个指针变量无论它指向的变量占几个字节，该指针变量只占四个字节；一个变量的字节是用该变量首字节的地址来表示。

**7）详见代码**

指针和二维数组（有点难）

指针和函数

**指针和结构体**

**8.多级指针**

Int I =10;

Int \*p = &I;

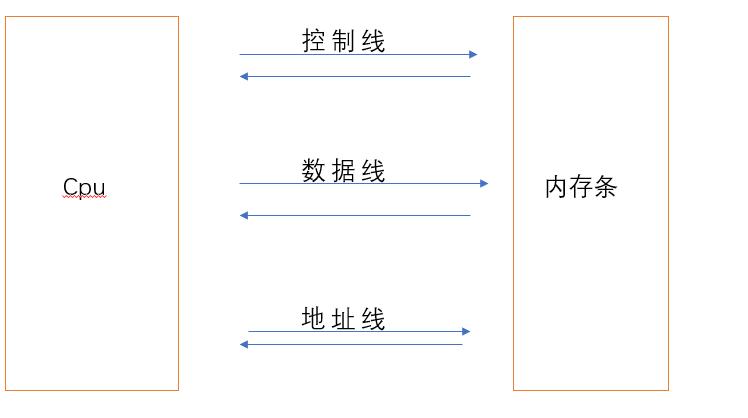
Int \*\*q = &p;

Int \*\*\*r = &q;

\*\*\*r = i

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 变量名字 | I | P | Q | R |
| 内容 | 10 | 1000H | 2000H | 3000H |
| 地址 | 1000H | 2000H | 3000H | 4000H |

**9.cpu和内存条的关系**



内存中的数据读入cpu，cpu处理完在vc6++中的结果为：写入内存，内存数据太多时将一次性写入硬盘。

控制线：控制数据传输的方向

数据线：传输数据

地址线：确定哪一个单元。一根地址线两种状态。如果有64根，就可以访问2的64次方的字节，也就是8gb。

动态内存分配（重要）

**1.传统数组（静态数组）的缺点：**

1）.数组长度必须事先确定，且只能是常整数，不能是变量。

例子：

Int a[5];//这样是对的 int len =5;int a[len]//这样是不对的。

2）.传统形式定义的数组，该数组的内存程序员无法手动释放；数组一旦定义，系统会为在一个函数运行期间， 该数组分配的存储空间就会一直存在，除非数组所在的函数运行结束。

3）.数组的长度不能在函数运行的过程中动态的扩充或者缩小。即数组的长度一旦定义，其长度就不能改变。

4）.A函数使用的数组，在运行的时候可以被其他函数使用，但是当 A运行结束的时候，就不能再被其他函数调用，而且传统方式定义的数组不能跨函数使用。

**2.为什么需要动态分配内存？**

动态数组很好的解决了传统数组的4个缺陷。

**3.动态内存分配举例——动态数组的构造**

Int a【5】//int占4个字节，本数组有20个字节，每四个字节被当做一个int变量使用。

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

黄色的代表a【0】然后是a【1】就是供a【1】来使用，后面依次来分配

Parr = （int \*）malloc（4\*len ）的图

Parr指向了前四个字节，\*parr代表那四个字节的变量

\*parr \*(parr+1) \*(parr+2) \*(parr+3) \*(parr+4)

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

蓝色是parr ↑parr+1 ↑parr+2 ↑parr+3 ↑parr+4

这里代表20个字节

**4.动态内存和静态内存的比较**

静态内存是由系统自动分配，由系统自动释放。

静态内存是在栈里面分配的

动态内存是由程序员手动分手动释放

动态内存是在堆里面分配的。

**5.跨函数使用内存的关系**

静态内存不可以跨函数使用的更准确的说法是：静态内存在函数执行期间可以被其他函数使用；静态内存在函数执行完毕之后就不能被其他函数使用了。

动态内存可以跨函数使用：动态内存在函数执行完毕之后仍然可以被其他函数使用。

结构体？

1. **为什么需要结构体？**

为了表示一些复杂的事物，而普通的基本类型无法满足需求

1. **什么叫做结构体？**

把一些基本类型数据组合在一起形成的一个新的复合数据类型。

1. **如何定义一个结构体？**

3种方式

1. **怎样使用结构体变量？**
   1. **赋值和初始化**

定义的同时可以整体赋值

如果定义完之后，则只能单个单个的赋初值。

* 1. **如何取出结构体变量中的每一个成员（重点）**

1）.结构体变量名.成员名

2）p结构体（指针变量名）->成员名，但是p结构体->成员名在计算机内会被转换成（\*p结构体）.成员名，\*p其实就是取地址的意思，所以pst->age 等价于 st.age 就是因为我们知道他的转换。

3）pst->age的含义

Pst所指向的那个结构体变量中的age这个成员，所以就是age这个成员。

**5.结构体变量和结构体指针变量作为参数传递的问题**

推荐使用结构体指针变量作为函数参数来传递。

**6**.**结构体变量的运算**

不能相加不能相减也不能相互乘除。

但结构体变量可以相互赋值，st1 = st2是可以的。

举例

动态构造存放学生信息的结构体数组

**7.枚举**

定义：把一个事物所有可能的取值一一列举出来

怎样使用枚举

枚举的优缺点：代码更安全但是书写麻烦

**8.补码**

原码：也叫符号绝对值码 0正1负，其余二进制是该数字的绝对值的二进制位；简单易懂，加减运算复杂；增加了cpu的复杂度；零的表示不唯一。

反码：反码运算不便，没有在计算机中应用。

移码：移码表示数值平移n位，n称为移码量；移码主要用于浮点数的阶码的存储。

补码：

十进制转二进制:

如果是正数与原码相同

如果是负数，先求原码，求反码符号位不变数据位取反，反码加1是补码。

不够位数时，左边补1，因为是负数。得看把那个数定义成了什么类型，如int shortint

例如：-5的二进制原码是1000 0101 反码是1111 1010 补码是 1111 1011 补码转换为十六进制是fb 但是这是8位前面要补24个1二进制1，转换为16进制就是6个f。所以是fffffffb

0:000000000

二进制求十进制：

先看第一位0正1负，正数的话原码反码补码相同；

负数的话，所有位取反，末尾加1，所得数字是绝对值。补码的补码是原码。

**9.八位二进制所代表的十进制数示意图**

|  |  |
| --- | --- |
| 二进制 | 十进制 |
| 0000 0000 | | 0 | |
| 0000 0001 | | 1 | |
| …. ….. | | … | |
| 0111 1111 | | 127 | |
| 1000 0000 | | -128 根据先取反再加一求出来绝对值求得 | |
| 1000 0001 | | -127 | |
| 1000 0010 | | -126 | |
| …. | | .. | |
| 1111 1111 | | -1 | |

**10.学习目标：**

在vc6++中一个int类型的变量中所能存储的数字的范围是多少？

Int类型变量所能存储的十六进制最大的正数是7FFFFFFF

Int类型变量所能存储的绝对值最大的负整数用十六进制表示是：8000 0000（二进制是1+31个0）

具体可以看上面的表格

最小负数的二进制代码：1000 0000

最大正数的二进制代码：0111 1111

已知一个正数的二进制代码求出原始的数字：

数字超过最大正数会怎么样？

会溢出，保留后面的

不同类型数据的转换。

**11.链表**

**12.算法：**

通俗定义：解题的步骤和方法

狭义定义：对存储数据的操作；对不同的存储结构，要完成某一个功能所执行的操作是不一样的；比如：要输出数组中所有元素的操作和要输出链表中所有元素的操作肯定是不一样的。这说明：算法是依附于存储结构，不同的存储结构，所执行的算法是不一样的。

广义定义：广义的算法也叫泛型；无论数据是如何存储的，对该数据的操作都一样。

**13.我们至少可以通过两种结构来存储数据。**

**数组**：是连续的

优点:存取速度快

缺点：需要一块很大的内存；插入和删除元素的效率很低，因为动一个后面的都需要动

**链表**：是不连续的

|  |  |
| --- | --- |
| 数据 | 下一个元素的位置 |

优点：插入删除元素效率高；不需要很大的内存。

缺点：查找某个元素的位置效率很低，因为要根据1来看2的位置，再根据2看3的位置

**专业术语**：头指针：存放头结点的地址的指针变量。

头结点：头结点的数据类型和首节点的类型是一模一样的；是首节点前面的那个节点；

头结点并不存放数据；设置头结点的数据是为了方便对链表的操作。

首节点：存放第一个有效数据的节点

尾节点：存放最后一个有序数据的节点

**14.示意图**

○‐○‐‐○‐‐○‐‐○

第一个是头结点，还有一个头指针指着他；第二个是首节点。第五个是尾节点的指针域是空的，数据域不空。

定义链表需要的条件：只有头指针一个参数。

**15.位运算符**

**&**

按位与；&&逻辑与（并且）这俩含义不同

1&1 = 1;1&0=0;0&1=0;0&0=0;就是将数据转换为补码进行运算

**|**

按位或 与逻辑或不同

**~**

按位取反；把其的所有二进制数取反。

**^**

按位异或 ：相同为零，不同为1。

1^0 = 1 1^1=0 0^1=1 0^0=0

**<<**

按位左移，左移n位相当于乘以2的n次方

I<<2;表示把所有的二进制数左移二位。

**>>**

按位右移，右移相当于除以2的n次方，注意不能溢出。

I>>3 表示把i的所有二进制位右移三位，左边一般是补0，也有可能补1。

**现实意义**：通过位运算符我们可以把对数据的操作精确到每一位