# C语言概述

## 为什么学习C语言

### C的起源和发展

第一代语言机器语言001,第二代语言汇编语言ADD AX,第三代高级语言结构化语言a+b,Fortan,Basic,C,Pascal,1980之后面向对象的语言Algo,Simula67,Ada,Smalltalk,C++,java ,C#.

结构化语言有缺陷.

C语言与UNIX操作系统,B语言,C语言,起源于贝尔实验室.C++,C#都源于C.

函数和指针重要.指针用于访问硬件.

C语言很成熟,1994 C99标准至今.

语言三大分支:

C/C++

Java

C#

C++软件可以完全运行C语言

### C的特点

优点:代码量小,速度快,功能强大

三大操作系统: Windows C C++

Unix

Linux

都是用C语言编写.操作系统的作用是什么,操作控制硬件和数据的工机.

缺点:危险性高,容易通过不容易调试,

开发周期长,

可移植性弱.

### C语言的应用领域

系统软件开发:操作系统,驱动程序,数据库

应用软件开发:图形图像多媒体,软件开发,游戏开发

### C的重要性

C C++ 数据结构 学习算法必须学习数据结构

其他程序语言的基础 <<编程箴言>>

病毒的编写

## 怎么学习C语言

途径:多思考,多上机,多问为什么 代码必须写规范,一点不能漏,格式要完全一样,完全规范

目标:能看懂程序,能调试程序,自学能力要很强

参考资料:谭浩强<<C语言程序设计>>

## 学习的目标

了解程序语言及发展历史

熟练掌握C语言的语法规则

掌握简单的算法

理解面向过程的思想,有助于面向对象的学习

看懂程序,调试程序,将大问题转化为一系列小问题的思想

为学习C++,数据结构,C#, java 打下基础

## 常见问题答疑

### 学习Java为什么建议先学C语言

Java的大部分语法知识继承了C语言

C是面向过程的代表有助于理解面向对象

C中的指针是理解Java的引用的基础

### 没学习过计算机专业课程能够学懂C语言

需要一些计算机的底层知识

### 英语和数学不好能学好C吗

32个关键字:

Auto break case char const continue default do double else enum extern float for goto if int long register return short signed sizeof static struct switch typedef unsigned union viod volatile while

数学 哲学,想方法就要建立数学模型,需要用到数学,先学会怎么用,在想怎么创造.

刚开始一周很难。

## 课程计划

23讲

基本编程知识

数据类型

运算符和表达式

流程控制

函数

数组

指针

变量的作用域和存储方式

扩展数据类型

专题:字符串的处理

进制转换

补码

动态内存分配

综合应用:链表的使用

## 5举例子:一元二次方程

软件的使用

解决问题的思路同样是编程的思路

假死解决办法:Ctrl Alt Delete,任务管理器

保存 Ctrl S,时刻记得保存,未保存时有\*显示

记得加空格,代码写规范

关闭程序时,要关闭Workspace. Close Workspace

.cpp,源文件重要,中间文件不重要

**C语言编程预备知识**

1. **CPU 内存条 硬盘 显卡 声卡 主板 显示器 之间的关系**

操作系统把需要调用的数据从硬盘调进内存条,然后CPU进行处理,传给显卡和声卡,最后传输给显示器.各个模块插在主板上,彼此才能接通

1. **HelloWorld程序是如何运行起来的**

通过编译和链接生成可执行程序.exe,然后调用CPU执行

操作系统控制硬件

1. **什么是数据类型**

**内存很重要**

对数据进行操作,先对数据进行分类

C语言数据类型两类

整数

整型 --int -- 4字节

短整型 --short int -- 2字节

长整型 --long int -- 8字节

浮点数[实数]

单精度浮点数 -- float -- 4字节

单精度浮点型 -- double -- 8字节

字符

单字符 -- char -- 1字节

复合数据类型

结构体

枚举

共用体

1. **什么是变量**

变量的本质是内存中一段存储空间,程序结束,所占空间被释放

1. **CPU 内存条 vc++6.0 操作系统 之间的关系**

Vc++6.0对操作系统发出请求,使其在内存条中分配空间

1. **变量为什么必须初始化**

所谓初始化就是赋值的意思,没初始化时,输出的值不变.不变因为vc++6.0自动添加添充字.因为被分配的内存空间可能被使用过,存储着垃圾数据.

软件运行与内存的关系: 软件运行前申请存储空间,之后操作系统分配内存空间,软件运行结束后,分配的内存空间被释放,但不会清空遗留下来的数据.

1. **如何定义变量**

**数据类型 变量名 = 要赋的值;**

**数据类型 变量名;**

**变量名 = 要赋的值;**

**举例子: 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;**

**int I = 3, j = 5; 等价于 int I; int j; I = 3; j = 5;**

**int I, j, I = j = 5; 等价于 int I; int j; I = 5; j = 5;**

1. **什么是进制**

**十进制就是逢十进一**

**二进制就是逢二进一**

**逢N进一**

**C语言规定八进制前要加0,十六进制前要加0x或0X**

**printf的用法**

**int %d char %c float %f double %lf long int %ld**

**%d表示以十进制输出**

**%x或者%X表示以十六进制输出**

**%o表示以八进制输出**

1. **常量在C语言中是如何表示的**

**整数**

**十进制: 传统的写法**

**十六进制: 前面加0x或者0X**

**八进制: 前面加0,注意不是字母o,O而是数字零**

**浮点数**

**传统的写法: float x = 3.2;**

**科学计数法: float x = 3.2e3; //x的值是3200**

**float x = 123.45e-2; //x的值是1.2345**

**字符**

**单个字符用单引号括起来**

**‘A’表示字符A**

**‘AB’ 错误**

**“AB” 正确**

**字符串用双引号括起来**

**“A” 正确, 因为”A”代表了’A’ ’\0’ 的组合**

1. **常量是以什么样的二进制代码存储在计算机中的**

**整数是以补码的形式转化为二进制代码存储在计算机中的**

**实数是以IEEE754标准转化为二进制代码存储在计算机中的**

**字符的本质实际也是与整数的存储方式相同,通过ANSI C 码转换**

**编码**

**int i = 86;**

**11. 代码规范化**

**林锐 <<高质量C/C++编程>>**

**注释占代码的1/3**

**好处: 整洁,不容易出错,可读性更高**

**代码成对的敲,if,while,for 后空一格,是为了与函数进行区分**

缩进,内部代码需要缩进

换行,功能相对独立,就需要换行

程序分为三部分:

// 定义变量

// 操作变量

// 输出

**12. 什么是字节**

**存储数据的单位，是硬件所能访问的最小单位**

**CPU只能处理内存内的数据，一个字节有八位**

**1 字节 = 8位 通过位运算符控制位**

**1k = 1024字节**

**1M = 1024 K**

**1G = 1024 M**

**13. 不同类型数据之间互相赋值问题**

**需要研究补码的知识**

**P54**

**int i = 45;**

**long j = 102345;**

**i = j;**

**printf (“%ld %d\n”, i, j);**

**float x = 6.6;**

**double y = 8.8;**

**printf (“%f %lf”, x, y);**

14. 什么是ASCII码

ASCII码不是一种值，是一个规定

ASCII码规定了不同的字符是使用哪个数值去表示，它规定了：

‘A’—— 65

‘B’ -- 66

‘a’ -- 97

‘b’ -- 98

‘o’ -- 48

和GB2312码、UTF-8码相似 · · · ·

15. 字符的存储【字符本质上与整数的存储方式相同】

基本的输入输出函数的用法

printf() – 将变量的内容输出到显示器上

四种用法

1. printf(“字符串”)；

2. printf(“输出控制符”，输出参数)；

3. printf(“输出控制符1 输出控制符2”，输出参数1, 输出参数符2)；

输出控制符和输出参数的个数必须匹配。

4. printf(“输出控制符 非输出控制符”， 输出参数);

输出控制符包含如下：

%d -- int

%ld -- long int

%f -- float

%c -- char

%lf -- double

%x %X %#x -- int 或 long int 或 short int

%o -- 同上

%s -- 字符串

为什么需要输出控制符

1. 01组成的代码可以表示数据也可以表示指令

2. 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

scanf() ----【通过键盘将数据输入到变量中】

两种方法：

用法一： sacnf(“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中。&i

&i 表示i的地址 &是一个取地址符

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中。&i

非输入控制符必须原样输入

如何使用scanf编写出高质量代码

加编程的时间和输出结果

2023-03-21 22:13:44

在VC++6.0中的输出结果是：

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 printf("请输入三个值，中间以逗号隔开：");

2. scanf中最好不要使用非控制符，尤其是\n

3. 应该编写代码对用户的非法输入做适当的处理【非重点】

while ( (ch=getchar()) != ‘\n’)

continue;

运算符

算术运算符

+ - \* /（除）两个整数相除还是整数 ，可以进行强制数据类型转换 %（取余数）（取余必须是整数）

关系运算符

> >= < <= != (不等于) ==（等于）

逻辑运算符

!(非) &&（并且） ||(或)

！真 是假

！假 是真

真&&真 真

真&&假 假

假&&真 假

假&&假 假

真||真 真

真||假 真

假||真 真

假||假 假

C语言对真假的处理

非零是真

零是假

真是1表示

假是0表示

&&左边的表达式为假 右边的表达式肯定不会执行

||左边的表达式为真 右边的表达式肯定不会执行

不含有；的是表达式， 含有；的是语句

赋值运算符

= += \*= /= -=

优先级别

算术 > 关系 > 逻辑 > 赋值 （可以加括号，进行提示）

附录的一些琐碎的运算符知识

自增 自减 三目运算符 逗号表达式

流程控制

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

顺序执行

选择执行

定义

某些代码可能执行，也可能不执行，有选择的执行某些代码

分类

if

1. if最简单的用法

if （表达式）

语句

功能：

如果表达式为真，执行语句

如果表达式为假，语句不执行

2. if的范围问题

1.

if (表达式)

语句A；

语句B；

解释：if默认的只能控制一个语句的执行或不执行

if只能控制语句A的执行或不执行

if无法控制语句B的执行或不执行

或者讲：语句B一定会执行

2.

if (表达式)

{

语句A;

语句B;

}

此时if可以控制语句A语句B

由此可见：**if默认的只能控制一个语句的执行或不执行**

**如果想控制多个语句的执行或不执行**

**就必须把这些语句用{}括起来 F4进行调试错误显示**

3. if…else…的用法

4. if…else if…else 的用法

格式：

if （1）

A；

else if （2）

B；

else if （3）

C；

else

D；

5. C语言对真假的处理

非零是真

零就是假

真用1表示

假用0表示

6. if举例—求分数的等级

互换两个数字

对任意三个数字进行排序

7. if的常见问题分析

1. 空语句的问题

if （3 > 2）;

等价于

if （3 > 2）

; //这是一个空语句

2. if （表达式1）

A；

else

B；

是正确的

if （表达式1）；

A;

else

B;

是错误的，运行报错

3. if （表达式1）

A;

else if （表达式2）

B;

else if (表达式3)

C;

else

D；

即便表达式1和2都成立，也只会执行A语句

4. if （表达式1）

A;

else if （表达式2）

B;

else if (表达式3)

C;

这样写语法不会出错，但逻辑上有漏洞

5. 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）//7行

D；

这样写是不对的，正确的写法是：

不能组成一个语句

要么去掉7行的表达式4，

要么在7行的else 后加上if

6. 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）；//7行 无实际意义的语句

D；

7行表达式为无实际意义的语句

else （表达式4）；//7行

D；

等价于

else （表达式4）；//7行

D；

switch

循环执行

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；

2. 执行的流程【重点】

单个for循环的使用

for （i=0；i>3; ++i ）

语句A；

多个for循环的嵌套使用

1. for （1； 2； 3） //1行

for（4； 5； 6） 2行

A； //3行

B； //4行

整体是两个语句，1 2 3 是第一个语句，

4是第二个语句

2. for （1； 2； 3）

for（4； 5； 6）

{

A；

B；

}

整体是一个语句

3. for (7; 8; 9)

for （1； 2； 3）

{

A；

B；

for（4； 5； 6）

{

A；

B；

}

}

整体是一个语句

3. 范围问题

4.举例

1 + 2 + 3 + 4 + … +100

1 + 1/2 + 1/3 + 1/4 + … +1/100

练习

求1到100之间的奇数之和

求1到100之间的奇数的个数

求1到100之间的奇数的平均值

求1到100之间的奇数之和，再求1到100之间的偶数之和

while

1. 执行流程

格式：

while （表达式）

语句；

2. 与for的相互比较

for 和while 可以相互转换

for （1； 2； 3）

A;

等价于

1；

while （2）

{

A;

3；

}

while和for可以相互转化

但for的逻辑性更强，更不容易出错，推荐多使用for

3. 举例

从键盘输入一个数字，如果该数字是回文数，

则返回yes,否则返回no

回文数：正着写和倒着写一样

如：121 12321

4. 什么时候使用for ，什么时候使用while

没法说，用多了自然而然就知道了（看程序看多了就知道了）

do…while

格式：

主要用于人机交互

do

{

} while(表达式)

do … while至少执行一次， 并不等价于for，当然也不等价于while

穿插在课堂中的零散知识笔记：

算法：**模仿**

小算法的程序

判断一个数字是否是素数

判断一个数字是否是回文数

编程实现求一个十进制数字的二进制形式

把一个数字的每位是奇数的数字取出来组合形成新的

数字

求一个数字倒过来的数字

算法

选一本好书，看懂程序，看懂之后敲完程序还是错的

出错加深理解

看懂程序之后，再接着改程序，看出现什么效果，后敲 程序

如何看懂一个程序，分三步：

1. 流程（流程图）

2. 每个语句的功能

3. 试数

如果学习一些小算法的程序

1. 尝试自己去编程解决

2. 如果解决不了，就看答案

关键把答案看懂，这个是要花很大精力，也是我们学习的重点

3. 看懂之后尝试自己去修改程序，并且知道修改后的程序的不同输出结果的含义

4. 照着答案去敲

5. 调试错误

6. 不看答案，自己把答案独立敲出来

7. 如果程序实在无法彻底理解，就把它背会，很少碰到这样的情况

强制类型转换

格式：

（数据类型）（表达式）

功能：

把表达式的值强制转化为前面所执行的数据类型

例子：

（int）（4.5+2.2） 最终值是 6

（float）（5） 最终值是 5.000000

1.0默认是double类型

odd\_average = 1.0\*odd\_sum / odd\_cnt;

//1.0默认是double类型,%f不要忘记改

sum = sum + 1.0/i; //这样写是OK的

//sum = sum + (float)(1/i); 这样写是不对的

//也可以这样写：sum = sum + 1/(float)(i); 不推荐

浮点数的存储所带来的问题：

float和double都不能保证可以精确的存储一个小数

举例：

有一个浮点型变量x,如何判断x的值是零

if（x与一个非常小的值的差的绝对值 < =0.000001）

是；

else

不是；

if（|x-0.0001| <= 0.0000001）

是

else

不是

为什么循环中更新的变量不能定义成浮点型

不能保证精确地存储一个小数

试数详细步骤举例：

1-> i=1 1<1=100 成立

sum=0+1.0/1=1 ++i i=2

2-> i=2 2<=100 成立

sum=1+1.0/2 ++i i=3

3-> i=3 3<=100 成立

sum=1+1.0/2+1.0/3 ++i i=4

4-> i=4 4<=100 成立

sum=1+1.0/2+1.0/3+1.0/4 ++i i=5

..............................

试数：1234（回文数判断）

------------------------

1-> m=1234 成立

sum=10\*0+1234%10 =4

m=m/10=1234/10 = 123

2-> m=123 成立

sum=10\*4+123%10 = 43

m=m/10=123/10 = 12

3-> m=12 成立

sum=43\*10+12%10 = 432

m=m/10=12/10 = 1

4-> m=1 成立

sum=432\*10+1%10 = 4321

m=m/10 =1/10 = 0

5-> m=0 不成立

最终sum = 4321

如何看懂一个程序（斐波拉契序列），试数：n = 6

------------------------

1-> i=3 i<=6 成立

f3=f1+f2=1+2=3

f1=f2=2

f2=f3=3

i=4

2-> i=4 i<=6 成立

f3=f1+f2=5

f1=f2=3

f2=f3=5

i=5

3-> i=5 i<=6 成立

f3=f1+f2=8

f1=f2=5

f2=f3=8

i=6

4-> i=6 i<=6成立

f3=f1+f2=13

f1=f2=8

f2=f3=13

i=7

5-> i=7 i<=6 不成立

最终f3 = 13

进制

1. 什么叫n进制

逢n进一

2. 二进制转十进制

24 + 23  + 22 + 21 + 20 11111

3. 十进制转二进制

除二取余，商零后余数倒过来

4. 不同进制所代表的数值之间的关系

十进制的3981转化成为十六进制是F8D

十进制的3981和十六进制的F8D的本质都是同一个数字

一些琐碎的运算符知识

分类：

自增 自减 三目运算符 逗号表达式

前自增 —— ++i

后自增 —— i++

前自增和后自增的异同：

相同：

最终都使i的值加1

不同：

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是i加一之前的值

为什么会出现自增：

代码更精炼

自增的速度更快 直接放在CPU内部的寄存器中进行计算

学习自增要明白的几个问题

1. 我们编程时要尽量屏蔽掉前自增和后自增的差别

2. 自增表达式最好不要作为一个更大的表达式的一部分来使用或者说

i ++ 和 ++ i 单独成一个语句，不要把它作为一个完整的复合语句的一部分 来使用

如：

int m = i++ + ++i + i + i++; //这样不但是不规范的，而且是不可移植的 顺序点

printf(“%d %d ”,i++ ++i); //同上

三目运算符

A ? B : C

等价于

if （A）

B；

else

C；

逗号表达式

格式：

(A, B, C, D)

功能：

从左到右执行

最终表达式的值是最后一项的值

快捷键

Ctrl A Alt F8 全选然后对齐