期末复习C语言知识点归纳

一、概述部分

主要考察知识点：

C程序的结构特点； C程序的扩展名；程序的开发过程；

函数的构成；

C语言使用的基本符号：ASCII 字符集；

标识符的命名规则；

关键字、用户自定义标识符

1.C程序的基本模块是函数，一个C语言源程序可以由一个或多个函数组成,有且只有1个main()函数,可以放在程序中的任何位置.

2. 函数的定义是平行的，不能嵌套定义，但可以进行嵌套调用。

3.一个C程序不论有几个函数,都是从main()函数开始执行.

4. C源程序文件的扩展名".c" 。

5. VC++6.0下C语言程序的完整实现过程包括4个基本步骤：

- 编辑　形成源文件（.c）
- 编译，形成目标文件（.obj）
- 链接，形成可执行文件（.exe）
- 运行.

6. 函数的构成：变量定义、语句、注释

7. 标识符

按照 C 语言所定义的规则来命名的符号，用作变量名、数组名、函数名等，系统定义的关键字也遵循该规则。

规则：由字母、数字、下划线组成，第一个字符必须是字母或下划线。

C 语言中的标识符分：关键字、用户定义标识符

例如: area、a234、_1234 为合法标识符

6x、a-b、a+b 为非法标识符

注意: 标识符区分大小写。

PRINTF 和 printf 是不同的标识符

用户自定义的标识符应避开 C 语言的关键字及库函数名。

4. c 语言的 32 个关键字：看附表，留印象

二、数据类型、常量、变量、表达式

注意：

常用数据类型的关键字；

常量的表达形式；

变量的定义及赋值；

各类表达式的计算规则；

不同数据类型数据混合运算时的类型转换；

典型问题：数据的混合计算、整数的分解、数据交换

1. 数据类型

c 语言中，**整型、实型、字符型** 是最基本的数据类型.

此外还有构造类型，如数组，结构体，共用体等。

2. 常量

指在程序运行中，其值不能被改变。

分为整型常量、实型常量、字符常量、字符串常量。

- **整型常量**

C 语言中主要有十进制、八进制、十六进制等：**8 进制以 0 引导，16 进制以 0x 或 0X 引导，十进制不能以 0 开头.**

十进制　例如 12、-3、32767 等。

八进制　用数字 0 开头，由 0-7 这 8 个数字组成。

例如 010、016 为合法八进制数

127、018 为非法八进制数

十六进制 用数字 0 和字母 x 或 X 开头，用 0-9 表示前 10 个数字,字母 A-F(可小写)表示后 6 个数字。

例如 0xAF、0x10、0x34 为合法十六进制数

100、ABC、0xAFH 为非法十六进制数

- **实型常量（float 或 double 型）**

表示方式有两种: 小数形式和指数形式。

小数形式:

例如: **0.123、.123、123. 都是合法形式，为 double 型。**

0.123F （float 型）

指数形式 :

例如: **2.3026 可用以下任何一种指数表示**

0.23026E1、2.3026e0、23.026E-1

但下面的指数形式为非法

E3　　.5e3.6

- 字符型常量

一个字符常量占一个字节存储空间. 用单引号限定 。有普通字符和转义字符。如：

普通字符：如

'A'、'c'、'5'都是合法的字符常量。

转义字符：如

'\n'　换行　　'\\'反斜杠

'\''　单引号　　'\"'双引号

'\ddd'　三位八进制　　'\xhh'　二位十六进制

'\0x41'(表示以十六进制数 41 为 ASCII 码值的字符，即'A'的转义字符形式).

'\0'（ 表示空字符，在字符串中用作字符串的结束标志）

注意：转义字符表面上由多个字符组成，但在内存中只占1字节的空间。

- 字符串常量

用英文双引号括起来. 如："hello\n"

- 字符串中的字符个数称为字符串的长度.空字符串长度为0.
- 字符串存储中每个字符占一个字节,字符串结尾自动加一个结束标志符'\0',所以字符串存储的时候，占用的空间长度要比串的实际长度多1.

strlen("china"),求字符串的字符个数， 5（个）

sizeof("china"),求字符串占用的存储空间，6（字节）

注意下面两种情况：

（1） char ch[100]={ “Hello” };

该串的长度（字符个数）为 5，但数组 ch 的空间长度为 100。

（2）char ch[ ]={ “Hello” };

该串的长度（字符个数）为 5，但数组 ch 的空间长度为 6。

3. 变量

指在程序运行过程中其值可以改变的量，表示存储数据的空间，需要时要先定义。

变量的命名必须符合标识符的命名规则，且不能和 C 语言中关键字同名。(例如: main 是关键字)

例如： a、b、ab、a_1 为合法变量

define、printf 为非法变量

注意： ab 是一个整体，含义不同于 a 乘以 b。

变量定义，如：

```
int   x, a1,a2;
char   ch1,ch2;
float   m;
double s;
```

- VC++ 6.0 下，int 型数据占 4 个字节，float 型数据占 4 个字节， double 型数据 占 8 个字节，.char 型数据 占 1 个字节.

- 自加和自减运算符

例如: a++和++a 等价于 a=a+1

b--和--b 等价于 b=b-1

a++和++a 的区别:

假设 a 的初值为 5

表达式 a++值为 5，最后 a 值为 6。

表达式++a 值为 6，最后 a 值为 6。

- 条件运算

由"?"和":"构成,它是根据条件从两个表达式中选择一个进行计算取值的操作,优先级高于赋值运算符,低于算术运算和关系运算.

例如： int a=1,b=2,c=3,d=4, z;

z=(a>b)? c : d; 结果：z 值为 4

三、输入输出函数

重点：输入输出的格式控制

1. 数据输出 printf

格式: printf(格式符，输出项 1，输出项 2，….)

格式符	功能
%c	输出一个字符
%d	输出十进制整数
%f	输出小数形式浮点数
%e	输出指数形式浮点数
%s	输出一个字符串

数据宽度说明:

1)在%和格式字符之间插入一个整数来指定输出宽度。

例如: %4d、%5c

2)对于 float 和 double 类型的实数，可以用 n1.n2 形式

n1 指定输出数据的宽度(包括小数点)，n2 指定小数点后小数的位数。

例如: %12.3f、%12.0f

题型：判断输出结果时格式的正确与否！

例 1 输出整型变量 a 和 b 的值，数据宽度均为 4。

printf(“a=%4d，b=%4d”,a,b);

例 2 输出单精度变量 k 的值。

printf(“%10.2f”, k);

2. 数据输入 scanf

格式: scanf(格式符，输入项 1，输入项 2，…)

格式符	功能
%c	输入一个字符
%d	输入十进制整数
%f	输入单精度数
%lf	输入双精度数
%s	输入一个字符串

题型：判断输入数据时的格式正确与否！

例 1 从键盘上输入两个整数，保存在变量 a 和 b 中。

scanf(“%d%d”,&a,&b);

例 2 从键盘上输入一个双精度数，保存在变量 data 中。

scanf(“data=%lf”,&data);

键盘输入 23.5，则输入格式应为：data=23.5

3. 字符专用的 getchar 函数和 putchar 函数

分别用来输入字符和输出字符。

例 1　从键盘上输入一个字符保存在 ch 中。

```
char  ch;
ch=getchar();
```

例 2　

```
char  c1='A', c2;
c2=c1+1;
putchar(c2);
```

输出结果是'B'。

例 3　输出一个回车换行符。

```
putchar('\n');
```

例: 从键盘上输入一串字符，最后以'!'为结束标志。分别统计大写字母，小写字母、数字出现的次数。

```
#include<stdio.h>
main()
{
    char ch;
    int n1=0,n2=0,n3=0;
    printf("请输入一串字符以!结束:\n");
    ch=getchar(); //先输入一个字符
     while(ch!='!')
    {
        if(ch>='A' && ch<='Z')
            n1++;
```

```
            if(ch>='a' && ch<='z')
                n2++;
            if(ch>='0' && ch<='9')
                n3++;
              ch=getchar(); //  继续输入下一个字符
      }
          printf("大写字母出现次数是%d\n",n1);
          printf("小写字母出现次数是%d\n",n2);
          printf(“数字出现次数是%d\n",n3);
}
```

4. 字符串专用的输入输出函数 gets()和 puts()

例如：

```
char    str[100];
gets(ch);    //输入一个字符串存入字符数组 str
puts(ch);    //输出字符数组 str 中存放的字符串
```

6．复合语句

格式:

```
{   语句 1
    语句 2
    ….
```

语句 n

}

例 2 若已经定义 int a, b;且已赋值，要将 a 和 b 中的数进行交换，下面选项中不正确的是

A）{ a=a+b，b=a-b，a=a-b }

B）{ t=a，a=b，b=t；}

C）{ a=t；t=b；b=a；}

D）{ t=b；b=a；a=t；}

结构化程序设计的三种基本结构：

顺序结构、选择结构（分支结构）、循环结构

四、选择结构

注意：

条件表达式的构造；关系运算、逻辑运算；

选择结构的基本语句：

if 语句

if-else

if-else 的嵌套结构（else 与 if 的匹配规则）

switch 语句的语法要求，执行流程

1. 关系运算符

>、<、>=、<=、==、!=

关系运算的结果只有两种：真或假

C 语言中关系运算的结果为真用 1 表示，假用 0 表示。

例 1　6>=6　　‘a’ >’A’　’1’ < ‘5’结果均为真，值为 1

例 2　6!=6　‘D’==’d’　‘9’>’F’ 结果均为假，值为 0

2.　逻辑运算符

&& (与)　　|| (或)　　! (非)

优先级:　! > && > ||

例 1 若变量 a 值为 10，变量 b 值为 16，计算表达式。

a>b || a > sqrt(b) && b>=16

最后结果为真，表达式的值为 1

3. 条件表达式的构造：

注意：

数学上的表达式：　0≤x≤10

在 C 程序中的表达：　x>=0 && x<=10

判断 char 型变量 ch 中存放的是大写字母或小写字母：

判断整数 n 能否被整除 m 整除：

判断年份是否是闰年：

3. if 语句

单分支结构：

if(条件表达式) 语句

若表达式为真，则执行语句；否则 if 语句结束。

注意：

条件表达式可以是常量、变量、关系表达式、逻辑表达式

当常量、变量、关系表达式、逻辑表达式的取值不是 0 的时候，条件为真；若取值为 0，则条件为假。

4. if-else：二选一

if(条件表达式)

{语句组 1}

else

{语句组 2}

表达式的值非 0 表示真，执行语句组 1；否则，执行语句组 2.

例 1 从键盘上输入两个整数，输出较大的数。

```
#include <stdio.h>
main()
{
    int x,y;
    printf("请输入两个整数:\n");
    scanf("%d%d",&x,&y);
    if (x>y)
      printf("%d",x);
    else
      printf("%d",y);
}
```

5. if 语句的嵌套

多选一结构：

if(表达式 1)

{语句组 1}

else if(表达式 2)

{语句组 2}

else if(表达式 3)

{语句组 3}

else

{语句组 n}

注意：

else 与 if 的配对规则：

else 总是与它前面的、最近的、没有与其他 else 进行配对的 if 进行配对。

例 1 根据输入的学生成绩，大于或等于 90 分为 A，60 分以下等级为 E，其余每 10 分一个等级。

```
#include <stdio.h>
main()
{
    int g;
    printf("请输入一个整数:\n");


    scanf("%d",&g);
    if (g>=90) printf("A\n");
    else if(g>=80) printf("B\n");
    else if(g>=70) printf("C\n");
    else if(g>=60) printf("D\n");
    else printf("E\n");
}
```

6. switch 语句

一般格式：

```
switch(表达式) /*  switch 表达式为整型或字符型值！  */
```

{

case 常量 1：语句组 1　/* case 后面必须是常量或常量表达式！*/

case 常量 2：语句组 2

.....

default:语句组　　//也可省略该语句

}

可以使用 **break 语句** 来中止后面语句的执行.

- switch 表达式通常为整型或字符型值,case 中的常量类型与之对应,case 后面的表达式可以是常量或常量表达式,不能为变量表达式.

若表达式的值与表达式 1 相等，则执行语句 1；若与表达式 2 相等，则执行语句 2.....若均不相等则执行语句 n。

例 1 分析 switch 语句的执行

```
#include <stdio.h>
main()
{
```

```
    int g=3;
    switch(g)
    {
       case 1:    printf("****\n"); break;
       case 2:     printf("####\n"); break;
       case 3:     printf("&&&&\n");
       case 4:     printf("@@@@\n"); break;
       case 5:     printf("$$$$\n"); break;
       default:    printf("~~~~\n");
    }
}
```

程序运行结果：&&&&@@@@

五、循环结构

注意：

三种循环语句：

while 语句、do-while 语句、for 语句

语法格式，流程理解；运用；while 循环和 do-while 循环的区别

典型问题：

一组有规律的数的求和、求积数、素数判断（特别重要）

1. while 循环

格式:

while(条件表达式)　循环体

若条件表达式为真，则执行循环体。再判断表达式，若仍为真，则重复执行循环体直到表达式为假时循环结束。

注意：

条件表达式可以是常量、变量、关系

表达式、逻辑表达式

当常量、变量、关系表达式、逻辑表达式的取值不是 0 的时候，条件为真；若取值为 0，则条件为假。

例 1 编程计算 S=1+2+……100。

```
#include<stdio.h>
main()
{
      int k=1;
      s=0;     //和数变量初始值为 0，切记！
      while (k<=100)
    {   s=s+k;
        k=k+1;     //循环变量变化，必须的！
      }
       printf("%d",s);
}
```

2. do-while 循环

格式:

```
do
  循环体
while(表达式);
```

先执行循环体，然后判断表达式。若为真则重复执行循环体直到表达式为假时循环结束。

例 1 用 do-while 循环计算 10!

```
#include<stdio.h>
main()
{
    int k=1;
     double   s=1; //积数变量初始值为 1，切记！
  do
  {   s=s*k;   k++; }
    while (k<=10);
    printf("%ld",s);
}
```

注意： while 循环和 do-while 循环的区别

while 循环是先判断表达式，因此循环体有可能一次也不执行。

do-while 循环是先执行一次循环体，再判断表达式，所以不论表达式为真为假，循环体至少要执行一次。

3. for 循环

格式:

for(表达式 1;表达式 2;表达式 3)　循环体

先执行表达式 1，然后判断表达式 2，若为真则执行循环体。然后执行表达式 3，再判断表达式 2，若仍为真则重复执行循环体直到表达式 2 为假时结束。

例 1 用 for 循环计算 S=1+2+…100。

```
#include<stdio.h>
main()
{
    int k,sum=0;
```

```
    for(k=1; k<=100; k++)
        sum=sum+k;
    printf(“%d”,sum);
}
```

例 2 若一个三位数个位、十位、百位的立方和等于它本身则称这个数为水仙花数。例如 153 是一个水仙花数。编程输出所有的水仙花数。

```
#include<stdio.h>
main()
{
    int a,b,c,i;
    for(i=100; i<=999; i++)
   {
        a=i/100;
        b=i%100/10;
        c=i%10;
       if(a*a*a+b*b*b+c*c*c==i)
         printf("%d\n",i);
    }
 }
```

4. 令循环中断的 break 语句

break 语句可以用在 switch 语句中用于结束某分支的执行，也可用在循环语句中，使循环提前结束。

用在循环体内表示强行退出循环。

例 1 计算 S=1+2+3+…n，直到累加和大于 5000 为止。

```
#include <stdio.h>
main()
{
    int i,s=0;
    for(i=1;   ;i++)
    {
        s=s+i;
        if(s>5000) break;
    }
    printf("%d",s);
 }
```

注意：当有多重循环时，break 只能退出最内层循环。

素数判断的算法！

5. continue 语句

用在循环体中用来跳过本次循环余下的语句，立即开始下一轮循环的判断和执行，整个循环并未因 **continue** 结束。

例 1 写出下列程序运行结果。

```
int i, k=0, s=0;
for(i=1;i<=5; i++)
{
    s=s+i;
    if(s>5)
  {   printf("i=%d,s=%d,k=%d\n",i,s,k);
      continue;
  }
      k=k+s;
  printf("i=%d,s=%d,k=%d\n",i,s,k);
 }
```

运行结果:

i=1,s=1,k=1

i=2,s=3,k=4

i=3,s=6,k=4

i=4,s=10,k=4

i=5,s=15,k=4

六、数组

注意：

数组的定义和初始化、数组与循环的结合、数组的应用（如：最大值最小值问题、）、字符串与字符数组

数组是包含多项相同类型数据的一种数据结构，它使用同一个名字命名，再用下标进行分量标识．数组元素的下标从 0 开始，数组长度为 n 时，最末元素的下标是　n-1.

1. 定义一维数组

一维数组定义格式

数据类型　数组名[数组长度]；

//数组长度指数组中可以存放的数据元素的个数，用整数常量表示，也可用代表一个正整数的符号常量表示

例 1 定义一个包含 4 个整数的数组 a

```
int  a[4];
```

例 2 定义一个包含 3 个双精度数的数组 b

```
double  b[3];
```

注意:　C 语言中数组的下界从 0 开始计数。

例如:　a[4]的 4 个元素分别为 a[0]、a[1]、a[2]、a[3]

数组有一维数组,二维数组,和多维数组.数组中的每一个分量称为一个数组元素。

2. 数值型一维数组的初始化

用一对大括号将数组的初值括起来。

例如:int a[6]={10, 20, 30, 40, 50, 60};

注意:C 语言中,不允许初值个数多于数组定义的长度。

int a[ ]={10, 20, 30, 40, 50, 60};

给初值的情况下,数组的长度可省略,系统以初值的个数作为数组的长度。

- 对部分元素的初始化,例如:int a[6]={10, 20, 30}是对前 3 个元素的赋值.

例 1　int　a[3]={1, 2, 3};

此例中 a[0]值为 1、a[1]值为 2、a[2]值为 3

例 2　int　a[5]={0};

此例中数组 a 的全部元素值均为 0

例 3　int　a[3]={1, 2, 3, 4};

此例中由于初值个数多于数组元素个数，所以非法。

例 4　int　a[ ]={0, 0, 0, 0};

此例中省略数组元素个数，初值为 4 个 0

等价于 int　a[4]={0};

注意:　数组名是一个常量值，不能对它赋值。

例如:　int　a[3];

a=5;　　此语句非法，应改为 a[0]=5;

3. 一维数组应用

例 1 从键盘上输入 10 个整数，输出最大数和最小数。

```
#include<stdio.h>
main()
{
    int a[10],i,max,min;
    printf("请输入 10 个整数:\n");


    for(i=0;i<=9;i++)
      scanf("%d",&a[i]);
    max=a[0];
    min=a[0];
    for(i=1;i<=9;i++)
```

```
    {
        if(a[i]>max) max=a[i];
        if(a[i]<min) min=a[i];
    }
      printf("最大数是%d\n",max);
      printf("最小数是%d\n",min);

}
```

例 2 斐波数列的定义如下:

1、1、2、3、5、8、13、……

编程输出斐波数列的第 40 项值。

```
#include<stdio.h>
main()
{
    long a[40];
    int i;
    a[0]=1;
    a[1]=1;
    for(i=2;i<=39;i++)
        a[i]=a[i-1]+a[i-2];
         printf("%ld",a[39]);
```

}

4.字符数组

字符串用字符数组存放。每元素存放一个字符.

如:

```
char ch[]={'\x10', '\x21', '\x1f', '\x5a'};
```

该数组中的初值字符为转义字符。

- 字符串

用双引号将一串字符括起来称字符串。如：”Hello,world!”

C 语言中用一维数组来存放，并以’\0’作为结束标志。

‘\0’就是 0，占用空间但不计入串的实际长度。

例如: 字符串”student”的长度为 7，占用空间为 8

通常利用字符数组存放字符串。如:

```
char str[20]={ "hello"};
```

该字符串的实际长度为 5 个字符,占用的数组空间长度为 20

'\0'是字符串的结束标志. 系统在存放一个字符串时,会在串的最后一个字符后添加'\0'。

- 任何一个一维数组在内存中都占用一段连续的空间.
- 用"%s"格式输入字符串时,遇到回车键结束,但获得的字符中不包含回车键本身,而是在字符串末尾添加'\0'.
- 使用一个一维 scanf()函数使用"%s"格式输入多个字符串时,输入的各字符串之间要以空格键分隔.

如： char ch[20];

scanf("%s",ch)； //从键盘输入一个字符串存入数组 ch

- 用%s 格式符为一个字符数组输入字符串时,只有第一个空格之前的字符串被读入到字符数组中.

如：对上面的 ch 数组，输入“Hello world!”则只有空格前面的“Hello”被存入数组。

- 字符串常用函数

使用下列函数时要包含头文件<string.h>

1)字符串拷贝函数

strcpy(s1,s2)

表示将 s2 复制到 s1 中。

2)字符串连接函数

strcat(s1,s2)

表示将 s2 连接到 s1 后面形成一个新字符串。

3)字符串长度函数

strlen(s)

计算出 s 的实际长度不包括'\0'。

4)字符串比较函数

strcmp(s1,s2)

若 s1>s2,函数返回值大于 0

若 s1=s2,函数返回值等于 0

若 s1<s2,函数返回值小于 0

- 字符串应用

例 1 从键盘上输入一个字符串，输出它的长度。

```
#include<stdio.h>
#include<string.h>
main()
{
        char* p, str[100];
        p=str;
        printf("请输入一个字符串:\n");
        gets(p);
        printf("%d",strlen(p));

}
```

5. 定义二维数组

- 定义格式:

类型名 数组名[一维上界][二维上界]

例 1　定义一个 2 行 2 列的整型数组 a

int　a[2][2];

此例中数组 a 共有 4 个元素，分别是 a[0][0]、a[0][1]、a[1][0]、a[1][1]。

例 2　定义一个 4 行 4 列的整型矩阵 a

int　a[4][4];

- 二维数组的初始化

用嵌套的大括号将数组初值括起来。

例 1　int　a[4][3]={ {1,2,3}, {4,5,6}, {7,8,9}, {10,11,12} };

例 2　int　a[4][3]={ {1,2}, {4,5}, {7}, {10} };

此例中系统自动给空缺的地方补 0

例 3　int　a[4][3]={ {1,2}, {4,5} };

系统自动给后面的行补初值 0

例 4　int　a[4][3]={1, 2, 4, 5};

此例中数组 a 的第一行中元素和第二行的第一个元素依次为 1,2,4,5 其余全部为 0

- 定义二维数组大小

例 1　int a[ ][3]={ { 1,2,3}, {4,5}, {6}, {8} };

有 4 个行花括号,因此说明是 4 行 3 列。

例 2　int a[ ][3]={1, 2, 3, 4, 5};表示一个 2 行 3 列数组。

- **定义并初始化二维数组时，数组的列数（第二维）不能省略！**

如：

`int a[2][]={{1,2},{3,4,5}}；是错误的。`

- 二维数组应用

例:从键盘上输入 5 个学生 3 门课程的成绩，分别计算每个学生的平均分和每门课程的平均分。

```
#include<stdio.h>
#define M 5
#define N 3
main()
{
    int a[M][N],i,j,sum;
    printf("请输入 15 个分数(0-100):\n");
    for(i=0;i<=M-1;i++)
      for(j=0;j<=N-1;j++)
        scanf("%d",&a[i][j]);
    printf("每个学生的平均分\n");
      for(i=0;i<=M-1;i++)
     {
```

```
            sum=0;
          for(j=0;j<=N-1;j++)
            sum=sum+a[i][j];
          printf("%4d",sum/N);
        }
          printf("\n");
      printf("每门课程的平均分\n");
        for(i=0;i<=N-1;i++)
        {
            sum=0;
          for(j=0;j<=M-1;j++)
            sum=sum+a[j][i];
          printf("%4d",sum/M);
        }
}
```

七、函数

注意：

函数的种类：库函数、自定义函数；

函数的定义；

函数的参数：形参、实参、参数的个数；

函数的类型；

函数的声明；

函数调用过程中参数传递问题；

1. 函数的定义

- 带返回值的函数

格式:

类型名　函数名(参数列表)

{

　　语句

　return 返回值

}

例 1 已知 F(X)=2*X+3，计算 F(1)+F(2)+….F(10)的值。

```
#include<stdio.h>
int f(int x)
{
```

```
    return 2*x+3;
}

void   main()
{
      int i,sum=0;
      for(i=1;i<=10;i++)
        sum=sum+f(i);
       printf("%d",sum);
}
```

例 2 编程输出 2 至 100 之间所有的素数之和。

```
#include<stdio.h>
int isprime(int num)
{
      int i;
      for(i=2;i<=num-1;i++)
        if(num%i==0)
           return 0;
           return 1;
}
void   main()
```

```
{
    int i,sum=0;
    for(i=2;i<=100;i++)
      if(isprime(i)==1)
        sum=sum+i;
      printf("%d",sum);
}
```

例 3 从键盘上输入两个整数，分别输出它们的最大公约数和最小公倍数。

```
#include<stdio.h>
int getmax(int a,int b)
{
    int result;
    result=a<b?a:b;
    while(a%result!=0 || b%result!=0)
        result=result-1;
        return result;
}
int getmin(int a,int b)
{
     int result=1;
```

```
        while(result%a!=0 || result%b!=0)
            result=result+1;
            return result;
}
void main()
{
      int a,b;
      printf("请输入两个整数:\n");
      scanf("%d%d",&a,&b);
      printf("最大公约数是%d\n",getmax(a,b));
      printf("最小公倍数是%d\n",getmin(a,b));
}
```

- 不带返回值的函数

格式:

```
void   函数名(参数列表)
{
       语句
}
```

- 函数的类型

表示函数返回值的类型，是函数定义

时，函数名前面的标识符，**若缺省，则系统默认为 int 型。**

- 函数调用过程中的传值和传地址

传值是指子程序中变量的变化不会影响主程序变量。

传地址是指子程序中变量的变化会影响主程序变量。

例 1 交换两变量 a 和 b 的值。(传值方式)

```
#include<stdio.h>
void change(int a,int b)
{
   int t;
   { t=a; a=b; b=t; }
}
void main()
{
   int a=3,b=5;
   printf("a=%d b=%d\n",a,b);
   change(a,b);
   printf("a=%d b=%d\n",a,b);
}
```

运行结果:

```
a=3   b=5
```

a=3　b=5

函数 change(int a,int b)为传值方式，最后 a 和 b 的值并未交换。故传值方式不能改变主程序变量的值。

例 2 交换两变量 a 和 b 的值。(传地址方式)

```
#include<stdio.h>
void change(int* a,int* b)
{
   int t;
   { t=*a; *a=*b; *b=t; }
}
void main()
{       int a=3,b=5;
   printf("a=%d b=%d\n",a,b);
   change(&a,&b);
   printf("a=%d b=%d\n",a,b);
}
```

运行结果:

a=3　b=5

a=5　b=3

函数 change(int* a,int* b)为传地址方式，最后 a 和 b 的值成功交换。故传地址方式能改变主程序变量的值。

八、指针

- 指针变量初始化的方法 int a;

  int *p=&a;

- 赋值语句的方法 int a;

  int *p;

  p=&a;

不允许把一个数赋予指针变量，故下面的赋值是错误的： int *p;p=1000; 被赋值的指针变量前不能再加“*”说明符，如写为*p=&a 也是错误的。

- 通过指针变量获得地址值

通过赋值语句将一个指针变量的值赋给另一个指针变量。

如： int *p,*q,a=0;

p=&a;

q=p;

则指针变量 p 和 q 保存的是同一个变量 a 的地址，即 p 和 q 指向同一个整型变量。对 p 指向的变量的操作也就是对 q 指向的变量的操作。

如： *p++; printf(“%d”,*q); 的输出结果为 1。

- 给指针变量赋“空”值

在 stdio.h 头文件中，系统定义了一个符号常量 NULL，其值为 0，可通过给指针变量赋一个“空”值（即 0 值），使其不指向任何内存单元，方法为：

p=NULL；　或　p=0；　　或　　p=’\0’；

指针变量定义及赋值，举例：

例 1　定义两个指向整型变量的指针 p1 和 p2。

int　*p1, *p2;

例 2　定义两个整型变量 m 和 n 及两个指针 p 和 q，并使 p 和 q 分别指向 m 和 n。

int　m, n, *p, *q;

p=&m;　　表示 p 存放变量 m 的地址

q=&n;　　表示 q 存放变量 n 的地址

例 3　定义一个包含 10 个整数的数组 a 和一个指针 p，并使 p 指向 a。

int　a[10], *p;

p=a;　表示 p 指向数组 a 的首地址，即 a[0]的地址。

注意：例 3 中 p=a 不能写成 p=&a，因为 a 本身就是地址。

例 3　指针间的赋值

假设已有下列定义：

int *q, *p, k=0;

q=&k;

例如:　p=q;　表示两指针间赋值，读作 p 指向 q。

此时 p 和 q 均指向变量 k 的地址。

例 4　空指针

在 C 语言中空值用 NULL 表示，其值为 0

例 5　定义指针时可以给指针赋值 NULL

例如:　int *p=NULL;　表示 p 没有指向任何地址

等价于 p='\0'; 或 p=0;

注意:　此时指针 p 没有实际单元，所以暂时不能访问。

例 6　通过指针引用存储单元

假设已有下列定义:

```
int   *p, m, n=0;
p=&n;
```

例如:　m=*p;　表示将 p 指向存储单元内容给变量 m

等价于 m=n;

*p=*p+1;　等价于 n=n+1;

注意:　*p=*p+1; 还可以写成下面的形式

*p+=1;　或 ++*p;　或 (*p)++;

例 7　指针的移动

当指针指向一片连续空间时，移动才有意义。

假设已有下列定义:

```
int   a[5], *p, *q, k;
p=a;
```

例如: q=p+2; 表示将 q 指向元素 a[2]的地址

p++; 表示将 p 向后移一个单元

q--; 表示将 q 向前移一个单元

k=p[2]; 等价于 k=a[3];

例 8 指针的比较

若两指针指向一片连续空间，可比较大小。

假设已有下列定义:

```
int   a[10], *p, *q, k;
p=a;
q=p+2;
```

例如: if(p<q) printf("p 小于 q");

由于条件成立，所以输出 p 小于 q。

k=q-p; 表示 p 和 q 之间相差的元素个数

最后 k 值为 2

- **函数之间地址值的传递**

1) 形参为指针变量时实参和形参之间的数据传递

在函数调用过程中，主调函数通过实参传递给被调函数的形参，这种调用称为传值调用。形参的改变不会影响实参。

如果实参传递给形参的不是普通变量，而是地址，则形参和实参指向的实际上是同一个内存单元，形参指向的存储单元的改变，也就是实参指向的存储单元的改变。这种将地址作为参数传递的调用方式称为传引用调用。

例：

传值调用：

```
main()
{
    int a=1,b=2;
    fun(a,b);
    printf(“%d,%d”,a,b);
}
void fun(int p,int q)
{
    int t;
    t=p;p=q;q=t;
}
```

输出结果： 1，2

传引用调用：

```
main()
{
    int a=1,b=2;
    fun(&a,&b);
    printf(“%d,%d”,a,b);
}
void fun(int *p,int*q)
{
    int *t;
    t=p;p=q;q=t;
}
```

输出结果：2，1

传引用调用时应当注意实参的基类型必须与形参的基类型一致。

通过这种方式，可以在被调函数中直接改变主调函数中变量的值。

- **函数返回地址值**

函数的返回值不光可以为基本类型，还可以为指针类型，即可以定义返回指针的函数，称为指针型函数。

其一般定义形式如下：

基类型 *函数名（形参表）

{

函数体

}

其形参一般为指针。

注意：在被调函数中定义的变量属于局部变量，其生存期为函数执行过程中。当被调函数执行完返回主调函数中时，被调函数中定义的变量将不再存在。所以，如果被调函数返回的地址是被调函数中定义的变量的地址，则返回的地址值无效。

- **指针的应用**

例 1 从键盘上输入两个整数，输出较大数。

(要求使用指针)

```
#include<stdio.h>
main()
{
    int a,b,*p,*q;
```

```
    p=&a;
    q=&b;
    printf("请输入两个整数:\n");
    scanf("%d%d",p,q);
    if(*p>*q)
       printf("%d",*p);
    else
       printf("%d",*q);
}
```

例 2 从键盘上输入 10 个整数，输出最大数和最小数。

```
#include<stdio.h>
main()
{
    int a[10],*p,k,max,min;
    p=a;
    printf("请输入 10 个整数:\n");
    for(k=0;k<=9;k++)
      scanf("%d",&p[k]);
    max=p[0];
    min=p[0];
    for(k=1;k<=9;k++)
   {
```

```
        if(p[k]>max) max=p[k];
        if(p[k]<min) min=p[k];
    }
    printf("最大数是%d\n",max);
    printf("最小数是%d\n",min);

}
```

九、 指针处理字符串

例 1 定义一个未知长度的字符串 s。

```
char   * s=”Hello”;
```

此例采用指针方式，s 指向字符串”Hello”的起始地址

- 字符串的赋值

假设已有下列定义:

```
char* sp, s[6];
```

例 1　给 sp 赋值”hello”，让指针变量 sp 指向字符串。

```
sp=”hello”;
```

例 2　给字符数组 s 赋值”hello”。

```
strcpy(s,”hello”);
```

表示将”hello”复制到 s 中

例 3　给字符数组 s 赋值”student”。

strcpy(s,”student”);

此例语法上没有错误，但数组 s 空间不够，最后会产生错误结果。

注意：

1)例 2 不能写成 s=”hello”；因为 s 是数组名即地址常量，故不能直接赋值。

2)strcpy 为字符串函数，需要包含头文件<string.h>。

- 字符数组的单独赋值

例 1　char str[10]={‘s’, ‘t’, ‘r’, ‘i’, ‘n’, ‘g’, ‘!’, ‘\0’};

此例中数组 str 共 10 个元素,最后 2 个元素也是’\0’此字符串共占 10 个空间，实际长度为 7。此例中若省略’\0’效果相同。

例 2　char str[ ]={‘s’, ‘t’, ‘r’, ‘i’, ‘n’, ‘g’, ‘!’, ‘\0’};

此例定义一个包含 7 个字符的字符串，最后一个’\0’不能省略。此时 str 可以作为字符串使用。

例 3　char str[ ]={‘s’, ‘t’, ‘r’, ‘i’, ‘n’, ‘g’, ‘!’};

此例只是定义一个 7 个字符的数组，但由于没有’\0’str 不能作为字符串使用，否则会产生错误。

例 9　字符数组的整体赋值

例 1　char　str[10]={“string!”};

char　str[10]=”string!”;

以上两种写法都是合法的，效果相同。

例 2　char　str[7]=”string!”;

这种方法虽然语法上没有错，但由于’\0’没有放入

注意：

数组中，可能会产生错误结果。

将上述语句改为 char　str[ ]=”string!”；

就完全正确了，系统自动计算数组大小。

这种写法简单又不会出错，推荐使用。

例 10 字符串的输入和输出

使用 scanf 和 printf 时，格式符均为%s。

例 1　char　str[15];

scanf(“%s”,str);

表示将输入的一串字符依次存放在 str 中。

例 2　char　str[15];

scanf(“%s”,&str[3]);

表示将输入的一串字符从第 4 个元素的地址开始依次存放。

例 3　char　str[15], *p;

p=str;

scanf(“%s”,p);

这和例 1 功能相同，只是使用指针方法。

例 4　char　str[ ]=”student”;

printf(“%s”,str);

表示输出整个字符串 str 的内容。

- 字符串应用

例 1 从键盘上输入一个字符串，输出它的长度。

```
#include<stdio.h>
#include<string.h>
main()
{
    char* p, str[100];
    p=str;
    printf("请输入一个字符串:\n");
    gets(p);
    printf("%d",strlen(p));
}
```

例 2 从键盘上输入一个字符串，输出其反字符串。

方法一(单个输出法)

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

```
main()
{
    char s[100];
    int i;
    printf("请输入一个字符串:\n");
    scanf("%s",s);
    for(i=strlen(s)-1;i>=0;i--)
        printf("%c",s[i]);
}
```

方法二(整体输出法)

```
#include<stdio.h>
#include<string.h>
main()
{
    char s1[100], s2[100];
    int i,j=0;
    printf("请输入一个字符串:\n");
    scanf("%s",s1);
    for(i=strlen(s1)-1;i>=0;i--)
      s2[j++]=s1[i];
      s2[j]='\0';
```

```
    printf("%s",s2);
}
```