数据结构

王迪

wangd@sdas.org

什么是数据结构

算法+数据结构=程序设计

程序设计：编制出用计算机处理问题的指令
算法：处理问题的策略
数据结构：给出问题的数学模型

为什么要学习数据结构

- 编程基础
- 计算机及相关专业考研考博课程
- 计算机等级考试课程
- 程序员考试课程

教材和参考书

- 教材：
  - 《数据结构》(C语言版本 第2版)，严蔚敏，李冬梅等，人民邮电出版社

- 参考书：
  - 《数据结构——用面向对象方法与C++描述》，殷人昆等，清华大学出版社
  - 《Data Structures and Algorithms》，Alfred V, Aho, John E.Hopcroft, Jeffrey D.Ullman. 清华大学出版社

不要将关键字重新定义为标识符

C++ 关键字全集整合

asm	do	if	return	typedef
auto	double	inline	short	typeid
bool	dynamic_cast	int	signed	typename
break	else	long	sizeof	union
case	enum	mutable	static	unsigned
catch	explicit	namespace	static_cast	using
char	export	new	struct	virtual
class	extern	operator	switch	void
const	false	private	template	volatile
const_cast	float	protected	this	wchar_t
continue	for	public	throw	while
default	friend	register	true	
delete	goto	reinterpret_cast	try	

自加、自减运算（单目运算）

说明：

①运算对象只能是一个变量。　　2++;　/* Error !*/

②前置是先运算，后引用，而后置则是先引用，后运算。

```
int  i, x;
i=5;
x=i++;  /*  x=i;  i=i+1;  */
i=5;
x=++i;  /*  i=i+1;  x=i;  */
```

运算的优先级

第一原则：单目运算的优先级高于双目运算

x=*p++ ⟺ x= *(p++) ⟺ **x=*p, p=p+1**

x=*++p ⟺ x= *(++p) ⟺ **p=p+1, x=*p,**

第二原则：

算术运算 → 关系运算 → 逻辑运算 → 赋值运算

a+b>c&&c!=0 ⟺ **((a+b)>c)&&(c!=0)**

if语句示例

```
#include  <stdio.h>
void  main(void)
{
   int  x ;
   scanf (  “%d” , &x);
   if (x!=0)  printf (“OK”  );
   else  printf( “ERROR” );
}
}
```

将if (x!=0)
改成：
if(x)或 if(x=1)或if(x==1)
if(0)或 if(1)
如何理解??

深刻理解三种循环结构如何执行

while

do while

for

1. while 循环（当型循环）

格式：

```
while(expression)
    statement;
```

流程：

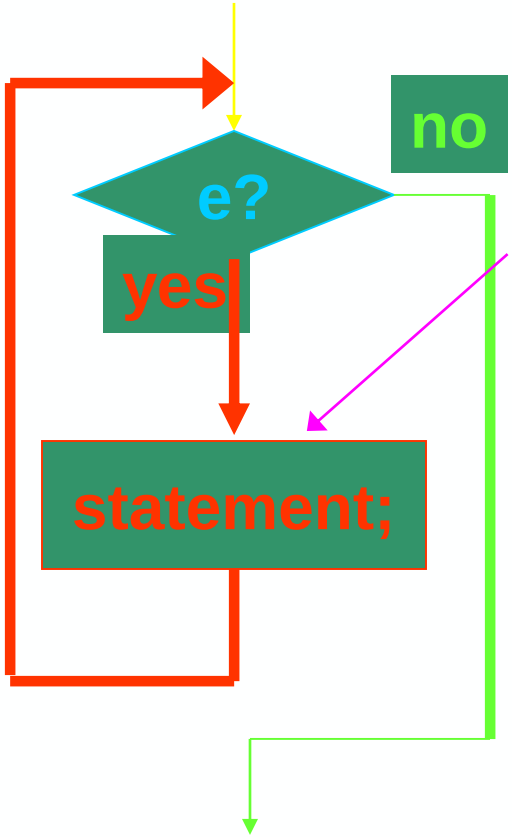


表 举例： 量非0，表
示 求 s= 1+2+3+4+…+100
代表不满足条件。

语
执

```
#include <stdio.h>
void main(void )
{ int  s=0, i=1;
  while (i<=100)
    {
      s=s+i; /* s+=i;
      i++;
    }
   printf (“s = %d \ n”, s) ;
}
```

初始化部分

条件测试

循环体

含有使条件
趋假的语句

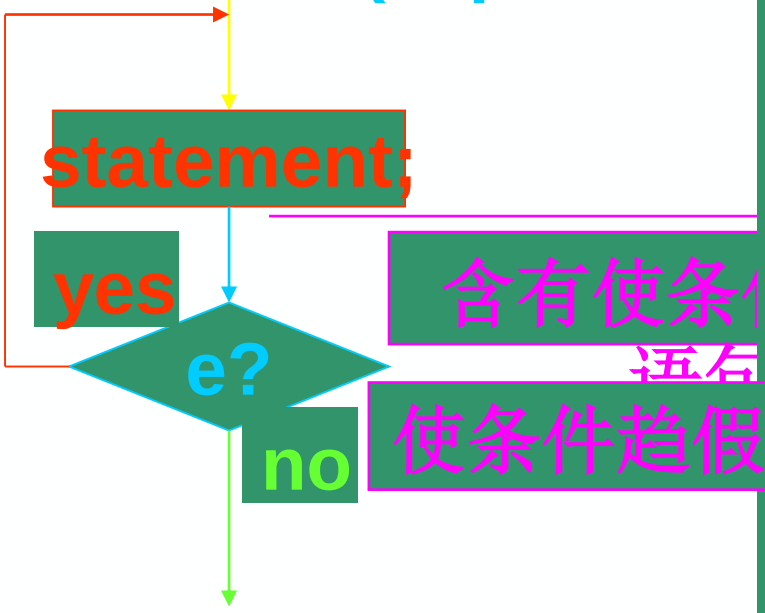
使条件趋假语句

2.do – while循环（直到型循环）

格式：

```
do {
    statement; }
```

流程：while (expressio



含有使条件
语句

使条件趋假

while循环与do-while循

举例：求：30！

```
#include <stdio.h>
void main(void)
{
  float s=1.0;
  int i=1;
  do{
     s*=i;
     i++;
   }while(i<=30);
 printf("30!=%f", s);
```

初始化。

循环体

测试条件

思考题：

用do-while实现s=1+2+…+100。用while实现30！。

❖do –while循环先执行循环体，后判条件。

初值表达式

测试表达式

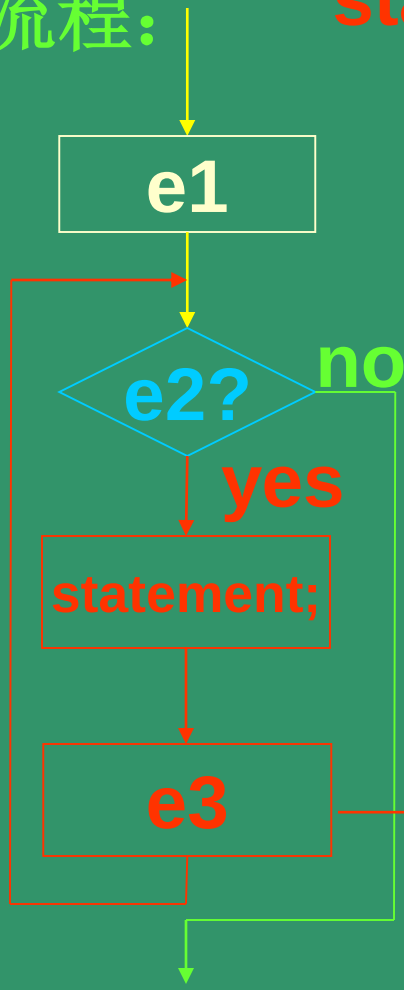
增值表达式

3. for循环

格式：

```
for (e1; e2; e3 )
    statement;
```

流程：



举例： 求：s=1+2+3+…+100

```
#include  <stdio.h>
void main (void )
{int  s,i ;

    s=0,i=1;
  for (  ;i<=100;  )
  {
    s =s + i;
    i++;
  }
  printf (“s= %d” , s) ;}
```

e1省略

e3省略

使e2趋假

在for循环中，e1、e2、e3都可以省略。

无限循环和空循环

①条件为恒真的循环——无限循环

```
while(1){…}
do{
    …}while(1);
for( ; ;){…}
```

靠条件控制的break语句退出循环。

例：程序等待直到输入字母A。

```
for ( ; ;)
{
   ch= getchar ( );
   if ( ch=='A')  break;
}
```

②循环体为空语句的循环——空循环

```
for (i=1 ;i<=MAX ; t++) ;
```

作用：程序延时。

空语句

循环示例

计算1!+2!+3!+...+9!+10!的和

```
#include<iostream>
using namespace std;
int main()
{
        long sum=0,f;
        for(int i=1;i<=10;i++)
        {
                f=1;
                for(int j=1;j<=i;j++)
                        f=f*j;
                sum=sum+f;
        }
        cout<<sum<<endl;
  return 0;
}
```

循环示例

```
#include<iostream>
using namespace std;
int main()
{
    int i,j;
   for (i=1;i<=9;i++)
   {
      for (j=1;j<=i;j++)
           cout<<i<<"*"<<j<<"="<<i*j<<" ";
      cout<<endl;
   }
    return 0;
}
```

```
1*1=1

1*2=2  2*2=4

1*3=3  2*3=6  3*3=9

1*4=4  2*4=8  3*4=12  4*4=16

1*5=5  2*5=10 3*5=15  4*5=20  5*5=25

1*6=6  2*6=12 3*6=18  4*6=24  5*6=30  6*6=36

1*7=7  2*7=14 3*7=21  4*7=28  5*7=35  6*7=42  7*7=49

1*8=8  2*8=16 3*8=24  4*8=32  5*8=40  6*8=48  7*8=56  8*8=64

1*9=9  2*9=18 3*9=27  4*9=36  5*9=45  6*9=54  7*9=63  8*9=72  9*9=81
```

循环示例

一辆汽车违反交规，撞人后逃跑。
现场有三人目击事件，但都没有记住车号，只记下车号的一些特征。
甲说：牌照的前两位数字相同；乙说：牌照的后两位数字相同，但与前两位不同；
丙是数学家，他说：四位的车号刚好是一个整数的平方。
请根据以上线索求出四位车号。

```
#include<iostream>
using namespace std;
int main()
{
  int i,j,k,c;
  for(i=1;i<=9;i++)              //i:车号前二位的取值
    for(j=0;j<=9;j++)            //j:车号后二位的取值
      if(i!=j)                    //判断二位数字是否相异
      { k=i*1000+i*100+j*10+j;   //计算出可能的整数
        for(c=31;c*c<k;c++);      //判断该数是否为另一整数的平方
        if(c*c==k) cout<<k;
      }
  return 0;}
```

字符串的比较 strcmp（str1，str2）

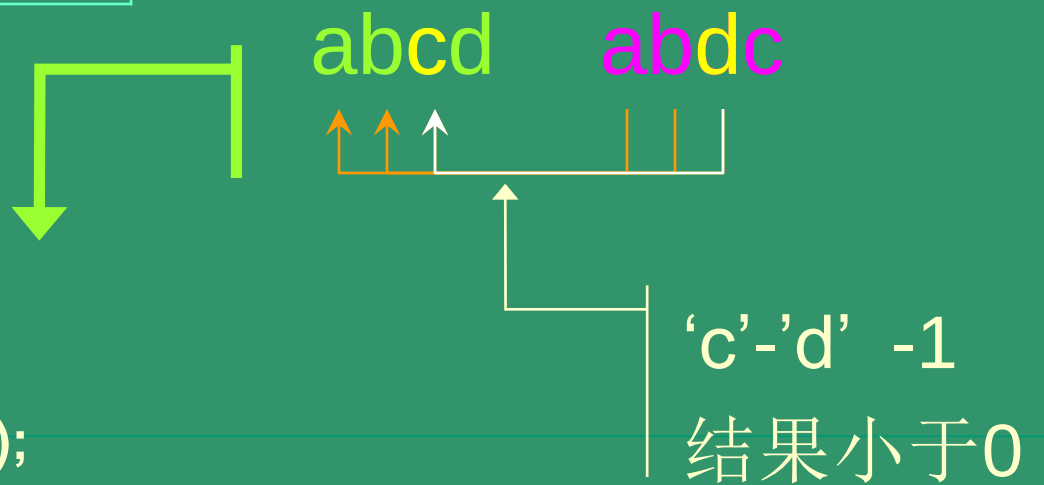
作用：对str1和str2 进行逐位无符号字符比较，直到对应位字符能够确定关系或到串尾为止。返回整型比较结果。

字符的数值关系及是字符的ASCII码值的数值关系。

比较结果如下：

比较结果	strcmp的值
str1<str2	<0
str1==str2	==0
str1>str2	>0

```
char  str1[ ]={”abcd”};
char  str2[ ]={“abcd”};
int  iRe1,iRe2,iRe3;
iRe1=strcmp(str1,”abdc”);
iRe2=strcmp(str1,str2);
iRe3=strcmp(”abcde”,str2);
```



数组的常用算法

数组是同类型数据的集合。

便于整体处理数据，数组操作的主要算法有：

1. 求极值；
2. 排序；
3. 查找；

一维数组的极值

```
#include <stdio.h>
void  main(void)
{
  int  a[10]={1,6,-2,5,4,32,47,-66,13,14};
  int iMax, iPos, i;
  iPos=0;
  iMax=a[0];
  for(i=1; i<10; i++)
        if(a[i]>iMax)
            {
              iMax = a[i];
              iPos = i;
             }
  printf("Max=%5d  Position=%5d",iMax,iPos);
}
```

假定最大值及其位置。

循环比较

当前元素比最大值大，将其赋值为新的最大值并记录其位置。

二维数组的极值

```
#include  <stdio.h>
void   main(void)
{
 float  fMin, a[3][4]={ 1.0, 3.0, 5.2, 7.4, 4.6, 5.5, 4.2, 1.2,
                        10.5, 0.23,1.3, 0.5};
 int i, j, iRow=0,iCol=0;
 fMin=a[0][0];
 for(i=0; i<3; i++)
      for(j=0;j<4;j++)
          if(a[i][j]<fMin)
              {
                fMin=a[i][j];
                iRow = i;
                iCol = j;
               }
 printf("%f7.2,iRow%5d,iCol%5d",fMin,iRow,iCol);
 }
```

假定最小值及其位置。

比较求最小值，记录其位置。

二重循环遍历所有元素

冒泡排序

基本思想：每趟不断将记录两两比较，并按“前小后大” 规则交换

21，25，49， 25*，16， 08

21，25，25*，16， 08 ， 49

21，25， 16， 08 ，25*，49

21，16， 08 ，25， 25*，49

16，08 ，21， 25， 25*，49

08，16， 21， 25， 25*，49

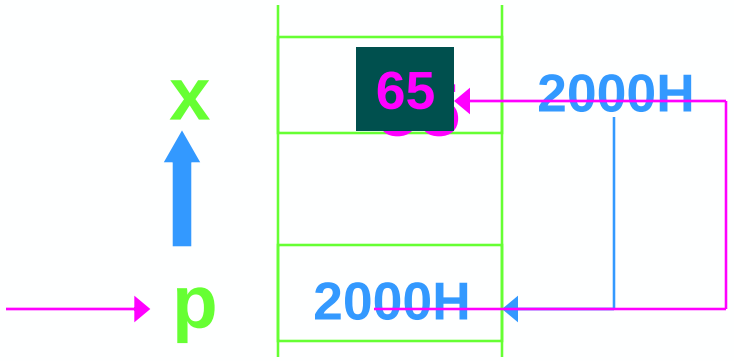
C语言的指针支持：

(1)函数的地址调用；

(2)动态分配内存；

(3)数组的地址引用。

```
#include <stdio.h>
void main (void )
{
  int x ,*p;
  x=55;
  p=&x;
  printf ( “ %d, %u ”, x, *p) ;
  *p=65;
  printf ( “ %d, %u”, x, *p) ;
}
```




```
int  *p;   *p=2;    /* Error!  */
```

(1)指针必须指向对象后，才能引用。
(2) &和 * 为互补运算。

五种算术运算

p1++; /*含义指向a后的整型单元*/

p1- -; /*指向a前的整型单元*/

p1+n; /*指向a后的n个整型单元*/

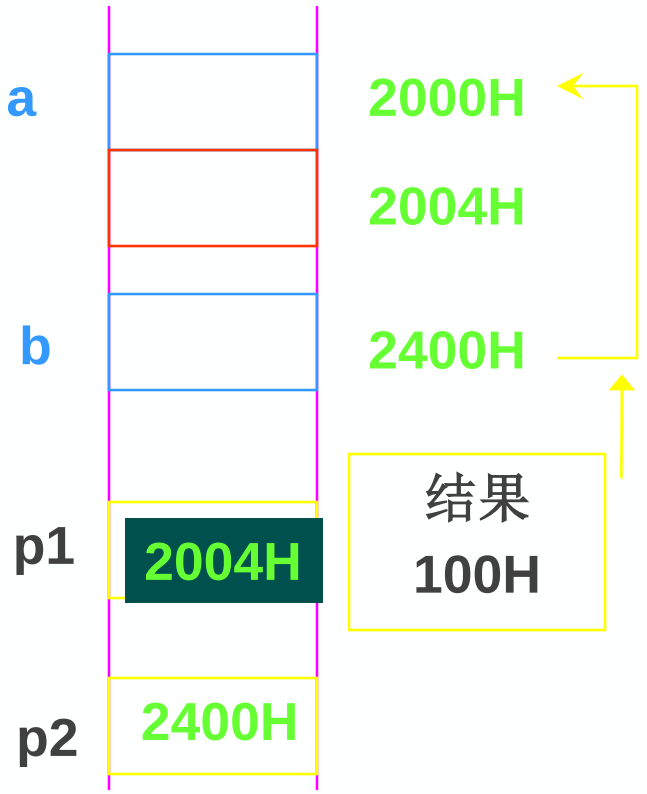
p1- n; /*指向a前的n个整型单元*/

p2- p1; /*a和b之间差的单元数*/

p ± n 相当于:

p的实际内容 ± n×sizeof(*p);

```
int  a, b, *p1, *p2;
p1=&a;
p2=&b;
```



指针与数组

数组是同类型的变量的集合，各元素按下标的特定顺序占据一段连续的内存，各元素的地址也连续，指针对数组元素非常方便。

1. 指针与一维数组

通过指针引用数组元素可以分以下三个步骤：

(1)说明指针和数组　　int　*p,a[10];

(2)指针指向数组　　p=a;　　/*指向数组的首地址*/

p=&a[0]; /*指向数组的首地址*/

(3)通过指针引用数组元素

当指针指向数组的首地址时，则下标为i的元素地址为：

p+i 或a+i

引用数组元素可以有三种方法：

下标法：　a[i]

指针法：　*(p+i)

数组名法：*(a+i)

注意：数组名是常量地址，不能改变！

a=p;　/*Error!*/

C语言的动态分配函数（ <stdlib.h> ）

malloc(m)：开辟m字节长度的地址空间，并返回这段空间的首地址

sizeof(x)：计算变量x的长度

free(p)：释放指针p所指变量的存储空间，即彻底删除一个变量

C++的动态存储分配

```
int *p1= new int;
或 int *p1 = new int(10);
delete p1;
```

new 类型名T（初值列表）

功能：

申请用于存放T类型对象的内存空间，并依初值列表赋以初值

结果值：

成功：T类型的指针，指向新分配的内存

失败：0（NULL）

```
int *p1 = new int[10];
delete [ ]p1;
```

delete 指针P

功能：

释放指针P所指向的内存。P必须是new操作的返回值

C++中的参数传递

- 函数调用时传送给形参表的实参必须与形参在类型、个数、顺序上保持一致

- 参数传递有两种方式
  - 传值方式（参数为整型、实型、字符型等）
  - 传地址
    - 参数为指针变量
    - 参数为引用类型

传值方式

把实参的值传送给函数局部工作区相应的副本中，函数使用这个副本执行必要的功能。函数修改的是副本的值，实参的值不变

```
#include <iostream.h>
void swap(float m,float n)
{float temp;
 temp=m;
 m=n;
 n=temp;
}
```

```
void main()
{float a,b;
 cin>>a>>b;
 swap(a,b);
cout<<a<<endl<<b<<endl;
}
```

传地址方式——指针变量作参数

形参变化影响实参

```
#include <iostream.h>
void swap(float *m,float *n)
{float t;
 t=*m;
 *m=*n;
 *n=t;
}
```

```
void main()
{float a,b,*p1,*p2;
 cin>>a>>b;
 p1=&a;   p2=&b;
  swap(p1, p2);
cout<<a<<endl<<b<<endl;
}
```

传地址方式——引用类型作参数

什么是引用？？？

引用：它用来给一个对象提供一个替代的名字。

```
#include<iostream.h>
void main(){
    int i=5;
    int &j=i;
    i=7;
    cout<<"i="<<i<<" j="<<j;
}
```

引用类型作形参的三点说明

（1）传递引用给函数与传递指针的效果是一样的，形参变化实参也发生变化。

（2）引用类型作形参，在内存中并没有产生实参的副本，它直接对实参操作；而一般变量作参数，形参与实参就占用不同的存储单元，所以形参变量的值是实参变量的副本。因此，当参数传递的数据量较大时，用引用比用一般变量传递参数的时间和空间效率都好。

引用类型作形参的三点说明

（3）指针参数虽然也能达到与使用引用的效果，但在被调函数中需要重复使用“*指针变量名”的形式进行运算，这很容易产生错误且程序的阅读性较差；另一方面，在主调函数的调用点处，必须用变量的地址作为实参。

结构体

在数据中，经常有一些既有联系，类型又不同的数据，它们又需要一起处理。

如：图书数据

字段：	书号	书名	价格
类型：	char	char	int

C语言允许用户按自己的需要将不同的基本类型构造成一种特殊类型，即结构。

用标识符命名的结构类型名。

```
struct   结构名{
                type   成员1；
                type   成员2；
                …

                type   成员n；
               };
```

结构标志。

结构类型中所含的成员项及其类型。

结构的定义确定了如下两点：

(1)定义结构类型，确定结构中的成员项的名称及类型。

(2)指明该结构类型的变量在内存中的组织形式。

结构指针

方式一：(*p).成员项名
方式二： p->成员项名

```
typedef struct LNode{
      ElemType   data;          //数据域
      struct LNode  *next;      //指针域
}LNode,*LinkList;
        // *LinkList为Lnode类型的指针
```

```
#include<stdio.h>
int main(void)
{
    struct stu
    {
        char *name;
        int num;
        int age;
        float score;
    } stu1;
```

```
        struct stu *p
        p = &stu1;
        stu1.name = "Wang Xiaodi";
        stu1.num = 1;
        stu1.age = 21;
        stu1.score = 99;
        printf(“%s的学号是%d，年龄是%d，今年的成绩是%d",
stu1.name , (*p).num ,  p -> age ,  p->score);
        return 0;
}
```

指向结构体的起始地址

文件的读写

```
#include <fstream>

ifstream inFile("book.txt");//读文件
inFile>>BK_head1>>BK_head2>>BK_head3;
while(!inFile.eof())  //逐行依次读取所有图书数据
        inFile>>L.BK[i].no >>L.BK[i].name>>L.BK[i].price;
inFile.close();

ofstream outFile("book_new.txt"); //写文件
for( i=0;i<L.length;i++)
        outFile<<setw(15)<<L.BK[i].no<<"\t"<<setw(50)<<L.BK[i].name<<"\
t"<< setw(5)<<L.BK[i].price<<endl;
```

总结---从编译器的角度理解C语言的语法规则和处理数据的方式

- **int x,y,*p;**（符号表，变量先定义后使用，定义后有内存单元）
- 语句的语法，如赋值语句：**x=a+b;**（正确） **a+b=x;**（错误）
- **#define N 5; (**后面有分号错误，因为N会被定义为“5;”**)**
- 函数调用通过栈，函数内的局部变量作用域，函数参数传递
- 在不同的函数中试图改变一个数据，可以通过：
  - 函数return
  - 指针或引用
  - 全局变量

总结---从机器执行的角度理解算法，将算法描述逐句变为程序

- 依次: 循环（for，while），注意循环变量的初值和终值
- 比较或分情况：分支（if，switch）
- 移动：赋值（后移a[i+1]=a[i]; 前移：a[i-1]=a[i];）
- 交换：三条赋值
- 记录或保存变量的值：一条赋值
- 对于链表，常用的三条语句如下：
  - p=L->next;　　//p指向首元结点
  - while(p!=NULL)　//p未到表尾
  - p=p->next;　　//p指向下一个结点

另外，指针保留技术：q=p;p=p->next;

建议大家--- 三个“一定要”

1. 一定要将算法理解透彻、明确数据数据结构后，再写程序
   - 明确算法思想的每一步
   - 确定逻辑结构、然后确定存储结构
2. 一定要掌握编译错误修改的技巧
   - 修改一个编译错误后重新编译
   - 先注释掉一部分代码
3. 一定要学会调试程序的方法
   - 设断点
   - F10
   - 减少测试的数据量

建议大家---三个“一定不要”

1 不要急于求成

- C++语法规则不熟练（指针、结构体、函数的定义和调用）
- 根据任务逐一编写、编译、调试各个功能模块，不该将所有代码写完再编译调试

2 不要好高骛远

- 写程序时依次考虑算法的正确性、可读性、健壮性、高效性

3 不要只读不写

- 先明确算法思想，多动手多思考，不耻下问