第5章　阅读和完善程序概述

5.1　阅读和完善程序

从本章开始，本书分析试题的后两道大题：第三题“阅读程序写结果”和第四题“完善程序”。阅读程序写结果每年有4题，每题8分，共32分，是NOIP考试中的一大拉分点。该题型考查的知识点非常多，主要包括计算机语言、多重循环的嵌套、数组的操作等。对于阅读程序题，考生一定要细心，程序从主函数main开始顺序执行，要注意循环的判断条件以及程序的输入和输出细节。

完善程序每年2题，每题14分，共28分。该部分的成败也直接决定了是否能够通过NOIP的初赛。完善程序题主要考查考生对于各类算法的理解，以及基本的函数、字符串、数组的操作。补充完善程序题的首要点就是要弄清楚出题者的意图，了解程序要实现什么样的功能，对于要实现这样的功能，需要考生将程序补充完整。完善程序所要填写的部分一般不会很难，只要弄清楚所在程序段要实现的功能就可以将程序补充完整。

5.2　常用解题方法

阅读程序写结果和完善程序题主要考查学生以下几个方面的能力。

- 程序设计语言的掌握情况。
- 程序中基本算法的掌握情况。
- 数学的知识面及运算能力。
- 细心、耐心的计算思维。

该部分的解题也是有规律可循的，本书主要总结了以下几种解题方法。

5.2.1　模拟法

在NOIP的阅读程序题中有相当一部分基础题，主要考查考生对程序设计语言的基本语法的掌握，这类题目利用纯“模拟法”就可以得出答案。

所谓“模拟法”就是利用人脑完全模拟程序的执行过程，只要题目不是很复杂，这种方法就比较奏效。但在模拟过程中，特别要注意各个变量的变化以及书写的认真和整洁，因为一个变量的计算错误就会引起整个程序结果的错误。

案例展示：

```
#include<iostream>
using namespace std;
int main(){
    int i,a,b,c,d,f[4];
    for(i=0;i<4;i++)
        cin>>f[i];
    a=f[0]+f[1]+f[2]+f[3];
    a=a/f[0];
    b=f[0]+f[2]+f[3];
    b=b/a;
    c=(b*f[1]+a)/f[2];
    d=f[(b/c)%4];
    if(f[(a+b+c+d)%4]>f[2])
        cout<<a+b<<endl;
    else
        cout<<c+d<<endl;
    return 0;
}
```

输入：

```
9 19 29 39
```

输出：_____

【分析】 该题是2008年阅读程序写结果第1题，对于无规律可循且不是很复杂的题目，可以直接使用“模拟法”，具体模拟如表5-1所示(a,b,c,d,数组f都是整型)。

表5-1 模拟法的具体模拟

数据模拟	对应语句
f[0]=9,f[1]=19,f[2]=29,f[3]=39	cin>>f[i];
a=f[0]+f[1]+f[2]+f[3]=9+19+29+39=96	a=f[0]+f[1]+f[2]+f[3];
a=a/f[0]=96/9=10	a=a/f[0];
b=f[0]+f[2]+f[3]=9+29+39=77	b=f[0]+f[2]+f[3]
b=b/a=77/10=7	b=b/a;
c=(b*f[1]+a)/f[2]=(7*19+10)/29=4	
d=f[(b/c)%4]=f[(7/4)%4]=f[1]=19	
(a+b+c+d)%4=0	if(f[(a+b+c+d)%4]>f[2])
if(f[0]>f[2]) f[(a+b+c+d)%4]>f[2]➡f[0]>f[2]➡9>29 不成立 执行 c+d	cout<<a+b<<endl; else cout<<c+d<<endl;

【参考答案】 23

5.2.2　先猜测，后验证

模拟法虽然奏效，但如果考生对整个程序要完成的功能不是很理解，就会造成模拟法的速度很慢。如果考生知道了程序的功能，那么对阅读程序的效率提高就会很大。所以在阅读程序时，考生可以借助以前阅读程序的功底以及程序中变量和函数的提示大胆猜测程序的功能，然后再验证。这就要求考生在平时学习时，一方面要及时总结经典或者常用的一些功能代码段，另一方面要求学员在阅读程序时能够先整体、后局部，及时将含有这些代码段的函数或者段落提取出来。

案例展示：

```
#include<iostream>
#include<cstring>
using namespace std;
#define MAX 100
void solve(char first[], int spos_f, int epos_f, char mid[], int spos_m, int epos_m)
{
    int i,root_m;
    if(spos_f>epos_f)
        return;
    for(i=spos_m;i<=epos_m;i++)
        if(first[spos_f]==mid[i])
        {
            root_m=i;
            break;
        }
    solve(first,spos_f+1,spos_f+(root_m-spos_m),mid,spos_m,root_m-1);
    solve(first, spos_f+(root_m-spos_m)+1,epos_f,mid,root_m+1,epos_m);
    cout<<first[spos_f];
}
int main()
{
    char first[MAX],mid[MAX];
    int len;
    cin>>len;
    cin>>first>>mid;
    solve(first,0,len-1,mid,0,len-1);
    cout<<endl;
    return 0;
}
```

输入：

```
7
ABDCEGF
```

```
BDAGECF
```

输出：_____

【分析】 该题是2008年阅读程序写结果第4题。阅读该题，可以发现里面有函数调用和递归调用，然后发现其中有first、mid以及root这三个单词，first常对应于二叉树的先序遍历，mid常对应于二叉树的中序遍历，root常对应于二叉树的根节点。由此可以大胆推测这是一个有关二叉树的题目。由solve的意思（解决）可以进一步推测该题是由先序的中序遍历解决后序遍历，然后根据输入验证输出结果，验证无误后可快速得出答案。

【参考答案】 DBGEFCA

5.2.3 表格法

在程序执行过程中，常常涉及双重循环、递归调用等稍微复杂的过程，这些过程模拟往往涉及较多的变量变化，稍不留神就可能会导致整个程序的错误。为了让整个过程整洁有序，快速发现程序的运行规律，设计表格是非常有必要的。

案例展示：

```
#include<iostream>
using namespace std;
int main(){
    int i,j,sum;
    sum=0;
    for(i=1;i<=4;i++)
        for(j=1;j<=4;j++)
        {
            sum=sum+i*j;
            cout<<sum+i*j<<' ';
        }
    return 0;
}
```

输出：_____

【分析】 该程序可以清晰地知道是一个4×4的双重循环求和，表格法更能清楚地展示结果。

i	1				2			
j	1	2	3	4	1	2	3	4
sum	1	3	6	10	12	16	22	30
i	3				4			
j	1	2	3	4	1	2	3	4
sum	33	39	48	60	64	72	84	100

【参考答案】 1 3 6 10 12 16 22 30　33 39 48 60 64 72 84 100

第6章　基本结构

6.1　基本知识介绍

C++程序必须有一个main函数，而且一个源程序只能有一个main函数。程序的运行总是从main函数开始；程序由一个main函数和0个或多个其他函数构成。main函数可以调用其他函数，但是不能被其他函数调用；语句均是以分号作为语句结束符；程序中大小写字母代表不同的含义；程序中使用的任何变量均需先定义、后使用。

main函数的基本结构：

```
int main(){
语句
    return 0;
}
```

6.1.1　常量与变量

(1) 常量

在程序中，值始终不变的数据称为常量。如字符常量'a'，整型常量10，实型常量3.14。

另外还有一种用标识符表示数值不变的常量，称为符号常量。一般符号常量均大写，如：

```
const float PI=3.14;
```

(2) 变量

变量是一个有名字、有特定属性的存储单元。在程序运行期间，变量的值是可以改变的。变量必须先定义、后使用。

变量可以在定义的时候初始化，如：

```
int a=1,b=1;
```

也可以连续赋值，如：

```
int a,b;
a=b=1;
```

但不允许一边定义一边连续赋值，如：

```
int a=b=1;     //不能同时定义 a,b 并初始化为 1
```

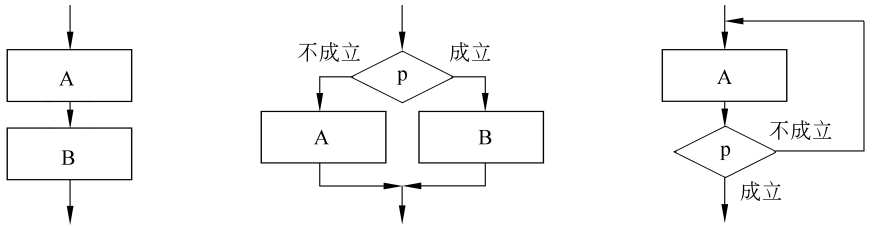
(3) 变量的分类

根据变量的作用域不同，可分为局部变量和全局变量。

根据变量存储类别可分为四类：自动变量、静态局部变量、外部变量、寄存器变量。

6.1.2 C++ 的三种基本控制结构

C++ 的三种基本控制结构如图 6-1 所示。



(a) 顺序结构　(b) 选择结构　(c) 循环结构

图 6-1　三种基本控制结构

1. 顺序结构

顺序结构按照程序的先后顺序自上而下执行。图 6-1(a)中 A 和 B 是顺序执行的，即执行完 A 框所指定的操作后，接着执行 B 框所指定的操作。

所有的程序从整体而言都是一种顺序结构。

2. 选择结构

选择结构又称分支结构，此结构包含一个判断框，根据给定的条件 p 是否成立而选择执行 A 框或 B 框。

选择结构又分为单分支、双分支和多分支三种。

(1) 单分支结构

单分支结构的表达式如下：

```
if(表达式)
    语句;
```

单分支结构的含义是：若表达式为真(值不为 0)，则执行语句，否则不执行。若包含多条语句，则加一对{}。

(2) 双分支结构

双分支结构的表达式如下：

```
if (表达式)
    语句 1;
else
    语句 2;
```

双分支结构的含义是：若表达式为真，则执行语句 1，否则执行语句 2。

(3) 多分支结构

多分支结构往往都是嵌套使用的，在 if 语句中又包含一个或多个 if 语句，称为 if 语句的嵌套。

if 语句的嵌套表达式如下：

```
if(表达式 1)
    语句 1;
else
    if(表达式 2)
        语句 2;
    else
        语句 3;
```

else 总是与它前面最近且未配对过的 if 匹配。

复合语句内的 if 关键字对于外界是不可见的。

例如：

```
if(表达式 1)              //第 1 个 if
    if(表达式 2)          //第 2 个 if
    {
        if(表达式 3)      //第 3 个 if
        语句 3;
    }
    else
        语句 2;           //与第 2 个 else 相匹配
```

(4) 用 switch 语句实现多分支选择结构

switch 语句表达式如下：

```
switch (表达式){
    case 常量表达式 1: 语句 1;break;
    case 常量表达式 2: 语句 2; break;
    ...
    case 常量表达式 n: 语句 n; break;
    default: 语句;
}
```

注意：

- 表达式的值必须是整型、字符型或枚举型；
- 多个 case 标号可以共用一组语句序列，以实现对多个常量执行同一个操作；
- default 可以省略；
- break 表示终止 switch，转而执行 switch 下面的语句；若不加 break，则执行完 case 后面的分支后会顺序执行下一个 case 分支。

3. 循环结构

(1) 用 for 语句实现循环

for 循环语句表达式如下：

```
for(初值表达式 1;循环条件表达式 2;循环变化表达式 3)
    {
        循环语句;
    }
```

注意：

- 三个表达式都可以省略，但是分号不能省略；
- 执行时，表达式 1 只在开始时执行一次；然后判断表达式 2 是否为真，若为真则执行循环语句，否则不执行。

(2) 用 while 语句实现循环

while 语句——当型循环表达式如下：

```
while(条件表达式)
{
    循环语句;
}
```

- 表达式为真(表达式为非 0)时，执行循环语句；
- 先判断，后循环；
- 当有多条语句要执行时，要用花括号把多个语句括起来；
- 在循环语句中必须有控制循环改变的语句，否则会出现死循环。

do…while 语句——直到型循环表达式如下：

```
do{
    循环语句;
}while(条件表达式);
```

- do…while 循环后面的分号一定不能丢；
- do…while 循环先执行、后判断，至少执行一次。

(3) 跳转语句

使用跳转语句可以实现程序执行流程的无条件跳转。

C++ 提供了四种跳转语句，分别如下。

- break 语句

在循环体内的 break 语句，可以使循环立即结束，退出循环继续向下执行。但是 break 语句只能退出本层循环，如：

```
for(…){
    while(…){
        …
        if(…)
```

```
            break;
        }
    }
```

上面的 if 条件若满足，则退出内层的 while 循环而不会退出 for 循环，开始下一次的 for 循环。

- continue 语句

continue 语句的作用是终止本次循环，开始执行下一次循环。

- return 语句

return 表示把程序流程从被调函数转向主调函数并把表达式的值带回主调函数，实现函数值的返回，返回时可附带一个返回值，由 return 后面的参数指定；也可用于循环体中满足条件时的结束语句。

- goto 语句

```
goto <标志>;
```

- 标志常用格式

```
<标志>:<语句>
```

goto 语句的作用是使程序执行流程跳转到标记的语句处。

6.2　历年真题解析与知识点巩固

题目 1　2018 年阅读程序第 1 题

```
#include<cstdio>
char st[100];
int main()
{
    scanf("%s",st);
    for(int i=0;st[i];++i)
    {
        if('A'<=st[i]&&st[i]<='Z')
            st[i]+=1;
    }
    printf("%s\n",st);
    return 0;
}
```

输入：

```
QuanGuoLianSai
```

输出：_____

【分析】 该题目实现的功能是：大写字母加 1，其他字符不变。

【参考答案】 RuanHuoMianTai

【核心知识点】 字符串或字符数组中大写字母的处理。

```
if('A'<=st[i]&&st[i]<='Z')
    st[i]+=1;
```

【巩固】 编写程序实现功能：输入一串字符，将其中的小写字母变成大写字母，大写字母变成小写字母，其余字符保持不变。

题目 2　2018 年阅读程序第 2 题

```
#include<cstdio>
int main()
{
    int x;
    scanf("%d",&x);
    int res=0;
    for(int i=0;i<x;++i){
        if(i * i %x ==1){
            ++res;
        }
    }
    printf("%d",res);
    return 0;
}
```

输入：

```
15
```

输出：_____

【分析】 本题的功能是：求 0^2 到 x^2 中对 x 取余值为 1 的个数。

【参考答案】 4

【核心知识点】 对数据进行筛选。

```
for(int i=0;i<x;++i){
    if(i * i %x ==1){
        ++res;
    }
}
```