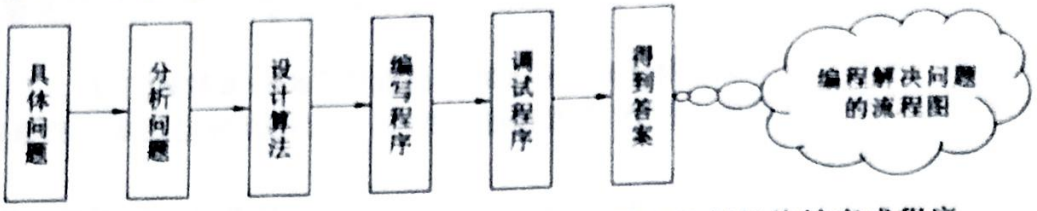
第二章　程序设计基础知识

“程序＝算法＋数据结构”，其中算法可以理解为解决实际问题的方法，数据结构则是计算机存储、组织数据的方式。



算法是程序设计的核心和灵魂，有了算法就可以结合数据结构转变成程序。

第1节　程序基本常识

一、算法的定义及特征

算法就是解决问题的操作步骤。一个算法必须满足以下五个重要的特征：

1.有穷性：执行有穷步，在有穷的时间内完成。

2.确切性：每一条指令必须有确切的含义，不会产生歧义。在任何条件下算法只有唯一的一条执行路径。

3.可行性：算法中的操作可以通过执行有限次来实现。

4.输入：一个算法有零个或者多个输入。

5.输出：一个算法有一个或者多个输出。

二、算法的复杂度

同一个问题可用不同的算法来解决，而一个算法质量的优劣将影响到算法乃至程序的效率。算法分析的目的在于选择合适的算法和改进算法。一个算法的评价主要从时间复杂度和空间复杂度来考虑。

（一）空间复杂度

空间复杂度指执行算法所需占用的内存空间。算法执行时所需的存储空间包括程序本身占用的空间、输入数据占用的空间以及算法执行时所需空间。

算法在时间的高效性和空间的高效性之间通常是矛盾的，所以一般会取一个平衡点。通常假设程序运行在足够大的内存空间中，所以研究更多的是算法的时间复杂度。

（二）时间复杂度

1.定义及表示

指算法执行时所需消耗时间，通常用算法执行次数来衡量，记作 T(n)＝O(f(n))，其中 f(n)是算法执行次数的函数。用大写 O()来体现算法时间复杂度的记法，称之为大 O 记法。

2.计算步骤

(1)找出基本操作确定规模 n。通常取最深层循环内的语句所描述的操作作为基本操作。

(2)计算执行次数的函数 f(n)并求出 T(n)=O(f(n))。O(f(n))是取 f(n)中 n 增长最快的那项系数为 1 的形式。

例 1:求出以下算法的时间复杂度。

```
void fundemo (int n){
    int i=1,j=100;
    while (i<n){
        i+=2;
        ++j;
    }
}
```

分析:该算法基本操作为 i+=2 和++j,规模为 n。i 最后的值为 1+2m(m 代表执行次数),则 1+2m+k=n(k 是足够小的数),m=(n−1−k)/2,故 f(n)=(n−1−k)/2,此时 f(n)增长最快的那项是 n,故 T(n)=O(f(n))=O(n)。

3.计算规则

(1)加法规则

T(n,m)=T1(n)+T2(m)=O(max{f(n),g(m)}) //并列算法

(2)乘法规则

T(n,m)=T1(n)*T2(m)=O(f(n)*g(m)) //嵌套算法

(3)比较规则

$O(1)<O(\log n)<O(n)<O(n\log n)<O(n^2)<O(n^3)<\cdots<O(n^k)<O(2^n)<O(n!)<O(n^n)$

课堂练习

1.【NOIP2011】在使用高级语言编写程序时,一般提到的“空间复杂度”中的“空间”是指(　　)。

A. 程序运行时理论上所占的内存空间　B. 程序运行时理论上所占的数组空间

C. 程序运行时理论上所占的硬盘空间　D. 程序源文件理论上所占的硬盘空间

2.【NOIP2013】斐波那契数列的定义如下:$F_1=1,F_2=1,F_n=F_{n-1}+F_{n-2}(n\geqslant 3)$。如果用下面的函数计算斐波那契数列的第 n 项,则其时间复杂度为(　　)。

```
int F(int n) {
    if (n<=2)
    return 1;
    else
    return F(n-1)+F(n-2);
}
```

A. O(1)　　B. O(n)　　C. O(n2)　　D. $O(F_n)$

3.【NOIP2013】T(n)表示某个算法输入规模为 n 时的运算次数。如果 T(1)为常数，且有递归式 $T(n)=2*T(n/2)+2n$，那么 T(n)=(　　)。

A. O(n)　　B. O(nlogn)　　C. $O(n^2)$　　D. $O(n^2\log n)$

4.【NOIP2015】设某算法的计算时间表示为递推关系式 $T(n)=T(n-1)+n$(n 为正整数)及 $T(0)=1$，则该算法的时间复杂度为(　　)。

A. O(logn)　　B. O(nlogn)　　C. O(n)　　D. $O(n^2)$

5.【NOIP2016】假设某算法的计算时间表示为递推关系式

$$T(n)=2T\left(\frac{n}{4}\right)+\sqrt{n}$$

$$T(1)=1$$

则算法的时间复杂度为(　　)。

A. O(n)　　B. $O(\sqrt{n})$　　C. $O(\sqrt{n}\log n)$　　D. $O(n^2)$

6.【NOIP2017】若某算法的计算时间表示为递推关系式：

$$T(N)=2T(N/2)+N\log N$$

$$T(1)=1$$

则该算法的时间复杂度为(　　)。

A. O(N)　　B. O(NlogN)　　C. $O(N\log^2 N)$　　D. $O(N^2)$

7.【NOIP2012】如果对于所有规模为 n 的输入，一个算法均恰好进行(　　)次运算，我们可以说该算法的时间复杂度为 $O(2^n)$。

A. 2^{n+1}　　B. 3^n　　C. $n*2^n$　　D. 2^{2n}

第2节 C++语言基础

由于篇幅所限，本书在此只介绍C++语言的大致语法和使用，以帮助读者快速读懂程序。更详细的使用请参考《信息学奥赛一本通》这本书的前面几章或者自行在网上查找资源。

一、程序的基本构成

```
#include<iostream>
using namespace std;
int main(){                    //main()是程序开始执行的地方
    cout<<"Hello world";       //输出 Hello world
    return 0;
}
```

(1)头文件<iostream>包含了程序中如输入输出即cin、cout的定义。

(2)using namespace std;告诉编译器使用std命名空间。命名空间是为了作为附加信息来区分不同库中相同名称的函数、类、变量等。本质上，命名空间就是定义了一个范围。

(3)下一行int main()是主函数，程序从这里开始执行。

(4)//是一个单行注释，用于解释说明。单行注释以//开头，在行末结束。

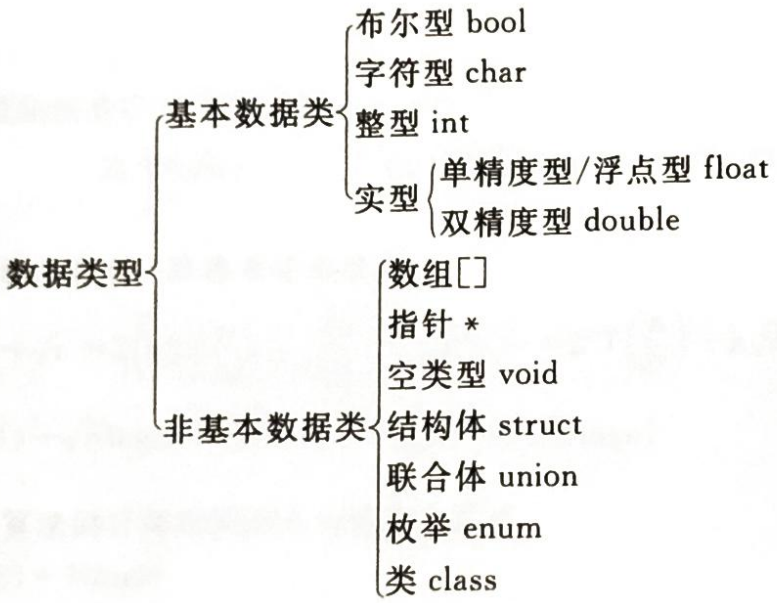
(5)cout<<"Hello World";用于输出，在屏幕上显示消息"Hello World"。

(6)return 0;终止main()函数，并向调用进程返回值0。

(7)分号是语句结束符。也就是说每个语句必须以分号结束。

(8){}称为块，其中的所有语句是一个语句块。

二、变量及数据类型




```
#include<iostream>
using namespace std;
int main(){
    float x;
    int i;
```

```
    x=3.7;
    i=(int)x;//强制类型转换
    cout<<"x="<<x<<",i="<<i<<endl;//输出 x=3.7,i=3
    return 0;
}
#include<iostream>
using namespace std;
struct stuInfo {
    int ID;
    char name[20];
    char sex;
    int age;
};
int main(){
    stuInfo x={13,"xiaoma",'m',35};
    stuInfo * p=&x;          //指针 p 指向 x
    cout<<x.ID<<endl;
    cout<<(* p).ID<<endl; //指针访问方式一
    cout<<p->ID<<endl;    //指针访问方式二
    return 0;
}
```

三、类

C++在C语言的基础上增加了面向对象编程。类是C++的核心特性，通常被称为用户定义的类型。类包含了数据表示和用于处理数据的方法。类中的数据和方法都称为类的成员。

```
class Box{//方法一
    public;//public 是类型修饰符，具体意义见“类访问修饰符”
        double length; //长度
        double width;  //宽度      } 类的数据成员，也就是类中的变量
        double height; //高度
        double getVolume(){
            return length *width*height;  } 类的成员函数，也就是类中的函数
        }
};
```

四、程序的基本结构

一般一个程序可以用顺序、选择和循环三种基本结构组合而成。

1.顺序结构

顺序结构是最简单、最常用的结构，语句与语句之间按从上到下的顺序执行即可。

2.选择结构

选择结构是先根据条件做出判断，再决定执行哪一种操作的算法结构，必须包含判断框。当条件 P 成立(或称为真)时执行 A，否则执行 B，不可能两者同时执行，但 A 或 B 两个框中可以有一个是空的，即不执行任何操作。常用的是 if、if…else、switch。

```
#include<iostream>
using namespace std;
int main(){
    int a=20;
    if (a==15){//如果 if 条件为真，则输出下面的语句
        cout<<"a 的值是 10"<<endl;
    }else if (a==20){//如果 else if 条件为真，则输出下面的语句
        cout<<"a 的值是 20"<<endl;
    }else if (a==25){
        cout<<"a 的值是 30"<<endl;
    }else {
        cout<<"没有匹配的值"<<endl;
    }
    cout<<"a 的准确值是"<<a<<endl;
    return 0;
}
```

3.循环结构

在一些算法中，经常会出现从某处开始，按照一定条件，反复执行某一处理步骤的情况，这就是循环结构。常用的有 while、for、do…while

```
#include<iostream>
using namespace std;
int main(){
    for (int a=1;a<10;a=a+1){
        cout<< "a 的值:"<<a<<endl;
    }
    int b=1;
    while (b<10){
        cout<<"a 的值:"<<b<<endl;
        b++;
    }
    return 0;
}
```

五、函数

函数是用来实现某些特定功能而封装成的一个模块。在创建函数时，必须编写其定义。所有函数定义包括以下组成部分：

返回类型　函数名(形参列表){

函数主体

}

1.函数名：每个函数都必须有一个名称。通常，适用于变量名称的规则同样也适用于函数名称。

2.形参列表：保存传递给函数的值的变量列表。如果没有值传递给函数，则其形参列表为空。

3.函数主体：处理函数正在执行的任务的一组语句，这些语句包含在一组大括号中。

4.返回类型：函数可以将值发送回调用它的程序模块。返回类型是要发送回的值的数据类型。

例如：编写求半径为 r 的圆的面积的函数 area。

```
//这是一个求值的函数，函数中设置一个形参 r，用于接收圆的半径。
//将参数类型和返回值类型定义为 double。
double area(double r) //返回值类型为 double
{
    double s;
    s=3.14*r*r;
    return s;//返回计算结果
}
```

在执行函数时，函数参数，局部变量（包括 const 局部变量），函数调用后返回的地址都在内存的栈区上创建，函数执行结束时这些存储单元自动被释放。

六、递归函数

递归，就是在运行的过程中调用自己。

构成递归需具备的条件：

1.子问题须与原始问题为同样的事，且更为简单；

2.不能无限制地调用本身，必须有个出口，化简为非递归状况处理。

例如编写代码求 1+2+…+n 的值，其中 n≤20。

```
#include<iostream>
using namespace std;
//求解 1+2…+n
int sum(int x){
    if (x==1)return 1;//出口，这样就不能无限制地调用本身了
    return sum(x-1)+x;//子问题 sum(x-1)与原始问题 sum(x)同样的事，且更为简单
}
int main(){
    int n;
    cin>>n;
    cout<<sum(n);
    return 0;
}
```

课堂练习

1.【NOIP2014】若有如下程序段，其中 s、a、b、c 均已定义为整型变量，且 a、c 均已赋值，c>0。

```
s=a;
for (b=1;b<=c;b++)
s+=1;
```

则与上述程序段功能等价的赋值语句是(　　)。

A. s=a+b　　B. s=a+c　　C. s=s+c　　D. s=b+c

2.【NOIP2014】要求按以下程序计算 s=1+1/2+1/3+…+1/10。

```
#include<iostream>
using namespace std;
int main(){
    int n;
    float s;
    s=1.0;
    for (n=10;n>1;n--)
        s=s+1/n;
    cout<<s<<endl;
    return 0;
}
```

程序运行后输出结果错误，导致错误结果的程序行是(　　)。

A. s=1.0;　　B. for (n=10;n>1;n--)

C. s=s+1/n;　　D. cout<<s<<endl;

3.【NOIP2014】有以下程序：

```
#include<iostream>
using namespace std;
int main(){
    int s,a,n;
    s=0;
    a=1;
    cin>>n;
    do{
        s+=1;
        a-=2;
    }while(a!=n);
    cout<<s<<endl;
    return 0;
}
```

若要使程序的输出值为 2，则应该从键盘给 n 输入的值是(　　)。

A. −1　　B. −3　　C. −5　　D. 0

4.【NOIP2016】有以下程序：

```
#include<iostream>
using namespace std;
int main(){
    int k=4,n=0;
    while (n<k){
        n++;
        if (n%3!=0)
            continue;
        k--;
    }
    cout<<k<<","<<n<<endl;
    return 0;
}
```

程序运行后的输出结果是(　　)。

A. 2,2　　B. 2,3　　C. 3,2　　D. 3,3

5.【NOIP2018】为了统计一个非负整数的二进制形式中 1 的个数，代码如下：

```
int CountBit (int x){
  int ret=0;
  while (x){
    ret++;
    _________;
  }
  return ret;
}
```

则空格内要填入的语句是(　　)。

A. x>>=1　　B. x&=x-1　　C. x|=x>>1　　D. x<<=1

6.【NOIP2013】下列程序中，正确计算 1,2,…,100 这 100 个自然数之和 sum（初始值为 0）的是(　　)。

A.

```
i=1;
do{
    sum+=i;
    i++;
} while (i<=100);
```

B.

```
i=1;
do{
    sum+=i;
    i++;
} while (i>100);
```

C.

```
i=1;
while (i<100){
    sum+=i;
    i++;
}
```

D.

```
i=1;
while (i>=100){
    sum+=i;
    i++;
}
```

7.【NOIP2014】若有变量 int a,float x,y,且 a=7,x=2.5,y=4.7,则表达式 x+a%3*(int)(x+y)%2/4 的值大约是(　　)。

A. 2.500000　　B. 2.750000　　C. 3.500000　　D. 0.000000

8.【NOIP2014】设变量 x 为 float 型且已赋值,则以下语句中能将 x 中的数值保留到小数点后两位,并将第三位四舍五入的是(　　)。

A. x=(x*100)+0.5/100.0;　　B. x=(x*100+0.5)/100.0;

C. x=(int)(x*100+0.5)/100.0;　　D. x=(x/100+0.5)*100.0;

9.【NOIP2014】以下程序段实现了找第二小元素的算法。输入是 n 个不等的数构成数组 S,输出 S 中第二小的数 SecondMin。在最坏情况下,该算法需要做(　　)次比较。

```
if (S[1]<S[2]){
    FirstMin=S[1];
    SecondMin=S[2];
}else {
    FirstMin=S[2];
    SecondMin=S[1];
}
for (i=3;i<=n;i++)
    if (S[i]<SecondMin)
    if (S[i]<FirstMin){
        SecondMin=FirstMin;
        FirstMin=S[i];
    }else {
        SecondMin=S[i];
}
```

A. 2n　　B. n−1　　C. 2n−3　　D. 2n−2

10.【NOIP2008】递归过程或函数调用时,处理参数和返回地址,通常使用一种称为(　　)的数据结构。

A. 队列　　B. 多维数组　　C. 线性表　　D. 栈

11.【NOIP2012】在程序运行过程中,如果递归调用的层数过多,会因为(　　)引发错误。

A. 系统分配的栈空间溢出　　B. 系统分配的堆空间溢出

C. 系统分配的队列空间溢出　　D. 系统分配的链表空间溢出

不定项选择题

1.【NOIP2013】下列程序中，正确计算 1,2,…,100 这 100 个自然数之和 sum（初始值为 0）的是(　　)。

A.	`for (i=1;i<=100;i++)`
	`  sum+=i;`
		B.	`i=1;`
			`while (i>100){`
			`  sum+=i;`
			`  i++;`
			`}`

C.	`i=1;`
	`do{`
	`  sum+=i;`
	`  i++;`
	`} while (i<=100);`
		D.	`i=1;`
			`do{`
			`  sum+=i;`
			`i++;`
			`} while (i>100);`

第 3 节 排序算法

一、内排序和外排序的定义

由于待排序的记录数量不同，使得排序过程中涉及的存储器不同，可将排序方法分为两大类内排序与外排序。

1.内排序：待排序记录存放在计算机内存中进行的排序过程。插入排序、快速排序、选择排序、归并排序、基数排序等目前竞赛研究的算法基本上都是内排序方法。

2.外排序：待排序记录的数量很大，以致于内存不能一次容纳全部记录，所以在排序过程中需要对外存进行访问的排序过程。

二、衡量效率的方法

1.内排序：比较次数.也就是时间复杂度。2.外排序：10 次数，也就是读写外存的次数。

三、排序稳定性

排序前后相同元素的相对位置不变，则称排序算法是稳定的，否则排序算法是不稳定的。如原序列 ri=rj 且 ri 位于 rj 之前，排序后 ri 仍在 rj 之前，则称该排序是稳定的。

四、常用排序算法复杂度

排序法	最坏时间	平均时间复杂度	稳定性	空间复杂度
插入排序	$O(n^2)$	$O(n^2)$	稳定	O(1)
选择排序	$O(n^2)$	$O(n^2)$	不稳定	O(1)
冒泡排序	$O(n^2)$	$O(n^2)$	稳定	O(1)
希尔排序	$O(n^s)$,1<s<2,取决于增量序列	$O(n\log_2 n)$	不稳定	O(1)
快速排序	$O(n^2)$	$O(n\log_2 n)$	不稳定	$O(\log_2 n)$
堆排序	O(n logn)	$O(n\log_2 n)$	不稳定	O(1)
归并排序	O(n logn)	$O(n\log_2 n)$	稳定	O(n)

课堂练习

1.【NOIP2011】体育课的铃声响了，同学们都陆续地奔向操场，按老师的要求从高到矮站成一排。每个同学按顺序来到操场时，都从排尾走向排头，找到第一个比自己高的同学，并站在他的后面。这种站队的方法类似于（）算法。

A.快速排序 B.插入排序 C.冒泡排序　D.归并排序

2.【NOIP2018】以下排序算法中，不需要进行关键字比较操作的算法是（ ）。

A.基数排序 B.冒泡排序 C.堆排序 D.直接插入排序

3.【NOIP2009】排序算法是稳定的意思是关键码相同的记录排序前后相对位置不发生改变，下列哪种排序算法是不稳定的（)。

A.冒泡排序 B.插入排序 C.归并排序 D.快速排序

4.【NOIP2009】快速排序最坏情况下的算法复杂度为（）

A.O(logn)　B.O(n)　C.O(nlogn)　D.O（n^2）

5.【NOIP2009】快速排序平均情况和最坏情况下的算法时间复杂度分别为（）

A.平均情况 O（nlogn），最坏情况 O（n^2）

B.平均情况 O（n），最坏情况 O（n^2）

C.平均情况 O（n），最坏情况 O（nlogn）

D.平均情况 O（logn），最坏情况 O（n^2）

6.【NOIP2012】如果不在快速排序中引入随机化，有可能导致的后果是（）

A.数组访问越界 B. 陷入死循环 C.排序结果错误 D.排序时间退化为平方级

7.【NOIP2010】基于比较的排序时间复杂度的下限是（ ），其中 n 是待排的元素个数。

A.O(n) B. O(nlogn) C.O(logn) D.O(n^2)

8.【NOIP2013】（ ）的平均时间复杂度为 O（nlogn），其中 n 是待排序的元素个数。

A.快速排序 B.插入排序 C.冒泡排序 D.基数排序

9.【NOIP2014】以下时间复杂度不是 O（m）的排序方法是 ().

A.插入排序 B.归并排序 C.冒泡排序 D.选择排序

10.【NOIP2017】对于给定的序列{a}, 我们把(i,j) 称为逆序对当且仅当 i<j 且 $a_i>a_j$。那么序列 1，7，2，3，5，4 的逆序对数为（）个。

A.4 B.5 C.6 D.7

11.【NOIP2012】使用冒泡排序对序列进行升序排序，每执行一次交换操作将会减少 1 个逆序对，因此序列 5，4，3，2，1 需要执行（ ）次交换操作，才能完成冒泡排序。

A.0 B.5 C.10 D.15

12.【NOIP2017】设 A 和 B 是两个长为 n 的有序数组，现在需要将 A 和 B 合并成一个排好序的数组, 任何以元素比较作为基本运算的归并算法在最坏情况下至少要做（）次比较。

A.n^2 B. nlogn C.2n D.2n-1

不定项选择题

1.【NOIP2009】排序算法是稳定的意思是关键码相同的记录排序前后相对位置不发生改变，下列排序算法稳定的是（)。

A.插入排序 B.基数排序 C.归并排序 D.冒泡排序

2.【NOIP2017】下列算法中，（）是稳定的排序算法。

A.快速排序 B.堆排序 C.希尔排序 D.插入排序

3.【NOIP2010】原地排序是指在排序过程中（除了存储待排序元素以外的）辅助空间的大小与数据规模无关的排序算法。以下属于原地排序的有（）

A.冒泡排序 B.插入排序 C.基数排序 D.选择排序

4.【NOIP2016】下列算法中运用分治思想的有（）

A.快速排序 B.归并排序 C.冒泡排序 D.计数排序

5.[NOIP2013]（）的平均时间复杂度为 O（nlogn）,其中 n 是待排序的元素个数。

A.快速排序 B.插入排序 C.冒泡排序 D.归并排序

6.【NOIP2017】以下排序算法在最坏情况下时间复杂度最优的有（）

A.冒泡排序 B.快速排序 C.归并排序 D. 堆排序

7.【NOIP2011】对于序列“7、5、1、9、3、6、8、4”，在不改变顺序的情况下，去掉（ ）会使逆序对的个数减少 3。

A.7 B.5 C.3 D.6