QQ:2305201452
呆@西西弗斯

C++复习提纲要点

笔试题型：选择题 20*2，填空题 15*1，阅读题 15*2，完善题 15*1

上机题型：编程 50 + 改错 50（改错题：错误位置固定，共五处。）

一、基础知识

1、标识符的规定、整形常量的规定

{合法标识符由字母或下划线开始，由字母、数字、下划线组成，且不能和关键词同名。}

2、基本类型的长度，单字符和字符串的区别

{char 1bit//short(int) 2bits//int 4bits //Long(int) 4bits//float 4bits//double 8bits}
{字符串常量"a"　占两个字节，存放'a'和'\0'，值为 0x6100
字符型常量'a'　占一个字节，存放'a'，值为 0x61}

3、基本运算符的单一和混合使用（逻辑非、复合赋值、自增、指针运算符）

{p66~76　表 2-5
基本规律：
（1）单目运算符优先级高
（2）算术运算符优先级高，关系运算符次之，逻辑运算符低，赋值运算符优先级更低。
（3）最高优先级为括号和成员运算符，最低运算符维度和运算符。}

4、基本输入输出、字符和字符串的输入输出

5、字符和整数的区别和转换

{字符型用来保存字符，存储的是该字符的 ASCII 码，占用一个字节。字符型数据从本质上说也是整数，可以是任何一个 8 位二进制整数。字符型数据强制转换成整型变量，将字符的 ASCII 码赋给整型变量。}

6、编写程序的基本环节

7、易错的知识点（'\0'和'0',赋值和判等符号、判断变量在二者当中、大小写转换）。

二、控制结构

1、for 循环当中三个表达式的理解

{for （表达式 1<循环变量赋初值>；表达式 2<循环条件>；表达式 3<循环变量增值>）
注意：1、for 语句中的三个表达式可以部分省略或全部省略，但“；”不能省略，若省略表达式 2,则表示循环条件恒为真。
2、for 语句中三个表达式可以是任何有效的 C 语言表达式。}

2、do-while 循环的特点

3、循环的阅读（注意循环条件的判断）、循环次数的判断

{三种循环形式的区别是：
① 循环变量和初值：while 和 do while 语句一般在进入循环前赋值，for 语句一般由 for 语句本身的表达式 1 赋值；
② 循环控制变量的变化：while 和 do while 语句中在循环体中变化，for 语句一般在本身的表达式 3 变化；
③ 循环条件的检测：while 和 for 语句在循环体前检测，这样有可能循环体一次也不执行，而 do while 循环在循环体后检测，这样循环至少要被执行一次。}

4、switch 语句的阅读（注意 break)

{ switch （ 表达式）
{case 常量表达式 1： 语句序列 1；break;
case 常量表达式 2：语句序列 2；break;
…
case 常量表达式 n：语句序列 n; break;
default： 语句序列 n+1；}

注意： 1、switch 与 if 不同，它仅能判断一种逻辑关系，即表达式是否等于指定的常量，而 if 可以计算并判断各种表达式。
2、case 子句后必须为常量，常常是整型和字符型。
3、default 可以省略，这时，不满足条件什么也不执行。
4、若 case 语句后没有 break，则执行完 case 语句后的语句序列后，继续向下执行其他的 case 分支后的语句序列，直到遇到 break 语句或开关语句的右花括号为止。

break 在 switch 语句中，可以使流程跳过判断体，执行下面的程序。在循环体中，也可以从循环体内跳出循环体，提前结束循环。break 只能退出一层循环(结束当前循环)或 switch 语句。}

三、函数

1、函数的嵌套

2、函数的返回值

{在 C++中，可以通过被调用函数返回值（没有返回值的函数必须说明为 void）的方法来将函数运行结果返回给调用函数。这个返回值有一个确定的类型（基本数据类型和各种构造数据类型），在函数定义时必须将返回值的类型写在函数名的前面。}

3、带参数的宏定义

{对带参数的宏，在调用中，不仅要宏展开，而且要用实参去代换形参。
带参宏定义的一般形式为：#define 宏名(形参列表) 字符串
#define M(y) y*y+3*y//宏定义
k=M(5); //宏调用
在宏调用时，用实参 5 去代替形参 y，经预处理宏展开后的语句为 k=5*5+3*5
注意：
1）、带参宏定义中，形参之间可以出现空格，但是宏名和形参列表之间不能有空格出现。
2）、在带参宏定义中不必指明数据类型。而在宏调用中必须指明数据类型。

3）、函数调用时要把实参表达式的值求出来再传递给形参，而宏展开中对实参表达式不作计算，直接按照原样替换。}

4、局部静态变量

{采用 static 修饰的局部变量，虽然具有全局生命期，但其可见性却是局部的，其作用域和自动变量相同。静态局部变量在其作用域内是可见的，也是存在的；当超出它的作用域后，虽然是不可见的，但它仍是存在的。这是静态变量的特点。当再次进入它的作用域时，变量仍使用为其分配的存储空间，因此这些变量仍保留上次运算的值。}

5、简单递归函数的阅读和定义

{递归函数的定义特点：

- 函数要直接或间接调用自身，
- 要有递归终止条件检查，即递归终止的条件被满足后，则不再调用自身函数。
- 如果不满足递归终止的条件，则调用涉及递归调用的表达式。在调用函数自身时，有关终止条件的参数要发生变化，而且需向递归终止的方向变化。}

四、数组与指针

1、一维数组的定义和使用

{数组是同类型有序数据的集合。数组中的数据称为数组元素，要引用某个数组元素必须给出两个要素，即数组名和下标(其中的下标只能为整型表达式或为字符型表达式)。
定义方法：<存储类型><类型说明符> <数组名>[常量表达式] 如：int a[10]; }

2、二维数组的初始化和使用

{定义：类型 数组名[常量表达式（行数）][常量表达式（列数）];如：int a[3][4];
初始化：1、分行赋值: int a[3][4]={{1,2,3,4}（第一行）,{5,6,7,8}（第二行）,{9,10,11,12}（第三行）};
2、顺序赋值：int a[3][4]={1,2,3,4,//给第 0 行 5,6,7,8,//给第 1 行 9,10,11,12//给第 2 行};
3、部分赋值：int a[3][4]={{1},{5},{9}}; //a[0][0]=1, a[1][0]=5, a[2][0]=9 其余元素为 0
int a[3][4]={{0,1},{5}}; // a[0][0]=0, a[0][1]=1, a[1][0]=5 其余元素为 0
4、分行或全部赋值时，可以省略第一维，第二维不可省。
5、不能给数组整体赋值。
6、用 static 定义的数组不赋初值，系统均默认其为 0。}

3、指针的运算（赋值、算术、比较）

{指针变量只能指向和其同类型的变量，不能指向其它类型的变量，也不能指向常量和表达式。C++允许将一个常量经强制类型转换后赋值给指针变量。

1、 **赋值运算**：1）把一个变量的地址赋予相同数据类型的指针变量 e.g.int a,*p; p=&a;
2） 把一个指针变量的值赋予指向相同类型变量的另一个指针变量 e.g.int a,*pa=&a,*pb; pb=pa;
3） 把数组的首地址赋予指向数组的指针变量 e.g.int a[5],*p; p=a;
4） 把字符串的首地址赋予指向字符类型的指针变量 e.g. char *p; p=”I Love You!”;

2、 指针**加减一个整数的算术运算**：一般的，在指针指向一个数组后，指针加减一个整数的算术运算才有意义。指针变量加减一个整数 n 的含义是将指针指向的当前元素向前或向后移动 n 个元素。
int a[5],*p; p=a//p 指向数组 a，也就是 a[0];p=p+2//向后移两位此时指向 a[2]
一般的，当有了定义“int a[10],*p=a;”后，有：
p=1;//通过指向运算符“”赋值 a[0]=1;//直接对数组元素赋值 p[0]=1;//指针名作数组名用
另（p234）：（*p）++;//相当于 a++（指针不移动） *p++; //首先*p，然后 p=p+1，指针下移一位

++*p//相当于++（*p），就是*p=*p+1（指针不移动）　　*++p//相当于*（++p），先指针下移一位后取内容

3、 两个指针变量之间的**算数运算**：两个指针变量之间只能做**减法**算数运算。只有指向同一数组的两个指针变量之间的减法才有意义。两指针相减所得之差是两个指针所指数组元素之间相差的元素个数。

4、 两个指针变量之间的**关系运算**：指向同一数组的两指针变量进行关系运算可表示他们所指数数组元素之间的关系。设有：int a[10],*p1=&a[0],*p2=a[5];

则有：p2>p1　　//值为真，p2 的值（地址）大于 p1 的值

P2==p1　　//值为假，p1、p2 值不同（指向的不是同一个地址）

P1！=0　　//值为真，p1 有指向不是空指针}

4、数组作为函数的参数（本质、实参和形参要求）

{分为两种情况：一种是数组元素作实参；一种是数组名作参数（形参和实参）。不存在数组元素作形参的情况。

一、 数组元素作函数参数：数组元素作函数实参，用法与一般变量作实参相同，是“值传递”。

二、 用数组名作函数参数：用数组名作函数参数，实参与形参都应用数组名。这时，函数传递的是数组在内存中的地址实参中的数组地址传到形参中，实参形参共用同一段内存。}

5、通过指针或字符指针访问数组的形式和方法

{指针与一维数组：一级指针与一维数组的关系十分密切。指向一维数组的指针变量，实际上是指向一维数组元素的指针变量。可以利用指向一维数组的指针变量，完成对数组数据的操作处理。

二维数组有三种地址：数组地址、行地址和元素地址。设有定义“int a[4][4];”，则三种地址为：

1）数组首地址：a

2）行地址：a+0、a+1、a+2 和 a+3

3）元素地址：a[i]或 a[i]+j 或&a[i][j]等

由同类型指针所组成的数组称为指针数组。或者当某个数组被定义为指针类型，就称这样的数组为指针数组。指针数组的每个元素都是一个指针变量。

定义指针数组的一般格式为：[<存储类型>] <数据类型> *<数组名>[<长度>]；

可以给指针数组赋初值，赋初值有多种方式，同普通的数组。定义指针变量时的"数据类型"可以选取任何基本数据类型，也可以是结构体类型、枚举类型或类类型。}

6、四个常用字符串函数及实现、其它常用字符串算法（倒序、大小写转换、字符数统计等）

{1. 字符串复制函数 strcpy

原型：strcpy(char to[ ], const char from[ ]);

功能：将字符串 from 拷贝到字符串 to 中。

```
char str1[10]={ "database"};
char str2[10]={ "Computer"};
cout <<strcpy(str1,str2)<<endl;
```

运行结果是: Computer

2. 字符串部分复制函数 strncpy

原型：strncpy(char to[ ], const char from[ ], int numchars);

功能：将字符串 from 中前 numchars 个字符拷贝到字符串 to 中。

```
strncpy(str1,str2, 4);
cout<<str1<<endl;
```

输出结果：Compbase

3. 字符串拼接

原型：strcat(char target[ ], const char source[ ]);

功能：将字符串 source 接到字符串 target 的后面。

strcat(str1,str2);
cout<<str1<<endl;
输出结果：C++language
4. 字符串部分拼接
原型：strncat(char target[ ], const char source[ ], int numchars);
功能：将字符串 source 的前 numchars 个字符接到字符串 target 的后面。
strncat(str1,str2,4);
cout<<str1<<endl;
输出结果： C++Lang
5. 字符串比较
原型：int strcmp(const char first[ ], const char second);
功能：比较两个字符串 first 和 second。如果 first 大于 second，函数值就为 1，如果两个字符串相同，函数值就为 0，如果 firet 小于 second，函数值就为-1。
cout<<strcmp(“abc”,” cba”)<< ‘\n’ ;
cout<<strcmp(“123”,” xyz”)<< ‘\n’ ;
输出：-1
1
6. 求字符串长度
原型：strlen(const char string[ ]);
功能：统计字符串 string 中字符的个数。
char a[ ]={“students”}
cout<<strlen(a)<< ‘\t’ <<sizeof(a)<< ‘\t’ ;
cout<< strlen(“String”)<< ‘\n’ ;
输出：8 9 6
strlen 函数的功能是计算字符串的实际长度，不包括‘\0’在内。sizeof 是运算符，计算的是数组长度，包括‘\0’。}

7、字符数组和指针的结合及区别，*p 和 p 的区别

{用字符指针处理字符串：

```
char *str=“I Love You”,*p=str;
cout<<str<<’ \n’ ;   //整体输出 str
while(*p)  cout<<*p++;  //逐个字符输出 str
cout<< ‘\n’ ;
for(int i=0;i<strlen(str);i++)  //输出 str
    cout<<*(str+i);}
```

8、动态内存分配和释放的概念

{用 new 运算符申请的堆空间没有名字，只能用指针指向其首地址或使用引用。在申请的空间释放以前，该指针不能再指向其它地址，以防内存泄露。当没有足够的堆空间用于分配时，new 运算符返回空指针 NULL。delete 运算符的功能是用来释放（删除）使用 new 在堆上申请的空间，将申请的空间归还给系统。

使用 new 在堆上申请空间

```
int *p1=new int; //p1 指向 new 申请的空间
float *p2=new float(3.14);//p2 指向 new 申请的空间，初始化为 3.14
char *p3=new char[20]; //用 p3 指向在堆上申请的字符数组
int *p4=new int[3*5];//用 p4 指向在堆上申请的 3 行 5 列的二维数组
```

```
int &ref=*new int;//申请堆空间来初始化引用
delete ptr; //释放指针 ptr 指向的堆空间
delete &ref;//释放引用堆空间时要加“&”号}
```

9、两种排序算法

{1、冒泡排序也称“两两比较法”，它通过对相邻的两数组元素的比较(a[i]和 a[i+1]比较或 a[i]和 a[i-1]比较)，来改变数据的位置，以达到排序的目的。

```
for (j=0; j<n-1; j++)
      for (i=0; i<n-1-j; i++)
      {     if (a[i]>a[i+1])
             {    t=a[i];
                  a[i]=a[i+1];
                  a[i+1]=t;
              }
      }
```

2、选择排序法也称“逐个比较法”，它将一个元素和其后的所有元素逐个比较(a[i]和 a[j] (j>i))，每一趟比较都是在找一个最小值，在一个下标区间内进行，完成后，就收缩左端点(下标加 1)，再重复找最小值并和左端点元素交换，直到区间只有一个数组元素为止。

```
          for(i=0;i<n-1;i++)    //趟循环，第 i 趟
              for(j=i+1;j<n;j++)//趟内比较，找最小值
      if(a[j]<a[i])             //交换 a[j]、a[i]
            {   temp=a[i];
                a[i]=a[j];
                a[j]=temp;
            }
```

注：对于冒泡排序，就是 a[i]的 i+1 从某个值开始，直到元素的最大下标，对于选择排序，就是 a[j]的 j 从某个值开始，直到元素的最大下标}

五、结构体、类与文件

1、结构体类型与变量

{结构体变量必须在定义了结构体类型的基础上才能定义。

```
struct <结构体类型名>
{
          <类型名><变量 1>;
          <类型名><变量 2>; …
};
```

定义结构体变量有三种方式：

1、先定义结构体类型，后定义结构体变量。这是使用最多的方式；

```
struct   stu
{    int  num;
     char  name[20];
     char  sex;
};
```

2、在定义结构体类型的同时定义结构体变量。这种方式也常使用；

```
struct   stu
{    int  num;
     char  name[20];
     char  sex;
}s1,s2;
```

```
stu s1,s2;
```

3、定义结构体类型时省略类型名，直接定义结构体变量。这种方式只能一次性的定义变量，使用得较少。

```
struct
{     int num;
      char name[20];
}s1,s2;
```

例：结构体变量的初始化：

```
struct stu
{   int num;
  char name[20];
  char sex;
};
stu s1={101,”马犇”,’W’},
  s2={102,”杨森”,’M’};
```

注意：结构体类型的变量在内存依照其成员的顺序顺序排列，所占内存空间的大小是其全体成员所占空间的总和。

结构体变量不能整体赋值，但同种类型的结构体变量之间可以相互赋值。

结构体变量的引用主要是输入/输出、赋值。多数情况下只能引用结构体变量的成员，而不能直接引用结构体变量。

结构体变量成员的使用与一般变量的使用相同。使用的格式为：结构体变量名.成员名

例：引用结构体变量成员。

```
struct book
{    long shuh;          //书号
     char shum[50];      //书名
     char zuoz[20];      //作者
   }b1,b2;
   b1.shuh=100001;       //为书号赋值
   strcpy(b1.shum,”从0到无穷大”);  //为书名赋值
   cin.getline(b1.zuoz,20); //输入作者
}
```

2、常见类的定义（复数、学生、矩形等）、权限限定符、类外定义函数、对象的创建

{{类的定义格式：

```
class  类名
{    private :
          成员数据;
          成员函数;
     public :
          成员数据;
          成员函数;
     protected
          成员数据;
          成员函数;
};
```

```
class  Student
{     private :
          char Name[20];
          float Math;
          float Chiese;
      public :
          float  average;
          void SetN(char *name); //成员函数
          void SetSc(float ,float); //成员函数
          float GetSc(float&, float&);//成员函数
 };
```

用关键字 private 限定的成员称为私有成员，对私有成员限定在该类的内部使用，即只允许该类中的成员函数使用私有的成员。类就相当于私有成员的作用域。

用关键字 public 限定的成员称为公有成员，公有成员的数据或函数不受类的限制，可以在类内或类外自由使用；而用关键字 protected 所限定的成员称为保护成员，只允许在类内及该类的派生类中使用保护的数据或函数。即保护成员的作用域是该类及该类的派生类。

注意：1、定义类时不可以初始化。2、类体内首行如果是 private，则可以省略。

在类内定义构造函数的一般格式为：ClassName(<形参表>)　　{函数体…}

在类外定义构造函数的一般格式为：ClassName::ClassName(<形参表>)　　{函数体…}

类与对象的关系就如同结构体类型与结构体变量的关系一样，类的对象是具有该类类型的某一特定实体。类是用户定义的一种类型，程序设计中可以使用这种类型名来说明变量，类类型的变量称为对象，又称为类的实例。}

3、类的构造和析构函数

构造函数有以下特点：

1. 函数名特点。构造函数的函数名必须与类名相同。构造函数的主要作用是完成初始化对象的数据成员以及其它的初始化工作。
2. 返回值特点。在定义构造函数时，不能指定函数返回值的类型，也不能指定为 void 类型。
3. 重载特点。一个类可以定义若干个构造函数。当定义多个构造函数时，必须满足函数重载的原则。
4. 缺省值特点。构造函数可以指定参数的缺省值。例子 p278

析构函数的作用与构造函数正好相反，是在对象的生命期结束时，释放系统对象所分配的空间前，做一些善后工作。

析构函数也是类的成员函数，定义析构函数的格式为：

```
ClassName::~ClassName( )
{
    ......      //函数体
}
```

关于析构函数，有几点说明：

1、析构函数是一个特殊的函数，它的名字同类名，并在前面加“~”字符，用来与构造函数加以区别。析构函数不指定数据类型，并且也没有参数。

2、一个类中只可能定义一个析构函数。

3、析构函数是成员函数，函数体可写在类体内，也可写在类体外。

4、同构造函数一样，如果没有为类定义析构函数，系统会自动为类添加一个默认的析构函数，函数体为空。

5、析构函数一般不显式调用，它由系统自动调用。一般有两种情况：

A、如果一个对象是局部的，它具有块作用域，当对象所在的块结束时，该对象的析构函数被自动调用。

B、全局对象和静态对象，程序运行结束时，该对象的析构函数被自动调用。

6、当一个对象是使用 new 运算符被动态创建的，在使用 delete 运算符释放它时，delete 将会自动调用析构函数。如果不使用 delete 运算符释放，即使到了程序结束，也不会自动调用析构函数释放它}

4、枚举常量

{枚举类型是一种用户定义的数据类型，

其一般定义形式为：

```
enum [枚举类型名]
{
  标识符[=整型常数],
  标识符[=整型常数],
  ...
  标识符[=整型常数],
};
```

提示：在输出枚举变量时，输出的是枚举成员的值。如果想要输出枚举成员本身，则需要输出枚举成员对应的字符串常量。
}

5、基本输入输出流对象

{例：把 1～100 的所有数据存放到文件 data1.dat 文件中。

```
#include<fstream.h>
void main(void)
{   fstream outfile("data1.dat",ios::out);
    //ofstream outfile("data1.dat");//推荐使用
    //fstream outfile;
    //outfile.open("data1.dat",ios::out);
    for(int i=1;i<=100;i++)
        outfile<<i<<" ";
    outfile.close();//注意关闭文件！
}
```

例。把上例中存放在文件 data1.dat 中的 1～100 所有数据读出并显示在屏幕上。

```
#include<fstream.h>
#include<iostream.h>
void main(void)
{   fstream infile("data1.dat",ios::in);
    //ifstream infile("data1.dat");
    //fstream infile;//推荐使用
    //infile.open("data1.dat",ios::in);
      int a,i;
        for(i=1;i<=100;i++)
        {infile>>a;          //A
        cout<<a<<"\t";  //B     }
        infile.close();//注意文件关闭！
}
}
```

六、常见算法

1、交换算法函数的多种形式（通过或不通过函数）

2、完全数算法

{完数即所以因数和等于其本身的数。

```
void main()
{       int i,j,sum;
    for(i=1;i<=1000;i++)
    {     sum=0;
         for(j=1;j<=i/2;j++)
            if(i%j==0)
            sum+=j;
```

```
        if(sum==i)
           cout<<i<<endl;
    }
}
}
```

3、斐波那契数列算法

{

数组算法	函数算法	迭代算法

void main (void)
{ int i;
int f [20]={1,1};
for (i=2 ; i<20 ; i++)
f [i]=f [i-1]+f [i-2];
for (i=0; i<20; i++)
{ if (i%5= =0)
cout<< “\n” ;
cout<<f [i]<< ‘\t’ ;
}
}
	#include <iostream.h>
	int fib(int n)
	{
	if((n==1)||(n==2))
	return 1;
	return fib(n-1)+fib(n-2);
	}
	void main()
	{
	for(int i=1;i<=20;i++)
	{ cout<<fib(i)<<’\t’;
	if(i%5==0)
	cout<<’\n’;
	}
	}
		long int f1,f2,f3;
		int i;
		f1=1;f2=1;
		cout<<setw(10)<<f1<<setw(10)<<f2;
		for (i=3;i<=20;i++)
		{
		f3=f1+f2;
		cout<<setw(10)<<f3;
		f1=f2;
		f2=f3;
		if (i%5==0)
		cout<<’\n’;
		}
		cout<<’\n’;

}

4、进制转换算法（除基取余）

{ 十进制转换成二进制：

```
{   int N;
    cout<<"请输入一个数"<<endl;
    int a[20];
    cin>>N;
    int num;
    int i=0;
    while(N !=0)
    {
        num = N%2;
        a[i] = num;
        i++;
        N = N/2;
    }
    for(num=i-1;num>=0;num- -)
        cout<<a[num];
```

}
}

5、穷举和递推的思路

{枚举法也称穷举法，基本思想是，在有限范围内列举所有可能的结果，找出其中符合要求的解。
迭代与递推算法是通过问题的一个或多个已知的解，用同样的方法逐个推算出其他的解。递推法适用的问题包括数列问题、近似计算问题等。通常也采用循环结构。 }