C++程序设计（蓝书）课后习题参考答案1.0

——东大成贤计算机协会C++书后答案编写组

编者的话

C++是编程者入门首选的语言之一，其功能强大，不仅兼容C语言面向过程的编程思想，也具有先进的面向对象编程特点，能够学习并掌握它的基本用法对于打好计算机基础有着深远的意义。

学校使用的C++程序与设计一书（该书封皮为蓝色，下文皆用蓝书来代指）的课后习题没有答案，此前大家做了题并不知道自己所做是否正确，若想知晓其正确性需要去网上一道题一道题地搜索或者将代码敲到本地编译环境运行验证，历届学生对此都深感不便。为了方便大家对C++能够更好地理解掌握，也对临近的C++期末考试更有把握，我们自发整理了一份蓝书的课后习题参考答案和大家分享。

该答案的内容均由学生制作，内容包括**第一章到第九章**，与期末考试范围吻合，不过我们**不能保证其正确性**，若对答案有疑问或者有其他建议，可以通过QQ邮箱（1798928638@qq.com）的方式联系我，也**欢迎进入我们协会的新生咨询群批评指正**，对于发现的错误我们会尽量及时进行修改再版并对外发布。**用电脑看比较方便，对应章节和题目都有书签。**

我们是一个因对学习计算机知识感兴趣而自发组织起来的学生社团，欢迎对计算机同样感兴趣的人加入我们，这是我们新生咨询群聊QQ号：**333966455**

致谢

此次课后习题答案的组织能够及时地在期末复习前完成并发布，要感谢负责答案编辑的卢屹寒，还要感谢制作答案内容的赵秋荻、王子铭、江谨耀、赖原玮、吴子昂、沙靖超以及计协卓越计科小组中参与此次活动的成员们，感谢你们的充分理解、积极配合与辛勤付出。

第一章 C++概述

1.A

2.C

3.B

4.

> （1）在C语言的基础上进行扩充和完善，使C++兼容了C语言的面向过程特点，又成为了一种面向对象的程序设计语言
> （2）可以使用抽象数据类型进行基于对象的编程
> （3）可以使用多继承、多态进行面向对象的编程
> （4）可以担负起以模版为特征的泛型化编程

5.

```
#include<iostream>

using namespace std;

int main()

{

cout<<"Hello, C++!"<<endl;

return 0;

}
```

第二章

1.A

2.D

3.A

4.D

5.D

6.A

7.A

8.A

9.1.25

10. 9

11.e

12.

(1).

```
#include<iostream>
using namespace std;
int main()
{
    int a, b, c;
    cin >> a >> b >> c;
      if (a > b)
      {
        if (a > c)
        {
          cout << "max= "<<a << endl;
```

```
        }
        else
        {
          cout <<"max= " << c << endl;
        }
     }
  else
  {
    if (b > c)
    {
      cout << "max= "<<b << endl;
    }
    else
    {
      cout << "max= "<<c << endl;
    }
 }
```

(2).

```
  for (int j = 0;j < 100;j++)

   {

    cout << j << endl;

   }
```

(3).

```
  float a, b, h,y;

   cin >> a >> b >> h;

   y = (a + b) * h / 2.0;
```

(4).

```
int year;

   cin >> year;

   if (year % 4 == 0 && year % 100 != 0)

   {

     cout << "该年是闰年" << endl;

   }

   if (year % 400 == 0)
```

```
    {
      cout << "该年是闰年" << endl;
    }
```

13.

0,0

14.

3,7,13,19,29

15.

```
      1
     2 2
    3 3 3
   4 4 4 4
  5 5 5 5 5
```

16.

C++

17.

x>y u>z

18.

i<=20
if(i>0)i—
b+=i

19.

i<=9
j%7==0
cout<<j<<endl

20.

```
f1+f2
i%5==0
f1=f
```

21.

```
n=num
n
n=n/10
```

22.

```
cin>>num
num%i==0
break
```

23.

```
//357整除
#include<iostream>
using namespace std;
int main(){
    int n;
    cin>>n;
    if(n%3==0&&n%5==0&&n%7==0)
    cout<<"yes";
    else cout<<"no";
    return 0;
}
```

24.

```
/判断该月有多少天
#include<iostream>
using namespace std;
int main(){
    int a,b;
    cin>>a>>b;
    if(a%100!=0&&a%4==0){
        if(b=2) cout<<"该月有29天";
    else if(b=1||b==3||b==5||b==7||b==8||b==10||b==12)
    cout<<"该月有31天";
    else cout<<"该月有30天" ; }
    else if(a%400==0){if(b=2) cout<<"该月有29天";
    else if(b=1||b==3||b==5||b==7||b==8||b==10||b==12)
    cout<<"该月有31天";
    else cout<<"该月有30天" ;


    }
    else {
```

```
    if(b=2) cout<<"该月有28天";
        else if(b=1||b==3||b==5||b==7||b==8||b==10||b==12)

{
        cout<<"该月有31天";}
        else cout<<"该月有30天" ;
    }


}
```

25.

```
//最小公约数
#include<iostream>
using namespace std;
int main(){
    int a,b,temp,r;
    cin>>a>>b;
    if(b>a){
    temp=a;
     a=b;
    b=temp;


    }
    while(b!=0){
        r=a%b;
        a=b;
        b=r;

    }
    cout<<"最大公约数为: "<<a;

}
```

26.

```
//计算e
#include<iostream>
#include<iomanip>
using namespace std;
int main(){
    int n;
    double j=1,z=1;
    cin>>n;
    for(int i=1;i<=n;i++){
        z=z*i;
        j=1.0/z+j;
    }
    cout<<setprecision(n)<<j;
```

```
}
```

27.

```
//计算e
#include<iostream>
using namespace std;
int main(){
    int n;
    cin>>n;
 for(int i=1;i<=n;i++){
    for(int j=1;j<=n-i;j++)
    cout<<" ";
    for(int k=1;k<=2*i-1;k++){
        cout<<"*";
    }
    cout<<endl;
}
    return 0;

}
```

28.

```
//计算e
#include<iostream>
using namespace std;
int main(){
    int n;
    cin>>n;
 for(int i=1;i<=n;i++){
    for(int j=1;j<=n-i;j++)
    cout<<" ";
    for(int k=1;k<=2*i-1;k++){
        cout<<"*";
    }
    cout<<endl;
}
    return 0;

}
```

29.

```
//8的阶乘
#include<iostream>
#include<string.h>
using namespace std;
int main(){
```

```
    int n,s=0,m=1;
    cin>>n;
    for(int i=1;i<=n;i++) {
     m=m*i;
     s=s+m;

    }
     cout<<s;
     return 0;
}
```

30.

```
//水仙花数
#include<iostream>
using namespace std;
int main(){
     int a,b,c;
     for(int i=100;i<=999;i++){
        c=i%10;
        b=(i-c)/10%10;
        a=i/100;
        if(i==a*a*a+b*b*b+c*c*c) cout<<i<<endl;
        else continue;
     }
     return 0;
}
```

31.

```
#include<iostream>
using namespace std;

int main()
{
    int sum = 1;//总数
    int num = 1;//每天吃的
    for (int i = 9; i >=1; i--)//第十天没吃，因此从第九天倒退回去
    {
        num = 2 * (num + 1);
        sum += num;
    }

    cout << sum;
    return 0;

}
```

32.

```
//数列
#include<iostream>
using namespace std;
int main(){
     int arr[21];
     for(int i=3;i<=20;i++)

 {

      arr[1]=1;
      arr[2]=1;
      arr[i]=arr[i-1]+arr[i-2];}

 for(int i=1;i<=20;i++)
 {
    cout<<arr[i]<<"\t";
 }
 return 0;
}
```

33.

```
//逆向输出
#include<stdio.h>
int main(){
    int a,b,c;
    scanf("%1d%1d%1d",&a,&b,&c);
    printf("\t");
    printf("%1d%1d%1d",c,b,a) ;

}
```

34.

```
//平均数，正数负数个数
#include<iostream>
using namespace std;
int main(){
    int sum=0,s=0,j=0,i=0,a;
    while(cin>>a){
        i++;
        if(a==0) break;
        else if(a>0) s++;
        else  j++;
        sum=sum+a;
    }
cout<<"正数个数为：<<s<<endl;
cout<<"负数个数为：<<j<<endl;
cout<<"平均数为：<< sum/(i-1);
```

```
return 0;

}
```

35.

```
//谁是候选人
#include<iostream>
#include<algorithm>
using namespace std;
int main(){
   int a,j=0,f=0,c=0,p=0,num=0;
   while(cin>>a){
    if(a==1) j++;
    else if(a==2) f++;
    else if(a==3) c++;
    else if(a==4) p++;
    else if(a==-1) break;
    else false;
    num++;
   }
   cout<<"Jerry:"<<j<<endl;
   cout<<"Flora:"<<f<<endl;
   cout<<"Candy:"<<c<<endl;
   cout<<"Paul:"<<p<<endl;
   if(j>num/2) cout<<"获胜者为Jerry";
   else if(f>num/2) cout<<"获胜者为Flora";
   else if(c>num/2) cout<<"获胜者为Candy";
   else if(p>num/2) cout<<"获胜者为Paul";
   else false;
}
```

第三章

1.C

2.A

3.C

4.C

5.C

6.B

7.A

8.C

9.ABC

10.A

11.C

12.类型 个数

13.递归

14.135

15.n=6

16.0 10

17.double 1

18.300 400 300 400

19.9

20.433 436 437

21.

```
#include<iostream>
#include<iomanip>
#include<cmath>
using namespace std;
int Distance(int x, int y, int x1, int y1)
{
return(sqrt((x1 - x) * (x1 - x) - (y1 - y) * (y1 - y)));
}
float Distance(float a, float b, float a1, float b1)
{
return (sqrt((a1 - a) * (a1 - a) - (b1 - b) * (b1 - b)));
}
int main()
{
int s, y, s1, y1;
cout << "请输入两点坐标" << endl;
cin >> s >> y >> s1 >> y1;
cout << "距离" << Distance(s, y, s1, y1) << endl;
float e, f, e1, f1;
cout << "请输入两点坐标" << endl;
cin >> e >> f >> e1 >> f1;
cout << "距离" << Distance(e, f, e1, f1) << endl;
return 0;
}
```

22.

非递归：

```
#include<iostream>
#include<iomanip>
#include<cmath>
using namespace std;
int main()
{
    long f1, f2;
    f1 = 1;
    f2 = 1;
    for (int i = 1; i <= 10; i++)
    {
        cout << f1 << " " << f2<<" ";
        f1 = f1 + f2;//第三个数
        f2 = f1 + f2;//第四个数


    }
    cout << endl;
    return 0;

}
```

递归：

```
int f(int n)
{
   if (n == 1 || n == 2)
        return (1);
    else
        return f(n - 1) + f(n - 2);
}
int main()
{
    int i;
    for (i = 1; i <= 10; i++)
    {
        cout << ” ” << f(i);
    }
}
```

23.

```
#include <iostream>
using namespace std;
 float p( float , float );
 int main()
{
float n,x;
 cin>> n >>x;
 cout<<p ( n, x )<<endl;
 return 0;
```

```
 }
float p( float n, float x )
 {
float f;
 if( n==0 )
{
   f=1;
   }
 else if( n==1 )
{
    f=x;
    }
 else if( n>=1 )
   f=((2*n-1)*x-p((n-1),x)-(n-1)*p((n-2),x))/2;
 return f;
 }
```

24.

```
//带参数的宏：
#include <iostream>
using namespace std;
#include <cmath>
#define S (a+b+c)/2
#define AREA sqrt(S*(S-a)*(S-b)*(S-c))
int main()
{
    float a, b, c;
    cin >> a >> b >> c;
    cout << AREA;
    return 0;
}
```

函数：

```
#include <iostream>
#include <cmath>
using namespace std;
float area1(float a,float b,float c)
{
    float s,p,area1;
    s=(a+b+c)/2;
    p=s*(s-a)*(s-b)*(s-c);
    area1=sqrt(p);
    return (area1);
}
int main()
   {
    float a, b, c, area;
    cin >> a >> b >> c;
    area= area1(a, b, c);
    cout << area;
    return 0;
}
```

25.

```
#include<iostream>
using namespace std;
double a[15];
double countvalue(int n)
{
    double sum = 0;
    for (int i = 1; i < n; i++)
    {
        if (i % 3 == 0 || i % 7 == 0)
        {
            sum += 1.0 * i;
        }
    }
    return sqrt(sum);
}
int main()
{
    int n;
    for (int i = 1; i <= 10; i++)
    {
        cin >> n;
        a[i] = countvalue(n);
        cout << a[i] << endl;
    }
    return 0;
}
```

26.

```
#include <iostream>
using namespace std;
#define N 5
int Max(int a, int b, int c)
{
    int m;
    if (a > b)
    {
        m = a;
    }
    else
    {
        m = b;
    }
    return (m > c ? m : c);
}
int main()
{
    int date[4][N];
    int ans;
    for (int i = 0; i < 3; i++)
    {
        for (int j = 0; j < N; j++)
```

```
        {
            cin>> date[i][j];
        }
        cout << endl;
    }
    for (int i = 0; i < N; i++)
    {
        ans = Max(date[0][i], date[1][i], date[2][i]);
        date[3][i] = ans;
    }
    for (int i = 0; i < 4; i++)
    {
        for (int j = 0; j < N; j++)
        {
            cout << date[i][j] <<"  ";
        }
        cout << endl;
    }
    return 0;
}
```

第四章

1.B

A : p指向a数组首地址，p[i]效果等同a[i]

B : * (a+i)指向a[i]存的那个值,假设i=1,则* (a+i)等于a[1]=1,再求1所指向的地址的值,无法求值

C： a[p-a]=a[0] 因为p指向a[0],地址是一样的,相减得0,即指向a[0]

D : &a[i]是a[i]这个位置的地址,*(&a[i])即把这个地址中存放数值取出来,即等于a[i]

2.D

p [1] [2]为a [1] [2]的值，其他均为 & a[1] [2]

3.C

p为int* 类型，共占4字节；int占4字节

4.D

*(p+2)与p[2]和 str [2]同结果

5.D

s是一个指针数组，s[0]指向“Student”**的首地址**，以此类推;ps是一个指针，初始化后指向"Father"**的首地址**。按字符串输出规则，* s[1]的值为"Teacher",ps的值为"Father"的首个字母"F"，* ps的值为"Father"

6.B

函数需要的参数为一个int类型和一个char*类型，x为int类型，数组名s为char *类型

7.A

略

8.26 14

9.10

10. 7

11. 1711717

12. s+n-1

p1<p2

p2--

13.p=q

*q=temp

14. char*p,int n

temp

p[j],p[j+1]

p[j]=p[j+1]

15.

```
#include<iostream>
#include<cmath>
#include<iomanip>
#include<string>
#include<algorithm>

using namespace std;



int main() {

  float a[10] = { 0.1 + 0.2 + 0.3 + 0.4 + 0.5 + 0.6 + 0.7 + 0.8 + 0.9 + 1.0 };

  float *p = a;

  int sum = 0;

  float sum = 0;
```

```
  for (int i = 0; i < 10; i++)
  {
    sum += *p;
    p++;
  }
  cout << sum;
  return 0;
}
```

16.

```
#include <iostream>
#include <stdio.h>
#include <stdlib.h>
using namespace std;
int a[5][5];

void transpose(int (*p)[5]) {
  for (int i = 0; i < 5; i++) {
    for (int j = 0; j < i; j++) {
      //( '-')< 注意此处 j < i
      //( 'o')< 若改为 j < 5 则会让已交换的值再次交换
      int tmp = *(a[i] + j);
      *(a[i] + j) = *(a[j] + i);
      *(a[j] + i) = tmp;
    }
  }
}
```

```
int main() {
  for (int i = 0; i < 5; i++) {
    for (int j = 0; j < 5; j++) {
      a[i][j] = rand();
      printf("%5d ", a[i][j]);
      //( '-')< %5d 意为输出长度固定为5的int值
      //( '-')< 仅增加输出数据的可读性 下同
    }
    printf("\n");
  }

  printf("\n");

  transpose(a);

  //( '-')< 循环打印
  for (int i = 0; i < 5; i++) {
    for (int j = 0; j < 5; j++) {
      printf("%5d ", a[i][j]);
    }
    printf("\n");
  }
  return 0;
}
#
```

17.

```
#include<iostream>
#include<cmath>
```

```
#include<iomanip>
#include<string>
#include<algorithm>
using namespace std;


int main() {
  string x, y;
  cin >> x >> y;
  int num1 = x.length();
  int num2 = y.length();
  int sum = 0;
  if (num1 > num2)
  {
    cout << x;
  }
  else if (num2 > num1)
  {
    cout << y;
  }
  else
  {
    for (int i = 0; i < num1; i++)
    {
      if (x[i] < y[i])
      {
        cout << y;
        break;
      }
      else if (x[i] > y[i])
```

```
      {
        cout << x;
        break;
      }
      sum++;
    }
  }
  if (sum == num1)
  {
    cout << "一样大";
  }
  return 0;
}
```

18.

```
#include<iostream>
#include<cmath>
#include<iomanip>
#include<string>
#include<algorithm>

using namespace std;

int main() {
  char s[10], a[10]; int i, j, f, k, g, b;
  f = 1;
  cin >> s;


  j = strlen(s);
  for (i = 0; i < j; i++)
    if (s[i] == '.')
    {
```

```
      f = 0; break;
    }
  if (f == 0)
  {
    int p;
    for (p = 0; p < i; p++)
      a[p] = s[p];
    a[p] = '\0';
  }
  else
  {
    int h;
    for (h = 0; h <= j; h++)
    {
      a[h] = s[h];
    }
    a[h] = '\0';
  }
  k = strlen(a);
  g = k;
  i = 0;
  while (a[i] != '\0')
  {
    b = k;
    switch (a[i])
    {
    case '1':
    {cout << "壹"; break; }
    case '2':
```

```
{cout << "贰"; break; }
case '3':
{cout << "叁"; break; }
case '4':
{cout << "肆"; break; }
case '5':
{cout << "伍"; break; }
case '6':
{cout << "陆"; break; }
case '7':
{cout << "柒"; break; }
case '8':
{cout << "捌"; break; }
case '9':
{cout << "玖"; break; }
case '0':
{cout << "零"; k = 0; break; }
}
switch (k)
{
case 1:
{break; }
case 2:
{cout << "拾"; break; }
case 3:
{cout << "百"; break; }
case 4:
{cout << "千"; break; }
case 5:
```

```
    {cout << "万"; break; }
    }
    k = b;
    k--;
    i++;
  }
  cout << "元";
  if (f == 0)
  {
    switch (s[g + 1])
    {
    case '1':
    {cout << "壹"; break; }
    case '2':
    {cout << "贰"; break; }
    case '3':
    {cout << "叁"; break; }
    case '4':
    {cout << "肆"; break; }
    case '5':
    {cout << "伍"; break; }
    case '6':
    {cout << "陆"; break; }
    case '7':
    {cout << "柒"; break; }
    case '8':
    {cout << "捌"; break; }
    case '9':
    {cout << "玖"; break; }
```

```
    case '0':
    {cout << " 零"; break; }
    }
    cout << "毛";
    switch (s[g + 2])
    {
    case '1':
    {cout << "壹"; break; }
    case '2':
    {cout << "贰"; break; }
    case '3':
    {cout << "叁"; break; }
    case '4':
    {cout << "肆"; break; }
    case '5':
    {cout << "伍"; break; }
    case '6':
    {cout << "陆"; break; }
    case '7':
    {cout << "柒"; break; }
    case '8':
    {cout << "捌"; break; }
    case '9':
    {cout << "玖"; break; }
    case 0:
    {cout << " 零"; break; }
    }
    cout << "分";
  }
```

```
  cin >> i;

  return 0;

}
```

19

```
#include <iostream>

#include <string.h>

using namespace std;



//( '-')< 这是用string类型来交换两个字符串

void swap_str(string &s1, string &s2) {

  string tmp = s1;

  s1 = s2;

  s2 = tmp;

}



//( '-')< 这是用字符数组来交换字符串

void swap_char(char *ch1, char *ch2) {

  char ch_tmp[100];

  char ch_zero[] = "\0";



  //储存ch1

  strcpy(ch_tmp, ch1);



  //清空ch1, 并把ch2赋给ch1

  strcpy(ch1, ch_zero);

  strcpy(ch1, ch2);



  //同理
```

```
  strcpy(ch2, ch_zero);

  strcpy(ch2, ch_tmp);

}


//( 'o')< 请注意string不是c语言内存在的类型 使用printf和scanf可能会出现难以预估的错误

int main() {


// string s1 = "Hello";

// string s2 = "World";

// cout << s1 << " " << s2 << endl;

// swap_str(s1, s2);

// cout << s1 << " " << s2 << endl;


  char ch1[100], ch2[100];

  cin >> ch1 >> ch2;

  swap_char(ch1, ch2);

  cout << ch1 << ch2;

  return 0;

}
```

20

```
#include<iostream>
using namespace std;
struct node
{
    node *next;
    int data;
}*head = new node;
void traverse()
{
    node *temp = head->next;
    cout << "遍历结果为：";
    while (temp)
    {
        cout << temp->data << " ";
        temp = temp->next;
    }
```

```
    cout << endl;
}
int len()
{
    return head->data;
}
void appen(int a)
{
    node *temp = head->next, *front = head;
    node *ne = new node;
    ne->data = a;
    while (temp && temp->data < a)
    {
        front = temp;
        temp = temp->next;
    }
    ne->next = temp;
    front->next = ne;
    head->data++;
}
void delet(int pos)
{
    if (pos > len() - 1)
    {
        cout << "超出链表范围" << endl;
    }
    else
    {
        node *temp = head->next, *front = head;
        for (int i = 1; i < pos; i++)
        {
            front = temp;
            temp = temp->next;
        }
        front->next = temp->next;
        head->data--;
        delete temp;
    }
}
int main()
{
    /*
    测试样例：
    5
    1 3 2 1 0
    3
    */
    head->next = NULL;
    head->data = 0;
    int n;
    cout << "请输入链表的初始长度" << endl;
    cin >> n;
    cout << "请插入链表的各个元素" << endl;
    for (int i = 1; i <= n; i++)
    {
        int t;
        cin >> t;
        appen(t);
```

```
    }
    cout << "链表长度为：" << len() << endl;
    traverse();
    cout << "请输入待删除的元素位置" << endl;
    cin >> n;
    delet(n);
    cout << "链表长度为：" << len() << endl;
    traverse();
}
```

第五章 类和对象

1. 下列关于类的定义格式描述中，错误的是__

A.类中成员有三种访问权限

B.类的定义可分说明部分和实现部分

C.类中成员函数都是公有的，数据成员都是私有的

D.定义类的关键字通常用class

解析：

A：正确

private 私有成员 仅允许本类的成员函数访问

public 公有成员 可以被任何函数访问

protected 保护成员 一般情况下与私有成员的含义相同。

区别在于：private完全私有，派生类不可访问，protected派生类可以访问。

B：正确。

C：错误。是否为公私有在于访问控制的关键字。

D：正确。

故选 **C** 选项

2.下列关于对象的描述中，错误的是__

A.定义对象时系统会自动进行初始化

B.对象成员的表示与C语言中结构变量的成员表示相同

C.属于同一个类的对象占有内存字节数相同

D.一个类所能创建对象的个数是有限制的

解析：

A：正确

B：正确 都用 '.' 或 '->' 进行访问

C：正确 数据类型相同，开辟的内存空间自然也相同

D：错误

故选 **D** 选项

3. 下列关于成员函数的描述中，错误的是

A.成员函数的定义必须在类体外

B.成员函数可以是公有的，也可以是私有的

C.成员函数在类体外定义时，前加inline可为内联函数

D.成员函数可以设置参数的默认值

解析：

A：错误 成员函数可以定义在类体外和类体内

B：正确 public 公有成员，private 私有成员

C：正确

D：正确 例如 void func(int parm = 10);

故选 **A** 选项

4. 关于构造函数，以下正确的说法是_

A.定义类的成员时，必须定义构造函数，因为创建对象时，系统必定要调用构造函数

B.构造函数没有返回值，因为系统隐含指定它的返回值类型为void

C.无参构造函数和参数为缺省值的构造函数符合重载规则，因此一个类中可以含有这两种构造函数

D.对象一经说明，首先调用构造函数，如果类中没有定义构造函数，系统会自动产生一个不做任何操作的缺省构造函数

A：错误 未定义构造函数，系统默认调用缺省构造函数(一个什么都不做的空函数)来完成初始化

B：错误 根本没有返回值，更不是void

C：错误 无参构造函数和参数为缺省值的构造函数在形式上均为：

类名 () {} 在形式上重复，系统无法区分，发生报错。

D：正确

故选 **D** 选项

5. 关于析构函数，以下说法正确的是_

A.析构函数与构造函数的唯一区别是函数名前加波浪线~，因此，析构函数也可以重载

B.当对象调用了构造函数之后，立即调用析构函数

C.定义类时可以不说明析构函数，此时系统会自动产生一个缺省的析构函数

D.类中定义了构造函数，就必须定义析构函数，否则程序不完整，系统无法撤销对象

解析：

A：错误 析构函数唯一且不可重载

B：错误 析构函数在类结束时调用

C：正确

D：错误 构造函数和析构函数都可以不定义，系统会默认提供缺省函数进行调用

故选 **C** 选项

6. 执行以下程序后，输出结果依次是_

```
class{
  int x;
  public:
  test(int a){
x=a;
cout<<x<<" 构造函数";
  }
  ~test(){
cout<<x<<" 析构函数";
  }
};

void main(){
  test x(1);
  x=5;
}
```

A. 1 构造函数 1 构造函数 5 析构函数 5 析构函数

B. 1 构造函数 5 构造函数 5 析构函数 5 析构函数

C. 1 构造函数 5 析构函数 5 析构函数 5 析构函数

D. 1 构造函数 1 析构函数 5 析构函数 5 析构函数

解析：

经运行 结果为 **B**

7. 下列表达方式正确的是_

A. class P{

public:

int x = 15;

void show(){cout<<x;}

};

B. class P{

public:

int x;

void show(){cout<<x;}

}

//错误 结尾缺少分号

C. class P{

```
int f;

};

f=25;//错误  类外部不可初始化
```

D. class P{

```
public:

int a;

void Seta(int x){a=x;}
```

//错误 未闭合

解析：见选项注释

答案为 **A**

8. 以下拷贝构造函数具有的特点中，错误的是_

A. 如果一个类中没有定义拷贝构造函数时,系统将自动生成一个默认的

B. 拷贝构造函数只有一个参数,并且是该类对象的引用

C. 拷贝构造函数是一种成员函数

D. 拷贝构造函数的名字不能用类名

解析:

A. 正确

A. 正确

A. 正确

D. 错误 拷贝构造函数的名字只能是类名

故选 **D** 选项

9. 下列关于友元函数的描述中错误的是_

A. 友元函数不是成员函数

B. 友元函数只可访问类的私有成员

C. 友元函数的调用方法同一般函数

D. 函数可以是另一类中的成员

解析:

A. 正确

B. 错误 都可以

C. 正确

D. 正确，反正不是自己类的成员

故选 **B** 选项

10. 下列关于类型转换函数的描述中,错误的是_

A. 类型转换函数是一种成员函数

B．类型转换函数定义时不指出类型型，也没有参数

C．类型转换函数的功能是将其函数名所指定的类型转换为该类型

D. 类型转换函数在一个类中可定义多个

解析:

A. 错误

B. 正确

C. 正确

D. 正确 多个转换为不同类型

operator 转换类型(){转换代码 return }```

故选 **A** 选项

11. 类体内成员有三种访问权限，他们的关键字分别是_,_,_.

解析：private,public,protected.

12. 如果一个类中没有定义任何构造函数时，系统会__.

解析：调用缺省构造函数

13. 静态成员是属于__的，它除了可以通过对象名来引用外，还可以使用__来引用。

解析： 整个类/所有对象， 类名

14. 友元函数是被说明在_内的__函数。友元函数可以访问该类的成员。

>>解析：当前类， 非成员

15.

```
#include<iostream.h>

 class A{

  int x,y;

  public:

    A(int a,int b){

      x=a;

      y=b;

      cout<<"ABC"<<'\t';

    }

    A(){
```

```
        x=3;
        y=4;
        cout<<"CBA"<<'\n';
    }
    void Show(){
        cout<<"x="<<x<<'\t'<<y<<'\t';
    }
    ~A(){
        cout<<"XYZ"<<'\n';
    }
};

void main(){
  A *s1 = new A(1,2),*s2 = new A;
  s2->Show();
  delete s1;
  delete s2;

}
```

问题一：本程序的执行后输出的结果是_

问题二：如果将语句s2->Show()改为s1->Show(),则执行结果是_

解析： 输出内容为

ABC CBA

x=3 y=4 XYZ (若问题2情况下输出是x=1 y=4 XYZ)

XYZ

答案见解析

16.

```
#include<iostream.h>
```

```
 class A{
  int x,y;
  public:
    A(int a){
      x=a;
      cout<<"x="<<x<<'\t'<<"class_A"<<'\n';
    }
    ~A(){
      cout<<"class_~A"<<'\n';
    }
};

class B{
  A y,z;
  int s;
  public:
    B(int a,int b,int c):y(a+b+c),z(3-a){
      s=c-b;
      cout<<"s="<<s<<'\t'<<"class_B"<<'\n';
    }
    ~B(){
      cout<<"class_~B"<<'\n';
    }
};

void main(void){
  B s(1,2,3);
}
```

问题：本程序共输出行，其中第三、四行分别是和

解析：执行顺序y(a+b+c) --> z(3-a) --> B() --> ~B() --> ~A() --> ~A()

x=6 class_A

x=2 class_A

s=1 class_B

class_~B

class_~A

class_~A

答案为：6，s=1 class_B，class_~B

17.

```
#include<iostream.h>

class node{

  int x,y;

  public:

    node(int a,int b){

      x=a;

      y=b;

      cout<<"node_1"<<'\n';

    }

    node(){

      x=a.x;

      y=a.y;

     cout<<"node_2"<<'\n';

    }

    void Show(){

      cout<<"x="<<'\t'<<"y="<<y<<'\n';

    }
```

```
};


void main(){
  node f1(5,6);
  node f2(f1);
  f2.Show();
}
```

问题一：具有拷贝功能的构造函数node()的参数表中缺少一个形参，这个形参的正确定义是。

问题二：node()中的形参被正确定义后，执行结果依次是__。

解析：node(){}里的a.x 显然找不到a，所以node(){}参数里应有a，根据x=a.x和node f2(f1);可知a应是node类型,该函数是类的拷贝函数问题一的答案应该是node& a

执行结果为：

node_1

x=5 y=6

故答案为：node& a，执行结果(见上)

18. 一下student类用于查找考试成绩在60分以下的学生及其学号，并统计这些学生的总人数请填空。

```
#include<iostream.h>
class student{
  int i,countl;
  float *stu;
  int *num;
  int n,m;
  public:
    student(int k){
      m=0;
      n=k;
```

```
      stu=new float[n];
      _________
      for(i=1;______;i++)
        cin>>num[i]>>stu[i];
    }
    void stat(){
      for(i=1;i<n;i++)
        if(stu[i]<60){
          ________;
          Show();
        }
    }
    void Show(){
      cout<<"number:"<<num[i]<<'\t'<<"grade:"<<stu[i]<<'\n';
    }
    void print(){
      cout<<"fluked total:"<<m<<endl;
    }
};
void main(){
  cout<<"请输入总人数：\n";
  int n;
  cin>>n;
  student a(n);
  a.stat();
  a.print();
}
```

解析

阅读程序可以发现问题：num未初始化，根据num[i]可知num和stu定义方式一样是数组，因此第一处填空应写

num=new int[n];```

依据两个for循环体内代码类似，可推测出第二初填空应写`i < n`

根据题意和Show()，显然变量m是总人数,而m除了初始化为0为做其他更改，结合题意可知 第二初填空应写`m++`

故答案为

num=new int[n];```

i < n```

m++```

19. 按照下列要求编程：

(1) 定义一个描述矩形的类Rectangle，包括的数据成员有宽（width）和长（length）；

(2) 计算矩形的周长；

(3) 计算矩形的面积；

(4) 改矩形的大小。

通过实例验证其正确性。

解析：见代码。

答案：

```
#include<iostream>

 class Rectangle {

    private:
       int width;
       int length;
    public:

       int getGirth() {

           return 2 * (width + length);

       }

       int getArea() {

           return width * length;

       }

       void set(int width, int length) {
```

```
            this->width = width;

            this->length = length;

        }

        Rectangle(int width, int length) {

            this->width = width;

            this->length = length;

        }

    };

int main(){

    Rectangle rect(10,20);

    std::cout << rect.getArea() << std::endl;

    std::cout << rect.getGirth() << std::endl;

    rect.set(30, 40);

    std::cout << rect.getArea() << std::endl;

    std::cout << rect.getGirth() << std::endl;

}
```

20. 编程实现一个简单的计算器。要求从键盘上输入两个浮点数，计算他们加减乘除的运算结果。

答案:

```
#include<iostream>

using namespace std;

class Counter{

    double a, b;

    public:
       double add()

       {

            return(a + b);

       }
```

```
        double sub()
        {

            return(a + b);

        }

        double mult()
        {

            return(a * b);

        }

        double divi() {

            return(a / b);

        }

        Counter() {

            cin >> a >> b;

        }

};



int main() {

    Counter c;

    cout << c.add()<<endl;

    cout << c.sub() << endl;

    cout << c.mult() << endl;

    cout << c.divi() << endl;

}
```

21. 编写一个关于求多个某门功课总分和平均分的程序，实现一个有关学生成绩的操作，该类名为Student。具体要求如下：

1. 每个学生信息包括姓名和某门功课成绩
2. 假设五个学生。

3. 使用静态成员计算5个学生的总成绩和平均分。

答案：

```
#include<iostream>
#include<string>
using namespace std;

class Student{

    private:

       int n;

       string *name;

       int *grade;

       static int sum;

    public:

       void push(string name,int grade) {

           this->name[n] = name;

           this->grade[n] = grade;

           sum += grade;

           n++;

       }

       Student(int num) {

           name = new string[num];

           grade = new int[num];

           n = 0;

       }

       void work()

       {

           cout << "求和：" << sum << "\n平均分：" << 1.0 * sum / n;

       }

};

    int Student::sum = 0;
```

```
int main() {

string name;

int grade;

Student stu(5);

for (int i = 0; i < 5; i++)

{

   cin >> name >> grade;

   stu.push(name, grade);

}

stu.work();

}
```

第六章

1.B

C++语言不支持双重继承.

2.C

请参考书P210 表6-1

3.A

略

4.A

基类的protected成员只能在基类中访问 派生类不能访问.

5.B

略

6.B,C

(朱林说的)考试中遇到这样的题目首选B,因为B中子类型是有明确的父子关系的

C中"就是"多少有点错 但是有B放在这里就只好选B了('-')

7.D

除特殊说明外,在派生类对象中处理的该成员默认为派生类中的成员

```
#include <iostream>

using namespace std;



class A {

  public:

    int a = 0;

};



class B: public A {

  public:

    int a = 0;

};



int main() {

  B b;

  b.a = 5;

  cout << b.A::a << endl;
```

```
    cout << b.B::a << endl;



    return 0;
}

//( '-')

//不信的自己去跑一下(用菜鸟教程的C++在线工具)推荐使用DEV-C++5.15里带的TDM-GCC 9.2.0 64位调试
```

8.D

纯概念,可以看一下书P213辅助李姐一下

9.C

略

10.C

略

11.

A

12.

A

13.

C

14.

B

15.

D

16.

D

17.

C

18.

单继承
多继承

19.

父类

20

属性与形参（或参数值）

21.

编译

运行

22.

virtual

23.

纯虚函数

对象指针

对象引用

24.

```
#include<iostream.h>

class Point

{  int x,y;

 public:

  Point(int a=0,int b=0) {x=a; y=b;}

  void move(int xoffset,int yoffset) {x+=xoffset; y+=yoffset;}

  int getx() {return x;}

  int gety() {return y;}
```

```
};

class Rectangle:protected Point

{  int length,width;

 public:

   Rectangle(int x,int y,int l,int w):Point(x,y)

   { length=l;width=w;}

   int getlength(){return length;}

   int getwidth(){return width;}

};

void main()

{ Rectangle r(0,0,8,4);

 r.move(23,56);

cout<<r.getx()<<","<<r.gety()<<","

<<r.getlength()<<","<<r.getwidth()<<endl;

}
```

分析：保护继承方式使基类的public成员在派生类中的访问属性变为protected，所以派生类Rectangle的对象r不能直接访问基类的成员函数move()、getx()和gety()。其改正方法有两种：1）将Rectangle的继承方式改为公有继承public；2）在Rectangle类中重定义move()，getx()和gety()函数，覆盖基类的同名函数。

void Rectangle::move(int xoffset,int yoffset){Point::move(xoffset,yoffset);}

void Rectangle::getx(){return Point::getx();}

void Rectangle::gety(){return Point::gety();}

25.

将类A与类B中的：

int x 改成：virtual int x

void display() 改成：virtual void display()

26.

```
#include <iostream.h>

class Base

{   int i;

  public:
```

```
    Base(int n){cout <<"Constucting base class" << endl;i=n;}
    ~Base(){cout <<"Destructing base class" << endl;}
    void showi(){cout << i<< ",";}
    int Geti(){return i;}
   };
class Derived:public Base
{   int j;
    Base aa;
   public:
    Derived(int n,int m,int p):Base(m),aa(p){
    cout << "Constructing derived class" <<endl;
    j=n;
    }
    ~Derived(){cout <<"Destructing derived class"<<endl;}
    void show(){Base::showi();
        cout << j<<"," << aa.Geti() << endl;}
   };
void main()
{ Derived obj(8,13,24);
 obj.show();
 }
```

说明：派生类的构造函数的执行次序，先调用基类的构造函数，再调用派生类中子对象类的构造函数，最后调用派生类的构造函数。析构函数的执行次序与构造函数正好相反，先调用派生类的析构函数，再调用派生类中子对象类的析构函数，最后调用基类的析构函数。

运行结果：

Constucting base class

Constucting base class

Constructing derived class

13,8,24

Destructing derived class

Destructing base class

Destructing base class

27.

class B

class C

class A

class D

28.

```
#include <iostream.h>

class A{

public:

    A() { cout<<"A's cons."<<endl; }

virtual ~A() { cout<<"A's des."<<endl; }

    virtual void f() { cout<<"A's f()."<<endl; }

void g() { f(); }

};

class B : public A{

public:

    B() { f(); cout<<"B's cons."<<endl; }

~B() { cout<<"B's des."<<endl; }

};

class C : public B{

public:

    C() { cout<<"C's cons."<<endl; }

~C() { cout<<"C's des."<<endl; }

void f() { cout<<"C's f()."<<endl; }

};

void main()
```

```
{   A *a=new C;

    a->g();

    delete a;

}
```

运行结果：

A's cons.

A's f().

B's cons.

C's cons.

C's f().

C's des.

B's des.

A's des.

分析：(1) 类B从类A公有继承，类C从类B公有继承，B是A的子类型，C是B的子类型，也是A的子类型。因此可以使用类C的对象去初始化基类A的指针；(2) 类A中定义了两个虚函数：虚成员函数f和虚析构函数。由于类C中的f函数与基类A中的f函数的参数和返回类型相同，因此类C中的成员函数f也是虚函数；由于基类A中定义了虚析构函数，因此派生类B和C的析构函数也都是虚析构函数；(3) 主程序中A *a=new C;隐含了两步操作，即：首先建立一个派生类C的对象，然后使用该派生类对象去初始化基类A的指针a；(4) 用new运算符创建C对象时，要调用C的构造函数。C是一个派生类，它必须负责对其直接基类的构造函数的调用，因此执行顺序是：先执行直接基类B的构造函数，再执行类C的构造函数体；(5) 类B也是一个派生类，它也必须负责调用它的直接基类A的构造函数，其执行顺序是：先执行直接基类A的构造函数，再执行类B的构造函数体；(6)执行类A的构造函数，输出：A's cons.；(7)类B的构造函数体中，首先调用了虚函数f，但由于是在构造函数中调用虚函数，所以对它的调用采用静态联编。因为类B中没有对f函数进行定义，因此调用基类A中的f函数，首先输出A's f().；继续执行类B的构造函数体，输出B's cons.；(8) 执行类C的构造函数体，输出：C's cons.；(9) 成员函数g中调用了虚函数f，此时满足动态联编的条件：C是A的子类型；存在虚函数f；在成员函数中调用了虚函数，因此，通过指针a访问g函数时，采用动态联编，即a->g()调用的是派生类C中的成员函数f，输出：C's f().；(10)delete a将调用析构函数。由于整个类族中都定义了虚析构函数，因此此处将进行动态联编，即调用的是基类指针a当前正在指向的派生类C的析构函数。由于C是一个派生类，所以它必须负责调用它的基类的析构函数，其执行顺序是：先调用派生类C的析构函数，再调用直接基类B的析构函数。而直接基类B也是一个派生类，所以它也必须负责调用它的直接基类A的析构函数。因此，析构的顺序是：C、B、A，输出：C's des.；B's des.；A's des.。析构的顺序与构造的顺序完全相反。

容易发生误解是，不清楚用new运算符建立对象时将自动调用构造函数； 不清楚派生类构造函数的执行顺序，误认为先调用派生类构造函数，再调用基类构造函数，导致先输出：C's cons.；不清楚派生类只负责对其直接基类构造函数的调用，在建立C类的对象时，错误地既调用了直接基类B的构造函数，又调用了间接基类A的构造函数，导致错误输出：B's cons.；A's cons.；不清楚在构造函数和析构函数中调用虚函数时只能进行静态联编，误认为此处也是动态联编，导致错误输出：C's f().；不清楚动态联编的实现条件，不知道在成员函数中调用虚函数是进行动态联编，导致对g函数的调用产生错误输出：A's f().；不清楚用delete运算符释放对象时将自动调用析构函数；不清楚对虚析构函数的调用采用的是动态联编，误认为释放的是基类A的对象，导致错误输出：A's des.；不清楚派生类的析构函数的执行顺序，误认为与构造时的顺序相同，导致错误输出：A's des.；B's des.；C's des.。

29.

答：继承方式决定了基类中的成员在派生类中的属性。三种继承方式的共同点：基类的private成员在派生类中不可见。区别：对于私有继承，基类的public，protected成员在派生类中作为private成员；对于公有继承，基类的public，protected成员在派生类中访问属性不变；对于保护继承，基类的public，protected成员在派生类中作为protected成员。

30.

答：派生类构造函数的执行顺序是先执行所有基类的构造函数（顺序按照定义派生类是指定的各基类顺序），再执行对象成员所在类的构造函数（顺序按照他们在类中的声明顺序），最后执行派生类构造函数体中的内容

31.

答：因为在派生类B已经重载了基类A的一个成员函数fn1()，所以要用作用域运算符对fn1()函数加以限定，调用基类的成员函数fn1()是A::fn1()；因为在派生类B没有重载成员函数fn2()，所以直接可调用fn2()。

32.

答：在多重继承中，如果多条继承路径上有一个公共的基类，则在这些路径的汇合点上的派生类会产生来自不同路径的公共基类的多个拷贝，如果用virtual把公共基类定义成虚基类，则只会保留公共基类的一个拷贝。引进虚基类的目的是为了解决二义性问题，使得公共基类在它的派生类对象中只产生一个基类子对象。

33.

答：多态是指同样的消息被不同类型的对象接收时导致完全不同的行为，是对类的特定成员函数的再抽象．c十+支持的多态有多种类型．重载(包括函数重载和运算符重载)和虚函数是其中主要的方式。

34.

答：带有纯虚函数的类是抽象类．抽象类的主要作用是通过它为一个类族建立一个公共的接口．使它们能够更有效地发挥多态特性。抽象类声明了一组派生类共同操作接口的通用语义．面接口的完整实现，即纯虚函数的函数体，要由派生类自己给出．但抽象类的派生类并非一定要给出纯虚函数的实现．如果派生类没有给出纯虚函数的实现，这个派生类仍然是一个抽象类。

35.

答：在C++-中不能声明虚构造函数．多态是不同的对象对同一消息有不同的行为特性．虚函数作为运行过程中多态的基础，主要是针对对象的，面构造函数是在对象产生之前运行的，因此虚构造函数是没有童义的 在C++中可以声明虚析构函数．析构函数的功能是在该类对象消亡之前进行一些必要的清理工作，如果一个类的析构函数是虚函数，那么，由它派生而来的所有子类的析构函数也是虚函数．析构函数设置为虚函数之后，在使用指针引用时可以动态联编，实理运行时的多态，保证使用基类的指针就能够谓用适当的析构函数指针对不同的对象进行清理工作

36.

```
#include <iostream>

using namespace std;
```

```
class rectangle

{

 public:

rectangle( double l,double w )

 { length = l;width = w; }

double area()

 { return( length*width ); }

double getlength()

{ return length; }

double getwidth()

{ return width; }

private:

  double length;

   double width;

};

class rectangular:public rectangle

{

 public:

  rectangular( double l,double w,double h ) : rectangle( l,w )

   { height = h; }

  double getheight()

{ return height; }

   double volume()

 { return area() *height; }

 private:

   double height;

};

int main()

{
```

```
rectangle obj1( 2,8 );

rectangular obj2( 3,4,5 );

cout<<"length="<<obj1.getlength()<<'\t'<<"width="<<obj1.getwidth()<<endl;

cout<<"rectanglearea="<<obj1.area()<<endl;

cout<<"length="<<obj2.getlength()<<'\t'<<"width="<<obj2.getwidth();

cout<<'\t'<<"height="<<obj2.getheight()<<endl;

cout<<"rectangularvolume="<<obj2.volume()<<endl;

}
```

37.

```
#include <iostream.h>

class Building

{public:

 Building(int f,int r,double ft)

 {floors=f;

  rooms=r;

  footage=ft;

 }

 void show()

 {   cout<<" floors: "<<floors<<endl;

    cout<<" rooms: "<<rooms<<endl;

    cout<<" total area: "<<footage<<endl;

 }

protected:

 int floors;

 int rooms;

 double footage;

};
```

```
class Housing:public Building
{public:
 Housing(int f,int r,double ft,int bd,int bth):Building(f,r,ft)
 { bedrooms=bd;
bathrooms=bth;
 }
 void show()
 {cout<<"\n HOUSING:\n";
  Building::show();
  cout<<" bedrooms: "<<bedrooms<<endl;
  cout<<" bathrooms: "<<bathrooms<<endl;
 }
private:
 int bedrooms;
 int bathrooms;
};
class Office:public Building
{
public:
 Office(int f,int r,double ft,int ph,int ex):Building(f,r,ft)
 { phones=ph;
  extinguishers=ex;
 }
 void show()
 {cout<<"\n HOUSING:\n";
  Building::show();
  cout<<" phones: "<<phones<<endl;
  cout<<" extinguishers: "<<extinguishers<<endl;
 }
```

```
private:
 int phones;
 int extinguishers;
};
void main()
{ Housing hob(5,7,140,2,2);
 Office oob(8,12,500,12,2);
 hob.show();
 oob.show();
}
```

38.

```
#include <iostream>
using namespace std;
class Employee
{
  public:
   Employee( char Snumber[]="\0", char Sname[]="\0", double bSalary=2000 )
   {
        strcpy(number,Snumber);
        strcpy(name,Sname);
        basicSalary=bSalary;
     }
     void input()
     {
        cout << "编号: " ; cin >> number;
        cout <<"姓名: " ; cin >> name;
     }
     void print()
```

```
    {
     cout<<"员工 ： "<<name<<"\t\t编号：  "<<number<<"\t\t本月工资：  "
<<basicSalary<<endl;
    }
  protected:
   char number[5];
   char name[10];
   double basicSalary;
};

class Salesman: public Employee
{
 public:
    Salesman(int sal=0)
    { sales=sal; }
    void input()
    {
       Employee::input();
       cout<<"本月个人销售额：  ";
       cin>>sales;
    }
    void pay()
    {
       salary = basicSalary+sales*commrate;
    }
    void print()
    {
       pay();
       cout<<"销售员 ： "<<name<<"\t\t编号：  "<<number<<"\t\t本月工资：  "
<<salary<<endl;          }
```

```
 protected:
     static double commrate;
     int sales;
     double salary;
};
double Salesman :: commrate=0.005;


class Salesmanager : public Salesman
{
  public:
     Salesmanager(double jSalary=3000)
     {
  jobSalary = jSalary;
     }
     void input()
     {
        Employee::input();
        cout<<"本月部门销售额：";
        cin>>sales;
     }
   void pay()
     {
        salary = jobSalary + sales*commrate;
     }
     void print()
   {
        pay();
        cout<<"销售经理 ： "<<name<<"\t\t编号："<<number<<"\t\t本月工资："
<<salary<<endl;         }
```

```
private:
   double jobSalary;
};
int main()
{
 cout<<"基本员工\n";
 Employee emp1;
 emp1.input();
 emp1.print();
 cout<<"销售员\n";
 Salesman emp2;
 emp2.input();
 emp2.print();
 cout<<"销售经理\n";
 Salesmanager emp3;
 emp3.input();
 emp3.print();
}
```

39.

```
#include<iostream.h>
class Mammal
{ public:
  Mammal()  {cout<<"call Mammal"<<endl; }
  virtual void speak()  {cout<<" call base class"<<endl;  }
};
class Dog :public Mammal
{ public:
   Dog()  {cout<<"call Dog\n"; }
```

```
  void speak() {cout<<"call Dog class\n";  }
};
void main()
{ Mammal a;
 a.speak();
 Dog b;
 b.speak();
}
```

第七章 运算符重载

1

A

2

A

3

C

4

D

5

D

6

函数

运算符

7

成员

友元

8

采用成员函数实现”+”运算符重载时，对象c1+c2，编译器将解释为：_c1.operator+(c2)__，

而采用友元函数实现”+”运算符重载时，对象c1+c2，编译器将解释为：*operator+(c1, c2)*。

9

在类名为classname中用友元函数声明 “>>”重载函数的格式为：

friend istream & operator>> (istream &, classname &)；。

10

this指针

成员函数参数

分析：将双目运算符重载为类的成员函数时，由于this指针在每次非静态成员函数操作对象时都作为第一个隐式参数传递给对象，因此它充当了二元运算符的左操作数，而该成员函数的形参则表示二元运算符的右操作数。

容易错误分析：(1)不了解this指针的作用，所以不知道二元运算符的左操作数如何表示；(2) 误将成员函数的参数作为二元运算符的左操作数，而不知道右操作数如何表示；(3) 混淆了成员函数和友元函数两种方式，误认为二元运算符的所有的操作数都必须通过函数的形参进行传递，函数的参数与操作数自左至右一一对应；

11

```
#include<iostream.h>

class Arr

{

    int x[20];

public:

    Arr(){for(int i=0;i<20;i++) x[i]=0;}

    Arr(int *p)

    {for(int i=0;i<20;i++)x[i]=*p++;}

    Arr operator+(Arr a)

    {

       Arr t;

       for(int i=0;i<20;i++)

       t.x[i]=___;

       return ____;

    }

    Arr operator+=(Arr a)
```

```
    {
       for(int i=0;i<20;i++)x[i]=____;
       return  ____;
    }
    void show()
    {
       for(int i=0;i<20;i++)cout<<x[i]<<'\t';
       cout<<endl;
    }

};
void main()
{
  int array[20];
    for(int i=0;i<20;i++) array[i]=i;
    Arr a1(array),a2(array),a3;
    a3=a1+a2;a3.show ();
    a1+=a3;a1.show();
}
```

①x[i]+a.x[i]

②t

③x[i]+a.x[i]

④*this

12

```
#include <iostream.h>
#include <iomanip.h>
#include <string.h>
#include <stdlib.h>
class Sales
{
```

```
public:
void Init(char n[]) { strcpy(name,n);

}
int& operator[](int sub);
char* GetName() { return name; }
private:
char name[25];
int divisionTotals[5];
};
int& Sales::operator [](int sub)
{   if(sub<0||sub>4)
    {   cerr<<"Bad subscript! "<<sub<<" is not allowed."<<endl;
        abort();
    }
    return divisionTotals[sub];
}
void main()
{   int totalSales=0,avgSales;
    Sales company;
    company.Init("Swiss Cheese");
    company[0]=123;
    company[1]=456;
    company[2]=789;
    company[3]=234;
    company[4]=567;
    cout<<"Here are the sales for "<<company.GetName()<<"'s divisions:"<<endl;
    for(int i=0;i<5;i++)
        cout<<company[i]<<"\t";
    for(i=0;i<5;i++)
        totalSales+=company[i];
    cout<<endl<<"The total sales are "<<totalSales<<endl;
    avgSales=totalSales/5;
    cout<<"The average sales are "<<avgSales<<endl;
}
```

运行结果：
Here are the sales for Swiss Cheese's divisions:
123 456 789 234 567
The total sales are 2169
The average sales are 433
分析：(1) 在上述程序中，并没有创建company的对象数组。程序中的下标被重载，以便在使用company时用一种特殊的方式工作。下标总是返回和下标对应的那个部门的销售额divisionTotals[]；(2) 重载下标运算符"[]"时，返回一个int型引用，可使重载的"[]"用在赋值语句的左边，因而在main()中，可对每个部门的销售额divisionTotals[]赋值。这样，虽然divisionTotals[]是私有的，main()还是能够直接对其赋值，而不需要使用函数Init()；(3) 在上述程序中，设有对下标的检验，以确保被赋值的数组元素存在。当程序中一旦向超出所定义的数组下标范围的数组元素进行赋值时，便会自动终止程序，以免造成不必要的破坏；(4)与函数调用运算符"()"一样，下标运算符"[]"不能用友元函数重载，只能采用成员函数重载。

13

```
#include <iostream.h>
```

```
#include <string.h>
class Employee
{
 public:
   Employee(void) {};
   Employee(char *name, char sex, int age, char *phone)
   {
     strcpy(Employee::name, name);
     Employee::sex = sex;
     Employee::age = age;
     strcpy(Employee::phone, phone);
   };
     friend ostream &operator<<(ostream &cout, Employee emp);
     friend istream &operator>>(istream &stream, Employee &emp);
 private:
   char name[256];
   char phone[64];
   int age;
   char sex;
};
ostream &operator<<(ostream &cout, Employee emp)
 {
   cout << "Name: " << emp.name << "; Sex: " << emp.sex;
   cout << "; Age: " << emp.age << "; Phone: " << emp.phone << endl;
   return cout;
 }
istream &operator>>(istream &stream, Employee &emp)
 {
   cout << "Enter Name: ";
   stream >> emp.name;
   cout << "Enter Sex: ";
   stream >> emp.sex;
   cout << "Enter Age: ";
   stream >> emp.age;
   cout << "Enter Phone: ";
   stream >> emp.phone;
   return stream;
 }
void main(void)
 {
   Employee worker;
   cin >> worker;
   cout << worker ;
 }
```

6. Name: tom; Sex: m; Age: 23; Phone: 321456

14

10

10

15

```
#include <iostream.h>
class Point
{public:
Point() { X=Y=0; }
int GetX() { return X; }
int GetY() { return Y; }
Point& operator ++();
Point operator ++(int);
Point& operator --();
Point operator --(int);
void Print() { cout<<"The point is ("<<X<<","<<Y<<")"<<endl; }
    private:
int X,Y;
};
Point& Point::operator ++()
{   X++;
    Y++;
    return *this;
}
Point Point::operator ++(int)
{   Point temp=*this;
    ++*this;
    return temp;
}
Point& Point::operator --()
{   X--;
    Y--;
    return *this;
}
Point Point::operator --(int)
{   Point temp=*this;
    --*this;
    return temp;
}
void main()
{   Point obj;
    obj.Print();
    obj++;
    obj.Print();
    ++obj;
    obj.Print();
    obj--;
    obj.Print();
    --obj;
    obj.Print();
}
```

分析：注意重载前缀单目运算符和后缀单目运算符的区别。前缀单目运算符重载为类的成员函数时，不需要显式说明参数，即函数没有形参；而后缀单目运算符重载为类的成员函数时，函数要带有一个整型形参。

16

```
#include<iostream.h>

class Counter
```

```
{public:
   Counter()
   Counter(int initialValue);
   ~Counter(){}
    int GetItsVal()const {return itsVal;}
    void SetItsVal(int x) { itsVal=x;}
    Counter operator+(const Counter  &);
private:
     int itsVal;
};
Counter::Counter(int initialValue):itsVal(initialValue){ }
Counter::Counter():itsVal(0) {  }
Counter Counter::operator+(const Counter &rhs)
{ return Counter(itsVal+rhs.GetItsVal());}
void main()
{  Counter varOne(2),varTwo(4),varThree;
  varThree=varOne+varTwo;
  cout<<"varOne:"<<varOne.GetItsVal()<<endl;
  cout<<"varTwo:"<<varTwo.GetItsVal()<<endl;
 cout<<"varThree:"<<varThree.GetItsVal()<<endl;
 }
```

第八章

1

A

2

C

3

C

4

D

5

C

6

D

7

D

8

cin

cout

cerr

clog

9

cur

beg

end

10

while循环中需要加循环终止条件，在①处应该判断文件是否结束，应填入“!me.eof()”；根据getline（）函数的原型可知，此处应填入一个字符串变量，故填入“buf”。即答案为：

!me.eof()

buf

11

向文件写内容前首先需要打开该文件并指明需要的输入输出操作；在使用完毕后应当关闭该文件。在读到文件结尾时应当结束读取操作。故本题答案为：

f.open(“c:\new”,ios::in|ios::out)

!f.eof()

f.close()

12

abcdef

123456

ijklmn

13

abcdefg1234567

14

Yan 5002011 278

15

```
#include<fstream.h>
#include<stdlib.h>
void main()
{
    fstream file1,file2;
    char buffer[100];
    int i=0;
```

```
    file1.open ("old.txt",ios::in);
    if(!file1)
    {cout<<"can not open this file!"<<endl;exit(2);}
    file2.open("new.txt",ios::out);
    while(!file1.eof())
    {
        file1.getline(buffer,sizeof(buffer));
        file2<<buffer<<"."<<endl;
    }
    cout<<"OK!"<<endl;
}
```

16

```
#include <iostream.h>
#include <fstream.h>
#include <stdlib.h>
void main()
{
    fstream f;
    f.open("abc.txt",ios::in);
    if(!f)
    {
        cout<<"abc.txt can't open.\n";
        abort();
    }
    char ch;
    int n=0;
    while(!f.eof())
    {
        f.get(ch);
        n++;
    }
    cout<<"该文件字符数为 "<<n<<endl;
    f.close();
}
```

17

```
#include <iostream.h>
#include <fstream.h>
#include <stdlib.h>
void main()
{
    fstream inf,outf;
    inf.open("del.cpp",ios::in);
    if(!inf)
    {
        cout<<"Can't open.\n";
        abort();
```

```
    }
    outf.open("del22.cpp",ios::out);
    if(!outf)
    {
        cout<<"Can't open.\n";
        abort();
    }
    char s[80];
    int n=1;
    while(!inf.eof())
    {
        inf.getline(s,sizeof(s));
        outf<<n++<<": "<<s<<endl;
    }
    inf.close();
    outf.close();
}
```

18

```
#include <fstream.h>
class Dog
{public:
     Dog(int w,int a):weight(w),age(a) {}
     ~Dog() {}
     int GetWeight() { return weight; }
     int GetAge() { return age; }
     void SetWeight(int w) { weight=w; }
     void SetAge(int a) { age=a; }
private:
      int age,weight;
};
int main()
{   char fileName[80];
    cout<<"Please input the file name: ";
    cin>>fileName;
    ofstream fout(fileName);
    if(!fout)
    {   cout<<"Can't open the file "<<fileName<<" for writing."<<endl;
        return(1);
    }
    Dog dog1(5,10);
    fout.write((char*)&dog1,sizeof(dog1));
    fout.close();
    ifstream fin(fileName);
    if(!fin)
    {   cout<<"Can't open the file "<<fileName<<" for reading."<<endl;
        return(1);
    }
    Dog dog2(0,0);
    fin.read((char*)&dog2,sizeof(dog2));
    fin.close();
    cout<<"Dog2's weight: "<<dog2.GetWeight()<<endl;
    cout<<"Dog2's age:    "<<dog2.GetAge()<<endl;
```

```
    return 0;
}
```

第九章 模板与异常处理

1

> 正确选项:A

函数模板由编译器根据程序员的调用类型实例化为可执行的函数

2

> 正确选项:C

模板声明以关键字template开始类型参数表必须用<>括起来，类型参数表中列举一个或多个类型参数项（用逗号分隔参数项)时，每一项由typename或class后跟一个标识符组成

3

> 正确选项:B

在调用函数时，编译器按最先遇到的实参的类型隐含生成一个模板函数，并用它对所有的参数进行类型一致性检查，类型不同将出现类型不一致的错误

4

> 正确选项:A

普通基类可以派生类模板

5

> 正确选项:C

建立类模板的对象时，首先需要将类模板实例化，即想模板传递参数完成类模板的实例化，然后在定义该类的对象。模板函数只有在使用时候才会进行实例化

6

> 正确选项:D

容器是随着面向对象语言的诞生而提出的，在C++ 中，就是标准模板库（STL ）

7

> 正确答案:
>
> i=20　j=10
>
> a=2.2　b=1.1

8

> 正确答案:
>
> 8.8

> 8

9

> 正确答案:
>
> 数组下标越界!
> s=55

10

> 正确答案:
>
> main function
> constructor
> exception1
> exception2
> main function

11

> 正确答案:
>
> 抽象类无法实例化，也就是无法创建对象。其派生类必须实现纯虚函数才能被实例化。
>
> 类模板按所给参数类型生成对应的类。
>
> 抽象类通常是作为基类，让派生类去实现纯虚函数。
>
> 类模板通常用于在需要编写多个形式和功能都相似的类时避免重复劳动

12

> 正确答案:
>
> 一个应用不一定要设计异常处理程序。
>
> 异常处理以结构化思想把异常检测与异常处理分离,增加了程序的可读性,便于大型软件的开发

13

> 正确答案:
>
> 抛出异常是为了在你的程序运行错误的情况下,程序也能继续执行下面的代码,而不会跳出这个的程序运行
>
> catch只是根据异常参数的类型（不管具体数值）处理异常
>
> 测试程序:

```
#include<iostream>
using namespace std;
int main ()
{
double m , n;
cin >> m >> n;
try
{
  cout << "before dividing" << endl;
```

```
  if (n == 0)
      throw -1;//抛出int类型异常
  else
      cout << m / n << endl;
  cout << "after dividing" << endl;
}
catch (double d)
{
  cout << "catch(double)" << d <<  endl;
}
catch (int e)
{
  cout << "catch(int)" << d <<  endl;
}
cout << "finished" << ebdl;
return 0;
}
```

14

```
//正确答案:
#include <iostream>
#include <cmath>
using namespace std;
int main ()
{
double x , y;
cin>> x >> y;
try
{
if (( 2 * x - y )<0)
   throw  2 * x - y ;
else
   cout<<" ln(2*x-y)="<<log( 2 * x - y ) << endl;
}
catch(double a)
{
cout<< "2*x+y的值为负数" << endl;
}
return 0;
}
```

15

```
//正确答案:
#include <iostream>
#include <time.h>
using namespace std;
#define N 10
template < typename T>
T avg ( T a[] )
{
    T s = 0;
    for (int i = 0; i < N; i++ )
```

```
        s += a[i];
    return s/N;
}
int main()
{
    srand ( time(NULL) );
    int a[N]; float b[N];
    cout<< "整型数组： 浮点型数组:" <<endl;
    for ( int i = 0; i < N; i++ )
    {
        a[i] = rand();
        b[i] = rand() / ( RAND_MAX + 0.0 );
        cout<< a[i] <<" "<< b[i] <<endl;
    }
    cout<<"整型数组平均值："<< avg(a) <<'\n'<<"浮点型数组平均值:"<< avg(b)
<<endl;
return 0;
}
```

16

```
#include<iostream>
using namespace std;
template <class T> class node
{
    private:
        node *next = NULL;
        T data;
    public:
        node* visnex();
        void adjnex(node *a);
        void fuzhi(T num);
        T visval();
};
template <class T> class Linklist
{
    private:
        node<T> *head = new node<T>;
    public:
        void insert(T a);
        void traverse();
};
template <class T> node<T>* node<T>::visnex()
{
    return next;
}
template <class T> void node<T>::fuzhi(T num)
{
    data = num;
}
template <class T> T node<T>::visval()
{
    return data;
}
template <class T> void node<T>::adjnex(node *a)
{
```

```
    next = a;
}
template <class T> void Linklist<T>::insert(T a)
{
    node<T> *ne = new node<T>, *nex = head->visnex(), *front = head;
    ne->fuzhi(a);
    while (nex)
    {
        front = nex;
        nex = nex->visnex();
    }
    front->adjnex(ne);
}
template <class T> void Linklist<T>::traverse()
{
node<T> *temp = head->visnex();
cout << "The result is:";
while (temp)
{
  cout << temp->visval() << " ";
  temp = temp->visnex();
}
cout << endl;
}
int main()
{
    Linklist<int> list;
cout << "Please input the number of elements in a initial linklist" << endl;
int n;
cin >> n;
cout << "Please input every element" << endl;
for (int i = 0; i < n; i++)
{
  int t;
  cin >> t;
  list.insert(t);
  list.traverse();
}
    system ("pause");
}
```