# 东 南 大 学 考 试 卷（A 卷）

课程名称 计算机科学基础（下） 考试学期 14-15-3 得分

适用专业 信息工程 考试形式 闭卷 考试时间长度 120 分钟

| 题 目 | 一 | 二 | 三 | 四 | 总 分 |
|---|---|---|---|---|---|
| 得 分 | 20 | 8 | 27 | 16 | 71 |
| 批阅人 | 徐 | 徐 | 印 | 印 | |

## 一、选择题（请将答案填写在下面表格中。每空 2 分，共 20 分）：

| (1) | (2) | (3) | (4) | (5) | (6) | (7) | (8) | (9) | (10) |
|---|---|---|---|---|---|---|---|---|---|
| A | D | D | C | B | A | C | C | B | B |

A 1. 以下关于链表的叙述，错误的是_____

A. 链表可以进行随机读写
B. 链表的节点包含指针域和数据域
C. 链表的节点可以在运行中动态建立
D. 单链表有前向和后向两种生成链表的方法

D 2. 基类的私有成员在其派生类中的访问属性为_____

A. 私有成员 B. 保护成员 C. 共有成员 D. 不可见

D 3. 以下关于构造函数的叙述中错误的是_____

A. 构造函数名必须与所在类的类名相同
B. 当新的对象被建立，该对象所属的类的构造函数自动被调用
C. 构造函数可以重载
D. 构造函数的返回类型为 void

C 4. 以下关于模板的叙述中错误的是_____

A. 引入模板可以增强代码的通用性，使之不受数据类型的限制
B. 模板中可以将数据类型作为参数
C. 可以重载的函数都可以用模板实现
D. 类模板的成员函数可以在类模板外定义

B 5. 以下关于动态内存分配的叙述中正确的是_____

A. 执行 int *p=new int(10);语句后，在堆区创建了长度为 10 的动态数组



B. 若内存不足，无法正常分配空间，则返回空指针 NULL
C. 撤销数组空间时，delete 后面的[]中必须指定数组的长度
D. 用 new 开辟数组空间时可以指定数组元素的初值

A 6. 以下关于赋值兼容的叙述中错误的是_____

A. 基类对象可向派生类对象赋值
B. 派生类对象可替代基类对象向基类对象的引用进行赋值或初始化
C. 若函数的参数是基类对象或基类对象的引用，其实参可以是派生类对象
D. 派生类对象的地址可以赋给指向基类对象的指针变量

C 7. 以下关于虚函数的表述中正确的是_____

A. 包含虚函数的类是虚基类
B. 包含虚函数的类不能定义对象
C. 当某一个类的成员函数被定义为虚函数，则由该类派生的所有派生类中，该函数始终保持虚函数特征
D. 虚函数不能被调用

C 8. 以下关于文件的叙述中错误的是_____

A. 磁盘文件操作是通过文件指针来指明当前应进行读写的位置
B. 二进制文件可以实现随机读写
C. 数值必须存储在二进制文件中
D. 可以用流插入和流提取运算符对文件进行读写

B 9. 以下关于基类和派生类的叙述正确的是_____

A. 基类的构造函数可以被派生类继承
B. 构造函数可以是虚函数
C. 基类的析构函数可以被派生类继承
D. 基类的析构函数可以是虚函数

B 10. 以下关于异常处理的说法错误的是_____

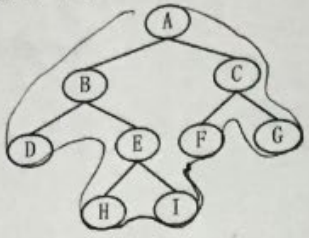
A. 程序中可能包含的两类错误有语法错误和运行错误两大类
B. catch 在捕获异常信息时，只能捕获类型而不能捕获值
C. 处理异常错误的机制包含检查、抛出和捕捉三部分
D. 若在执行 try 块内的语句过程中没有发生异常，则 catch 子句不起作用

## 二、填空题（每空 2 分，共 10 分）：

1. 给定一个数组 int a[8]={5, 2, 8, 1, 3, 7, 4, 6};，利用冒泡排序法对该数组进行升序排序，在第二轮冒泡排序结束后，数组中元素的顺序为 1 2 5 3 8 4 6 7。

2. 队列是一种限定存取位置的数据结构，它的存取特性为 后进先出。

3. 给定二叉树的结构如下，中序遍历的节点顺序是 A B D H E I F C G




4. 线性表是简单常用的一种数据结构，实现线性表的物理结构有两种，分别是 顺序表 和 链表 。

## 三．读程序写结果（每空 2 分，共 40 分）：

1. 阅读以下程序，写出运行结果（6 分）

```
#include<iostream>
using namespace std;
class Sample{
    int i;
    static int k;
public:
    Sample();
    Sample(int a);
    Sample(Sample &a);
    void disp(char *ch);
};
int Sample::k;
Sample::Sample(){
    i=0;
    k++;
}
Sample::Sample(int a){
    i=a;
    k=k+a;
}
Sample::Sample(Sample &a){
    i=a.i;
    k=a.k+++10;
}
```

1. 输出结果：

s1.i=2, k=2
s2.i=2, k=13
s3.i=2, k=13



```
void Sample::disp(char *ch){
    cout<<ch<<"i="<<i<<", k="<<k<<endl;
}
int main(){
    Sample s1(2),s2(s1),s3;
    s3=s2;
    s1.disp("s1.");
    s2.disp("s2.");
    s3.disp("s3.");
    return 0;
}
```

2. 阅读以下程序，写出运行结果（8 分）

```
#include<iostream>
using namespace std;
class A{
    int a1,a2;
public:
    A(int i,int j){
        a1=i;
        a2=j;
    }
    A(A &a){
        a1=a.a1;
        a2=a.a2;
    }
    void Move(int x,int y){
        a1+=x;
        a2+=y;
    }
    void Print(){
        cout<<'('<<a1<<','<<a2<<')'<<endl;
    }
};
class B:private A{
    int b1,b2;
public:
    B(int i,int j,int k,int l):A(i,j){
        b1=k;
        b2=l;
```

2.输出结果：
(11, 12)
(11, 12)
33, 34
(36, 40)

```
    }
    void Print(){
        cout<<b1<<','<<b2<<endl;
    }
    void f(){
        A::Print();
    }
     void fun(){
        Move(5,8);
    }
};
int main(){
    A a(11,12);
    a.Print();
    A aa(a);
     aa.Print();
    B b(31,32,33,34);
    b.fun();
    b.Print();
    b.f();
    return 0;
}
```

(11, 12)

(11, 12)

a=36 a=4?

33, 34

3. 阅读以下程序，写出运行结果（10 分）

```
#include<iostream>
using namespace std;
class one
{
    float x;
public:
    one(float a=0){ x=a; }
    one operator++()
    {
        one t;
        x++;   t=*this;   return t;
    }
    void process(){ Show(); }
    virtual void Show()
    {
        cout<<"x="<<x<<'\t';
    }
};
class two:public one
{
    float y;
public:
    two(float a=0){ y=a; }
    two operator++()
    {
        two t;
        ++y;   *this=t;   return t;
    }
    void Show()
    {
        cout<<"y="<<y<<'\t';
    }
};
class Three:public two
{
    float z;
public:
    ~Three(){cout<<"调用析构函数"<<'\n';}
    Three(float a=0){ z=a; }
    Three operator++()
    {
        Three t=*this;
        ++z;   *this=t;   return t;
    }
    void Show()
    {
        cout<<"z="<<z<<'\n';
    }
};
int main()
{
    one *p, a(2);
    two b(3);
    Three c(4);
    ++a;    p=&a;   p->Show();
    ++b;    p->Show();
```

3.输出结果：

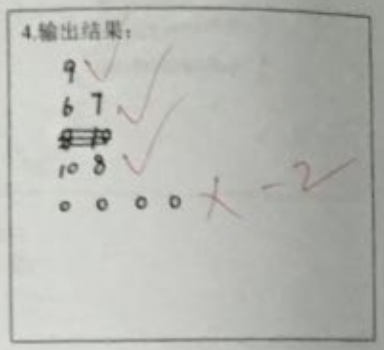
调用析构函数
x=3
y=4
z=5
x=3

```
    ++c;    p=&c;   p->Show();
    c.process();
    return 0;
}
```

4. 阅读以下程序，写出运行结果（8 分）

```
#include<iostream>
using namespace std;
int n1=0,n2=0,n3=0,n4=0;
class A{
    int x;
public:
    A(int a){
        x=a; n1++;
    }
    void print(){
        cout<<x<<endl;
    }
    int Getx(){   return x;   }
};
class A1:public A{
    int a;
public:
    A1(int x,int y):A(x){
        a=y; n2++;
    }
    void printA1(){
        cout<<Getx()<<"\t"<<a<<endl;
    }
};
class A2:public A{
    int b;
public:
    A2(int x):A(10){
        b=x; n3++;
    }
    void printA2(){
        cout<<Getx()<<"\t"<<b<<endl;
    }
```

4.输出结果：





```
};
class A3:public A1,public A2{
    int xx;
public:
    A3(int a,float b,int c,int d):A1(a,b),A2(c){
        xx=d; n4++;
    }
    void printA3(){
        cout<<xx<<endl;
    }
};
int main(){
    A a1(2);
    A1 a2(3,4);
    A2 a3(5);
    A3 a4(6,7,8,9),*p;
    p=&a4;
    p->printA3();
    p->printA1();
    p->printA2();
    cout<<n1<<"\t"<<n2<<"\t"<<n3<<"\t"<<n4<<endl;
    return 0;
}
```

5. 阅读以下程序，写出运行结果（8 分）

```
#include <iostream>
using namespace std;
class cs{
public:
    cs(int i):x(i){ cout << "cs constructor:" << i << endl; }
    ~cs(){ cout << "cs destructor:" << x << endl; }
private:
        int x;
};

void test_func2(){
 cout << "test func2" << endl;
 cs c(3);
}
```

```
void test_func1(){
    cout << "test func1" << endl;
    test_func2();
}

int main(){
    test_func1();
    return 0;
}
```

5.输出结果：

~~test1~~
test func1
test fun c2
cs constructor: 3
cs destructor: 3

+10

四．完善程序题 （每空 2 分，共 30 分）：

1. 完善日期类 Date，使得程序运行结果为：

2000 年 3 月 1 日
2000 年 5 月 1 日
2000 年 11 月 12 日
2001 年 1 月 1 日

```
#include <cmath>
using namespace std;
class  Date
{   int  year , month , day ;
    ____static_____int  md [ 12 ] ;
  public:
    Date (int y, int m , int d ) : year ( y ) , month ( m ) , day ( d )
    {
        if ((year%4==0 && year%100!=0) || (year%400==0))
            ~~day=d+1~~ day=d-1
    }
    void print ( ) {  cout << year << "年" << month << "月" << day << "日\n" ;  }
    Date &  operator ++ ( ) ;
};
int  Date:: md [ 12 ] = { 31 , 28 , 31 , 30 , 31 , 30 , 31 , 31 , 30 , 31 , 30 , 31 } ;
Date &  Date:: operator ++ ( )
{  day++ ;
   if (  month ==12 && day > 31  ) {  ++year ;  month=1 ;  day=1 ;  }
   if (  month !=12 && day > __30 || month==2 && day>28__ ) {
     ++month ;  day=1 ;
  }
```



```
   return __*this__ ;
}
void  main ( )
{  Date  d1(2000,2,29) , d2(2000,4,30) , d3(2000,11,11) , d4(2000,12,31) ;
   (++d1) . print( ) ;  (++d2) . print( ) ;  (++d3) . print( ) ;  (++d4) . print( ) ;
}
```

2. 顺序表类模板 Array 的定义，成员函数 binarySearch()的功能是：对已升序的数组折半查找某元素，并返回该元素的下标。完善下面程序，使得程序运行结果为：

原序列：bceghkmpwxz
查找字符 g 的下标=3

```
#include <iostream>
using namespace std;
const int  Max = 20;
template  < typename  T>  class  Array
{
    T  A[ Max ];              //存放数据元素的数组容器
   int  n;                    //实际元素个数
public:
    Array ( T a[ ], int m );
    int  binarySearch ( T  c );
     void print() { for ( int i=0; i<n; i++ ) cout<<A[ i ];  cout<<endl; }
};
template< typename  T > __Array < T > :: __Array ( T a [ ],  int  m )
{  n=m;
  for (  int i=0; i<m; i++ )  A[ i ] = a[ i ];  }
   template < typename  T >  int  Array <T>:: binarySearch (  T  c  )
   {  int  low=0,  high=n-1,  mid;
      while (__low <= high__ ){
          mid = ( low+high ) / 2;
          if ( A[ mid ] == c )__return mid__ ;
          if ( A[ mid ] < c )   low=mid+1;
           else  high=mid-1;
      }
       return  -1;
   }
   void main()
   {  char a[ ] = "bceghkmpwxz";     //已升序的字符序列
      Array  <char>  s( a,11 );
```

```
    cout << "原序列："; s.print();
    cout << "查找字符 g 的下标=" << s.binarySearch('g') << endl;
}
```

3. 用带表头结点的单链表实现对每个英文单词出现次数的统计。函数 count() 的功能是：对于一个英文单词，依次查找链表上的结点，若链表结点中已有该单词，则将该结点的 num 值加 1；否则在链首表头结点之后插入一个新结点，存放该单词，并置 num 为 1。

```
#include <string>
using namespace std;
class node
{
public:
  string word;
  int num;
  node * next;
};
node * count(node *head, string w)
{  node * p = head->next;
   while(p){
      if(p->word == w) { p->num++; break; }
      p = p->next;
   }
   if(p==0){
      p= new node();
      p->word =w; p->num =1;
      p->next=head->next;
      head->next=p;
   }
   return head;
}
```

4. 完善下面类 myString，实现字符串基本功能，其中运算符+=完成字符串链接。

```
#include <cstring>
using namespace std;
class myString
{
   char *str;
```



```
   int len;
public:
   myString(char *s=0)
   {
     if(s) { len=strlen(s); str=new char[len+1]; strcpy(str,s); }
     else { len=0; str=0; }
   }
   ~myString() { if(str) delete[] str; }
   myString & operator+=(myString &s)
   {
     if(s.str){
        char *p=str;
        if(p){
          str= new char[(len=s.len)+1];
          strcpy(str, p);
          strcat(str, s.str)
          delete [] str;
        }
        else {
          str=new char[(len=s.len)+1];
          strcpy(str, s.str);
        }
     }
     return *this;
   }
};
```

5 完善下面顺序栈的操作。

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

const int MAX=20;
class stack
{
public:
    int element[MAX]; //顺序栈空间
    int top; //top 为栈顶元素的下标，空栈时 top 为-1
};
void push(stack &s, int data) //入栈函数
```

```
{
    if ( s.top == MAX-1 ) {
        cout << "栈已满，无法入栈了!\n";
        exit(0);
    }
    s.element [s.top++] = data;
}
int pop( stack &s )    //出栈函数
{
  if ( s.top == -1 ) {
    cout << "栈已空，无法出栈了!\n";
    exit(0);
  }
    return s.element [ s.top-- ];
}
void setNull( stack &s ) //置栈为空栈
{
    s.top = ____-1____;
}
```