上海财经大学《程序设计基础》期末考试卷(A 卷)

参考答案

（ 2022 至 2023 学年 第 1 学期 ）

1. (循环控制。15 分）

编写程序计算三角函数 sin*x* 近似值。请根据 sin*x* 泰勒展开式，应用循环编写程序。

$$\sin x = \sum_{k=0}^{\infty} \frac{(-1)^k}{(2k+1)!} x^{2k+1} = x - \frac{1}{3!}x^3 + \frac{1}{5!}x^5 - \frac{1}{7!}x^7 + \cdots$$

当 $\left|\frac{(-1)^k}{(2k+1)!}x^{2k+1}\right| < 10^{-9}$ 时，循环结束。

程序如下：

```
#include <iostream>
using namespace std;
int main()
{
    double x, sinX = 0;
    cout << "Please input the value of x ): ";
    cin >>x;
    //请补充完成以下缺少的代码  15 分

    cout << "sin(" << x << ") = " << sinX << endl;
    return 0;
}
```

参考答案：

```
#include <iostream>
#include <cmath>
using namespace std;
int main()
{
   double x , sinX = 0;
   cout << "Please input the value of x: ";
   cin >> x;
   // 请补充完成以下缺少的代码  15 分
   double t = x;                                  //    2 分
   for(int k=1; fabs(t)>=1E-9; k++){              //    4 分
        sinX += t;                                //    3 分
        t *= x*x;                                 //    2 分
```

```
            t /= 2*k*(2*k+1);                    //    2 分
            t *= -1;                             //    2 分
        }
        cout << "sin(" << x << ") = " << sinX << endl;
        return 0;
}
```

2. (函数调用。 20 分)

编写名为 calculateStatistics 的函数，实现如下功能：计算整型数组元素中的平均值、方差，以及数组中大于等于平均值的元素个数和小于平均值的元素个数。假设数组 x，包含 N 个元素:

数组元素平均值： $$x\_mean = \sum_{i=1}^{N} x_i$$

数组元素的方差： $$x\_deviation = \frac{1}{N}\sum_{i=1}^{N}(x_i - x_{mean})^2$$

```
// 主程序
#include <iostream>
using namespace std;
const int NUM = 10;
int main()
{
   int x[NUM] = {12, 23, 41, 28, 19, 27, 38, 14, 52, 32};
   int greaterNum = 0, lessNum = 0;
   double x_mean =0, x_deviation = 0;
   x_mean = calculateStatistics(x, NUM, &x_deviation, greaterNum, lessNum);
   cout <<"数组元素平均值: " << x_mean << endl
        <<"数组元素的方差: " << x_deviation << endl
        <<"大于等于平均值的元素: "<< greaterNum << "个" << endl
        <<"小于平均值元素:"<< lessNum << "个" <<endl;
   return 0;
}

// 参考答案：
double calculateStatistics(int x[], int NUM, double *p_x_deviation,   //   5 分
                           int &greaterNum, int &lessNum){
   double sum = 0, mean;                                              //   2 分
   for(int i=0; i<NUM; i++)                                           //   3 分
      sum += x[i];
   mean = sum/NUM;                                                    //   1 分
```

```
    sum = 0;                                                    //  2 分
    for(int i=0; i<NUM; i++){                                   //  3 分
        sum += (x[i] - mean)*(x[i] - mean);
        x[i]>=mean ? greaterNum++ : lessNum++;
    }
    *p_x_deviation = sum/NUM;                                   //  2 分

    return mean;                                                //  2 分
}
```

3. 数组与字符串（改错题。20 分）

本题目的程序可以对输入的 3 个句子(包含空格的字符串)进行升序排序。并输出排好序的句子。程序正常运行示例如下：

```
Please input three sentences:
The Chinese people are hardworking people!
The Chinese people are selfless and enterprising people!
The Chinese people are brave to innovate!

Sorted sentences:
The Chinese people are brave to innovate!
The Chinese people are hardworking people!
The Chinese people are selfless and enterprising!
```

下面是实现上述功能的程序，请仔细阅读程序，找出其中错误并修改。

注意：

1）不能改变 main 函数中的数据定义：char sentences[MAXNUM][MAXLENGHT];

2）不能改变 stringSort 函数中的选择排序算法。

【评分标准】：以下共 9 个错误，根据发现、改正的错误数量得分。

改对　1 个　4 分

2 个　8 分

3 个　11 分

4 个　14 分

5 个　17 分

6 个及以上　20 分

参考答案：

```
1  #include <iostream>
2  #include <cstring>          //  string 修改为  cstring
3  const int MAXNUM = 3;
4  const int MAXLENGHT = 80;
```

```
5  using namespace std;
6
7  void stringSort(char name[MAXNUM][MAXLENGHT], int n);
8  int main(){
9      char sentences[MAXNUM][MAXLENGHT];
10     cout << "Please input three sentences:" << endl;
11     for(int i=0; i<MAXNUM; i++)
12         cin.getline(sentences[i], MAXLENGHT);
13
14     stringSort(sentences, MAXNUM);
15     for(int i=0; i<MAXNUM; i++)
16         cout << sentences[i] << endl;
17     return 0;
18 }
19
20 void stringSort(char str[MAXNUM][MAXLENGHT], int n){
21     char temp[MAXLENGHT];
22     for(int i=0; i<MAXNUM-1; i++){
23         int k = i;
24         for(int j=i+1; j<MAXNUM; j++)
25             if(strcmp(str[k], str[j])>0) k = j;
26         if(k!=i){
27             strcpy(temp, str[i]);
28             strcpy(str[i], str[k]);
29             strcpy(str[k], temp);
30         }
31     }
32 }
```

4. 指针与动态数组（改错题。 17 分）

下面程序功能包括：

1） 将 3 个字符串，每个字符串去除其中非数字和非英文字母的字符，并将小写字母转化为大写字母，然后进行连接合并；

2） 统计 3 个字符串中数字字符和英文字母的频次；

3） 输出连接合并后的字符串、以及 3 个字符串中数字字符和英文字母的频次。

例如：const char *str[3] = { "I love VC++ 2010!",
"Do you like Python 3.7?",
"Java SE18 is a good!"
};

去除非英文字母和非数字的字符  小写字母转化为大写字母，进行连接合并

mergestring：ILOVEVC2010DOYOULIKEPYTHON37JAVASE18ISGOOD
中英文字母和数字字符的频次：alphabet[ ]，digital[ ]

输出连接合并串  英文字母和数字字符的频次

Merged strings: ILOVEVC2010DOYOULIKEPYTHON37JAVASE18ISGOOD
数字字符的频次：0:2 1:2 2:1 3:1 7:1 8:1
英文字母的频次：A:2 C:1 D:2 E:3 G:1 H:1 I:3 J:1 K:1 L:2 N:1 O:6 P:1 S:2 T:1 U:1 V:3 Y:2
仔细阅读上述功能程序的实现，找出其中的错误，并（在不改变程序结构基础上）进行修改。

```
1  #include <iostream>
2  using namespace std;
3  #include <cstring>
4  const int MAXLENGTH = 3;
5  //去除非英文字母和非数字的字符，小写字母转化为大写字母，返回连接合并的字符串指针
6  char *MergeStatistics(const char *str[], int MaxLen, int digital[], int alphabet[]){
7      int mergedlen = 0;
8      for(int i=0; i<MaxLen; i++)
9          mergedlen += strlen(str[i]);
10     char *mergestring = new char(mergedlen+1);
11     int counter = 0;
12     for(int i=0; i<MaxLen; i++){
13         const char *ptr = str[i];
14         char c;
15         while((c = *ptr)!='/0'){
16             if(c>='A'&&c<='Z'|| c>='a'&&c<='z'){
17                 c = c -32;
18                 ++alphabet[c];
19                 mergestring[++counter] = c;
20             }
21             if(c>='0'&&c<='9'){
22                 ++digital[c];
23                 mergestring[++counter] = c;
```

```
            }
            ptr++;
        }
    }
    return mergestring;
}
int main(){
    const char *str[MAXLENGTH] = { "I love VC++ 2010!",
                                   "Do you like Python 3.7?",
                                   "Java SE18 is good!"
                                 };
    int digital[10] = {0};      // 存放数字字符频次的数组，
    int alphabet[26] = {0};     // 存放英文字母频次的数组
//去除非英文字母和非数字的字符，小写字母转化为大写字母，返回连接合并的字符串指针
    char *mergedStr = MergeStatistics(str, MAXLENGTH, digital, alphabet);
    cout << "Merged strings: " << mergedStr << endl;    // 输出连接合并的字符串
    for(int i=0; i<10; i++)     // 输出数字字符的频次
        if(digital[i]) cout << i << ":" << digital[i] <<"    ";
    cout << endl;
    for(int i=0; i<26; i++)    // 输出英文字母的频次
        if(alphabet[i]) cout << char(i+'A') << ":" << alphabet[i] <<" ";

    return 0;
}
```

【评分标准】：本题共 8 个错误，根据发现、改正的错误数量得分。

改对 1 个　4 分

2 个　8 分

3 个　11 分

4 个　14 分

5 个及以上　17 分

// 参考答案：

```
#include <iostream>
#include <cstring>
using namespace std;
const int MAXLENGTH = 3;
//去除非英文字母和非数字的字符，小写字母转化为大写字母，返回连接合并的字符串指针
char *MergeStatistics(const char *str[], int MaxLen, int digital[], int alphabet[]){
```

```
7     int mergedlen = 0;
8     for(int i=0; i<MaxLen; i++)
9         mergedlen += strlen(str[i]);
10    char *mergestring = new char[mergedlen+1];    //    把圆括号改为方括号
11    int counter = 0;
12    for(int i=0; i<MaxLen; i++){
13        const char *ptr = str[i];
14        char c;
15        while((c = *ptr)!='\0'){                          //    '/0'  改为  '\0'
16          if(c>='A'&&c<='Z'|| c>='a'&&c<='z'){
17              c = toupper(c);         //    或  if (c>='a'&&c<='z') c -=32;
18              ++alphabet[c - 'A']; // ++alphabet[c]改为++alphabet[c - 'A']
19              mergestring[counter++] = c;    // ++counter 改为 counter++
20          }
21          if(c>='0'&&c<='9'){
22              ++digital[c - '0'];       // ++digital[c]  改为++digital[c - '0']
23              mergestring[counter++] = c;    // ++counter 改为 counter++
24          }
25          ptr++;
26       }
27    }
28    mergestring[counter] = '\0';
29    return mergestring;
30 }
31 int main(){
32    const char *str[MAXLENGTH] = { "I love VC++ 2010!",
33                                   "Do you like Python 3.7?",
34                                   "Java SE18 is good!"
35                                 };
36    int digital[10] = {0};       //  存放数字字符频次的数组，
37    int alphabet[26] = {0};    //  存放英文字母频次的数组
38 //去除非英文字母和非数字的字符，小写字母转化为大写字母，返回连接合并的字符串指针
39    char *mergedStr = MergeStatistics(str, MAXLENGTH, digital, alphabet);
40    cout << "Merged strings: " << mergedStr << endl;    //  输出连接合并字符串
41    for(int i=0; i<10; i++)      //  输出数字字符的频次
42        if(digital[i]) cout << i << ":" << digital[i] <<"    ";
43    cout << endl;
44    for(int i=0; i<26; i++)    //  输出英文字母的频次
```

```
45          if(alphabet[i]) cout << char(i+'A') << ":" << alphabet[i] <<" ";
46      delete []mergedStr;       // 释放 new 申请的空间
47      return 0;
48  }
```

5. 递归 （10 分）

（1）

```
int gcd(int x, int y){                    ---------      1 分
    if(x%y==0)                            ---------      1 分
        return y;                         ---------      1 分
    else
        return gcd(y,x%y);                ---------      3 分
}
```

（2）y 除 x 的余数 x%y 总是小于 y，因此函数的第二个参数总是递减，因此总有一次调用时 y 整除 x（或是 y 递减过程中，或是当 y 变为 1 时），从而保证递归结束。

---------- 4 分

6. 参考答案：

（类。 18 分）

校园附近开张了一些汉堡连锁店。每个店铺的点餐系统都用动态数组对汉堡进行管理。其中汉堡是结构体 Burger，**采用接口（.h 文件）与实现（.cpp 文件）分离的多文件方法，创建一个汉堡商店类** BurgerShop，用一个动态数组（由 burger_arr 指针指向该动态数组）管理汉堡店中的所有汉堡。该动态数组以 5 个汉堡结构体为单位分配空间。例：汉堡店的汉堡从 5 增加到 6 时，动态数组空间从可以记录 5 个 Burger 扩充到可以记录 10 个 Burger；汉堡店每销售一个汉堡，从数组中删除该汉堡信息，并将汉堡总数 **burger_num** 减一。

请实现其拷贝构造函数、重载赋值操作符、重载 << 操作符、新增汉堡的 AddBurger 函数。

```
//BurgerShop.h 文件
#ifndef __BURGERSHOP_H
#define __BURGERSHOP_H
#define ALLOC_SPACE 5
#define MAX_PRICE 30
#define MAX_NUTRIENT 10

#include <iostream>
using namespace std;
struct Burger
{
```

```
    int price;
    int nutrient;
};

class BurgerShop
{
public:
    BurgerShop();                                          //构造函数
    BurgerShop(const BurgerShop& burgerShop);              //拷贝构造函数
    ~BurgerShop();                                         //析构函数
    void AddBurger(Burger burger);                         //新增一个汉堡
    void SellBurger(int index);                            //卖出一个汉堡
    BurgerShop& operator= (const BurgerShop& burgerShop);  //重载赋值操作符
    friend ostream & operator<<(ostream & os, const BurgerShop bp);
    //重载 << 操作符
    Burger & operator[ ](int index);     //重载 [] 操作符

private:
    Burger* burger_arr;
    int burger_num;
};
#endif //__BURGERSHOP_H
```

注： 调用 cout << burgerShopA; 在屏幕上的输出一个示例如下：

```
burger number: 8
burger 1-> price: 28 nutrient: 0
burger 2-> price: 15 nutrient: 2
burger 3-> price: 20 nutrient: 7
burger 4-> price: 21 nutrient: 4
burger 5-> price: 11 nutrient: 2
burger 6-> price: 23 nutrient: 4
burger 7-> price: 14 nutrient: 9
burger 8-> price: 19 nutrient: 2
```

```
#include <iostream>
#include <ctime>
#include <stdlib.h>
#include "BurgerShop.h"
using namespace std;

int main()
{
    srand(time(NULL));
    BurgerShop burgerShopA;
    for(int i=1;i<=8;i++)
```

```
    {
        int tmp = rand();
        Burger tmp_burger = { tmp%MAX_PRICE+1, tmp%(MAX_NUTRIENT+1)};
        burgerShopA.AddBurger(tmp_burger);
    }
    cout << burgerShopA <<endl;
    BurgerShop burgerShopB=burgerShopA;
    int no;
    cout << "which one do you like to buy?"<< endl;
    cin >> no;
    cout << "you want to buy :" << "(" << burgerShopA[no-1].price<<","
         <<   burgerShopA[no-1].nutrient << ")"<<endl;
    burgerShopA.SellBurger(no);

    cout << "The burgers in burgerShopA After the sell:"<< endl
         << burgerShopA <<endl;
    cout << "The burgers in burgerShopB :"<< endl << burgerShopB <<endl;

    //交换 burgerShopA 与  burgerShopB
    BurgerShop burgerShopX=burgerShopA;
    burgerShopA=burgerShopB;
    burgerShopB=burgerShopX;
    cout << "After interchange:" <<endl
        << "The burgerShopA:"<< endl << burgerShopA <<endl;
    cout << "The burgerShopB :"<< endl << burgerShopB <<endl;

    return 0;
}

// BurgerShop.cpp 文件
#include <stdlib.h>
#include <cstring>
#include <cmath>
#include <iostream>
#include "BurgerShop.h"
using namespace std;

BurgerShop::BurgerShop()
{
    burger_arr = NULL;
    burger_num = 0;
}

BurgerShop::BurgerShop(const BurgerShop & burgerShop)
```

```
{
    burger_arr = new Burger [burgerShop.burger_num];                    --------- 1 分
    for(int i=0;i<burgerShop.burger_num;i++)
        burger_arr[i] = burgerShop.burger_arr[i];                       --------- 1 分
    burger_num=burgerShop.burger_num;                                   --------- 1 分
}

void BurgerShop::AddBurger(Burger burger)
{
    burger_num++;                                                       --------- 1 分
    //realloc memory
    if (burger_num%ALLOC_SPACE==1)                                      --------- 1 分
    {
        Burger* tmp_arr = new Burger [burger_num+ALLOC_SPACE-1];
                                                                        --------- 1 分
        //manually copy
        for(int i=0;i<burger_num-1;i++)
            tmp_arr[i] = burger_arr[i];                                 --------- 1 分

        if (burger_arr!=NULL)
            delete [] burger_arr;                                       --------- 1 分
        burger_arr = tmp_arr;                                           --------- 1 分
    }
    //add new burger
    burger_arr[burger_num-1] = burger;                                  --------- 1 分
}


ostream & operator<<(ostream & os, const BurgerShop bp){
    os <<"burger number: " << bp.burger_num <<endl;                     --------- 1 分
    for(int i=0;i<bp.burger_num;i++)                                    --------- 1 分
        cout<<"burger "<<i+1<<"-> price: "
            <<bp.burger_arr[i].price<<" nutrient: "
            <<bp.burger_arr[i].nutrient<<endl;                          ---------- 1 分
    return os;                                                          ---------- 1 分
}

BurgerShop& BurgerShop::operator=(const BurgerShop &burgerShop)
{
    if (this == &burgerShop)
        return *this;                                                   ---------- 1 分
    burger_num = burgerShop.burger_num;
    burger_arr = new Burger [burgerShop.burger_num];                    ---------- 1 分
    for(int i=0;i<burgerShop.burger_num;i++)
```

```
            burger_arr[i] = burgerShop.burger_arr[i];                    ---------- 1 分
        return *this;                                                    ---------- 1 分
}

void BurgerShop::SellBurger(int index)
{
        if (index>burger_num II index < 1)
        {
                cout<<"no such burger\n";
                return ;
        }
        for(int i=index-1;i<burger_num-1;i++)
                burger_arr[i] = burger_arr[i+1];
        burger_num--;
}

Burger & BurgerShop::operator[ ](int index){
        return burger_arr[index];
}

BurgerShop::~BurgerShop()
{
        if (burger_arr!=NULL)
                delete [] burger_arr;
}
```