**实验2：复习与拓展2-字符串学习部分**

**类与对象初步**

**姓名\_\_\_金宣成\_\_\_班级\_\_\_计科03\_\_\_学号\_202203151310\_**

* **请阅读此说明：实验2满分100分，附加分20分；做完实验后请按要求将代码和截图贴入该文档。然后将此文档、源代码文件（.cpp）打包上传到学习通。**

**1、字符串-基础题：（10分）**

编写函数，完成和实现拷贝字符串n个字符的功能。

函数原型为： bool mystrncpy(char \*to , char \*from,unsigned startpos,unsigned len);

表示需要从from串中的startpos位置开始拷贝len个字符串内容到to串中，返回拷贝是否成功的信息(失败情况如位置无效，长度无效等)。

* **实验要求：**

**❶ 按照给定的函数原型定义并实现函数，并且设计主函数进行函数功能的测试。**

* **实验提交：**

**将完整的源代码和测试截图 粘贴在下面。**

* **测试参考：假设from串内容为：abcdefghijklmn**

**测试时使用可以参考如下可能的调用样本：**

**mystrncpy(to, from, 0,0);**

**mystrncpy(to, from, 3,4);**

**mystrncpy(to, from, 3,20);**

* **源代码粘贴处：**

#include <iostream>

#include <cstring>

#include <iomanip>

#include <cmath>

#include <algorithm>

using namespace std;

bool mystrncpy(char \*to , char \*from,unsigned startpos,unsigned len);

int main(){

    char from[15] = "abcdefghijklmn";

    char \* to;

    int n ,startpos = 0,len = 0;

    to = new char [n];

    cout <<  "输入目标数组的长度：";

    cin >> n;

    cout << endl << "输入开始位置和拷贝长度： ";

    cin >> startpos >> len;

    if(mystrncpy(to,from,startpos,len)){

            cout << "成功拷贝，结果为："  << to;

    }else{

        cout << "error" << endl ;

    }

    delete [] to;

    return 0;

}

bool mystrncpy(char \*to , char \*from,unsigned startpos,unsigned len){

    if(startpos + len > strlen(from) or len > strlen(to)){

        return 0;

    }

    int pos = 0;

    for(int i = startpos;i < startpos + len;i++){

        to[pos++] = from [i];

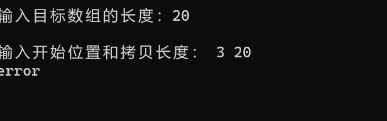
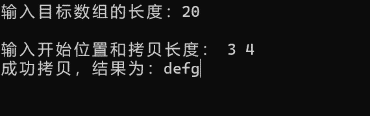
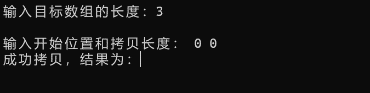
    }

    to[pos] = '\0';

    return 1;

}

* **程序测试截图：**

****

**2、字符串-进阶题：（20分+ 5分）**

从键盘接受一篇英文文章，统计其包含大写字母、小写字母、数字字符、空格、其他字符的个数，并统计文章的单词数（单词使用空格分隔。）

* **实验要求：**

**❶ 按照需求完成程序的设计和实现。**

* **实验提交：**

**将完整的源代码和测试截图 粘贴在下面。**

* **可选-实验拓展：考虑程序功能拓展到处理多篇英文文章。（附加 5分），请将设计完成的程序粘贴在后面。**
* **源代码粘贴处：**

#include <iostream>

#include <cstring>

#include <string>

#include <iomanip>

#include <cmath>

#include <algorithm>

using namespace std;

int main(){

    string essay;

    long long upnum = 0,downnum = 0,nnum = 0,blanum = 0,onum = 0,row = 0;

    while (getline(cin,essay))

    {

        row ++;

        for(int i = 0;i < essay.size();i++){

            if(essay[i] >= 'A' and essay[i] <= 'Z'){

                upnum++;

            }else if(essay[i] >= 'a' and essay[i] <= 'z'){

                downnum++;

            }else if(essay[i] >= '0' and essay[i] <= '9'){

                nnum++;

            }else if(essay[i] == ' '){

                blanum++;

            }else{

                onum++;

            }

        }

    }

    cout << "uppercase:" << upnum << endl;

    cout << "lowercase:" << downnum << endl;

    cout << "number:" << nnum << endl;

    cout << "space:" << blanum << endl;

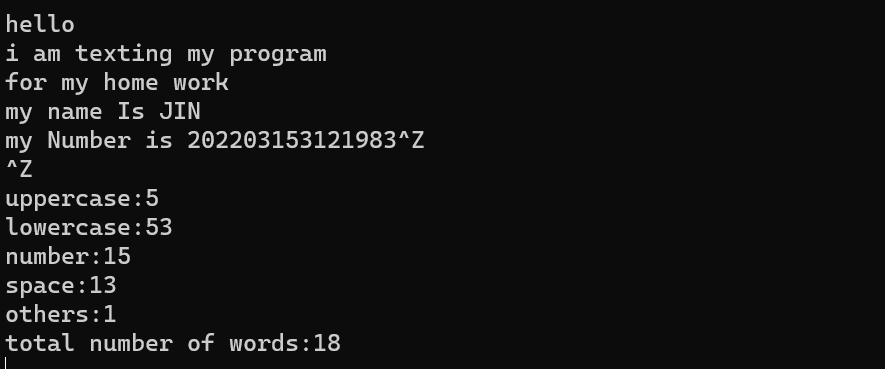
    cout << "others:" << onum << endl;

    cout << "total number of words:" << blanum + row<< endl;

    return 0;

}

* **程序测试截图：**

****

* **拓展源代码粘贴处（可选）：（上述方案已经可以实现）**

#include <iostream>

#include <cstring>

#include <string>

#include <iomanip>

#include <cmath>

#include <algorithm>

using namespace std;

int main(){

    string essay;

    long long upnum = 0,downnum = 0,nnum = 0,blanum = 0,onum = 0,row = 0;

    while (getline(cin,essay))

    {

        row ++;

        for(int i = 0;i < essay.size();i++){

            if(essay[i] >= 'A' and essay[i] <= 'Z'){

                upnum++;

            }else if(essay[i] >= 'a' and essay[i] <= 'z'){

                downnum++;

            }else if(essay[i] >= '0' and essay[i] <= '9'){

                nnum++;

            }else if(essay[i] == ' '){

                blanum++;

            }else{

                onum++;

            }

        }

    }

    cout << "uppercase:" << upnum << endl;

    cout << "lowercase:" << downnum << endl;

    cout << "number:" << nnum << endl;

    cout << "space:" << blanum << endl;

    cout << "others:" << onum << endl;

    cout << "total number of words:" << blanum + row<< endl;

    return 0;

}

**3、字符串-高阶题：（20分+10分）**

使用字符串/数组实现超级整数的存储和基本计算，比如完成一个表示范围可能达30000位的超级整数。要求实现其加法、减法、乘法运算。每一种运算定义成函数，并编写主函数测试你的超级整数。自行设计输入输出格式。比如，输入形式为： “数字1,数字2,运算符号” 例 ”123456788,3456789012345,+”使逗号分隔，需要判断表达式的合法性，并计算结果。

要求：，并讨论：1、选择使用数组或者使用字符串的理由；2、考虑基本存储类型的选择：char或者int？3、讨论你的采用的策略能表示超级整数的极限（可以到多大）

* **实验要求：**

**❶ 按照需求完成程序的基本功能设计和实现。**

**可选-设计并实现两种方式的输入处理：计算需要的大数可以由随机数生成。小数可以提供用户输入测试（基本功能）。（附加5分）**

**❷ 可选-思考： 1、考虑基本存储类型的选择：char或者int？选择使用数组或者使用字符串的理由；2、讨论你的采用的策略能表示超级整数的极限（可以到多大）（附加5分）**

* **实验提交：**

**将完整的源代码和测试截图 粘贴在下面。**

* **源代码黏贴处：**

#include <iostream>

#include <cstring>

#include <string>

#include <iomanip>

#include <cmath>

#include <algorithm>

using namespace std;\

string reverse(string str){

    int n = str.length();

    for(int i=0; i<n/2; i++){

        swap(str[i], str[n-i-1]);

    }

    return str;

}

string myplus(string &a,string &b){

    string res(a.size() > b.size() ? a.size() + 1: b.size() + 1, '0');

    int p1 = a.size() - 1,p2 = b.size() - 1,p3 = -1;

    for(;p1 >= 0 and p2 >= 0;p1--,p2--){

        int sum = 0;

        sum += a[p1] - '0';

        sum += b[p2] - '0';

        sum += res[++p3] - '0';

        if(sum >= 10){

            res[p3] = (char)(sum % 10 + 48);

            res[p3 + 1] = (char)(sum / 10 + 48);

        }else{

            res[p3] = (char)(sum + 48);

        }

    }

    if(p1 >= 0){

        for(int i = p1;i >= 0;i--){

            int sum = 0;

            sum += a[i] - '0';

            sum += res[++p3] - '0';

            if(sum >= 10){

                res[p3] = (char)(sum % 10 + 48);

                res[p3 + 1] = (char)(sum / 10 + 48);

            }else{

                res[p3] = (char)(sum + 48);

            }

        }

    }else if(p2 >= 0){

        for(int i = p2;i >= 0;i--){

            int sum = 0;

            sum += b[i] - '0';

            sum += res[++p3] - '0';

            if(sum >= 10){

                res[p3] = (char)(sum % 10 + 48);

                res[p3 + 1] = (char)(sum / 10 + 48);

            }else{

                res[p3] = (char)(sum + 48);

            }

        }

    }

    return reverse(res);

}

string remove0(string str) {

    int i = 0;

    while (i < str.size() - 1 && str[i] == '0') {

        i++;

    }

    return str.substr(i);

}

string myminus(string &a,string &b){

    int flag = 1;

    if(a < b){

        flag = -1;

        swap(a,b);

    }

    b = string(a.size() - b.size(), '0') + b;//给b高位补0，使ab等长

    string res(a.size(), '0');

    int borrow = 0;

    for(int i = a.size() - 1;i >= 0;i--){

        int diff = 0;

        diff = a[i] - b[i] - borrow;

        if(diff>= 0){

            borrow = 0;

        }else{

            diff += 10;

            borrow = 1;

        }

        res[i] = char(diff + '0');

    }

    if (flag == -1)

    {

        res = '-' + remove0(res);

    }

    return remove0(res) ;

}

string mymultiply(string &a,string &b){

    if(a < b){ //保证a比b长

        swap(a,b);

    }

    string res(a.size() + b.size(), '0');

    for(int i = a.size() - 1;i >=0;i--){

        int carry = 0;

        for(int j = b.size() - 1;j >=0 ;j--){

            int mul = (b[j] -'0') \* (a[i] - '0');

            int sum = mul + carry + res[i + j + 1] - '0';

            res[i + j + 1] = (sum % 10) + '0';

            carry = sum / 10;

        }

        res[i] += carry; // 加上进位

    }

    return remove0(res);

}

int main(){

    string a;

    string b;

    char func;

    cin >> a >> b >> func;

    if(func == '+'){

        cout << remove0(myplus(a,b)) << endl;

    }else if(func == '-'){

        cout << myminus(a,b) << endl;

    }else if(func == '\*'){

        cout << mymultiply(a,b) << endl;

    }else{

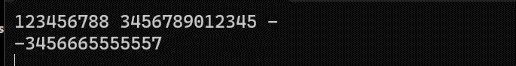
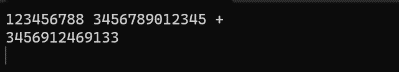
        cout << "error" << endl;

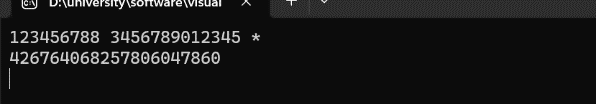
    }

    return 0;

}

* **程序测试截图：**



****

* **输入设计拓展（可选）：(上述代码已经实现输入由用户自定，下述增加了系统自动初始化功能)**

#include <iostream>

#include <cstring>

#include <string>

#include <iomanip>

#include <cmath>

#include <algorithm>

using namespace std;\

string reverse(string str){

    int n = str.length();

    for(int i=0; i<n/2; i++){

        swap(str[i], str[n-i-1]);

    }

    return str;

}

string myplus(string &a,string &b){

    string res(a.size() > b.size() ? a.size() + 1: b.size() + 1, '0');

    int p1 = a.size() - 1,p2 = b.size() - 1,p3 = -1;

    for(;p1 >= 0 and p2 >= 0;p1--,p2--){

        int sum = 0;

        sum += a[p1] - '0';

        sum += b[p2] - '0';

        sum += res[++p3] - '0';

        if(sum >= 10){

            res[p3] = (char)(sum % 10 + 48);

            res[p3 + 1] = (char)(sum / 10 + 48);

        }else{

            res[p3] = (char)(sum + 48);

        }

    }

    if(p1 >= 0){

        for(int i = p1;i >= 0;i--){

            int sum = 0;

            sum += a[i] - '0';

            sum += res[++p3] - '0';

            if(sum >= 10){

                res[p3] = (char)(sum % 10 + 48);

                res[p3 + 1] = (char)(sum / 10 + 48);

            }else{

                res[p3] = (char)(sum + 48);

            }

        }

    }else if(p2 >= 0){

        for(int i = p2;i >= 0;i--){

            int sum = 0;

            sum += b[i] - '0';

            sum += res[++p3] - '0';

            if(sum >= 10){

                res[p3] = (char)(sum % 10 + 48);

                res[p3 + 1] = (char)(sum / 10 + 48);

            }else{

                res[p3] = (char)(sum + 48);

            }

        }

    }

    return reverse(res);

}

string remove0(string str) {

    int i = 0;

    while (i < str.size() - 1 && str[i] == '0') {

        i++;

    }

    return str.substr(i);

}

string myminus(string &a,string &b){

    int flag = 1;

    if(a < b){

        flag = -1;

        swap(a,b);

    }

    b = string(a.size() - b.size(), '0') + b;//给b高位补0，使ab等长

    string res(a.size(), '0');

    int borrow = 0;

    for(int i = a.size() - 1;i >= 0;i--){

        int diff = 0;

        diff = a[i] - b[i] - borrow;

        if(diff>= 0){

            borrow = 0;

        }else{

            diff += 10;

            borrow = 1;

        }

        res[i] = char(diff + '0');

    }

    if (flag == -1)

    {

        res = '-' + remove0(res);

    }

    return remove0(res) ;

}

string mymultiply(string &a,string &b){

    if(a < b){ //保证a比b长

        swap(a,b);

    }

    string res(a.size() + b.size(), '0');

    for(int i = a.size() - 1;i >=0;i--){

        int carry = 0;

        for(int j = b.size() - 1;j >=0 ;j--){

            int mul = (b[j] -'0') \* (a[i] - '0');

            int sum = mul + carry + res[i + j + 1] - '0';

            res[i + j + 1] = (sum % 10) + '0';

            carry = sum / 10;

        }

        res[i] += carry; // 加上进位

    }

    return remove0(res);

}

int main(){

    string a;

    string b;

    srand(time(NULL));

    for (int i = 0; i < 20; i++) {

        char randomChar = (rand() % 10) + '0';

        b += randomChar;

    }

    b = remove0(b);

    char func;

    cout <<  "please input only one  number while another is :" << b << endl;

    cin >> a >> func;

    cout << ' ' << b << '=' << endl;

    if(func == '+'){

        cout << remove0(myplus(a,b)) << endl;

    }else if(func == '-'){

        cout << myminus(a,b) << endl;

    }else if(func == '\*'){

        cout << mymultiply(a,b) << endl;

    }else{

        cout << "error" << endl;

    }

    return 0;

}

* **思考（可选）：**

1. **我选择使用字符串来对大整数进行储存和运算，一方面是因为大整数由多位数组储存，可能超过了整数储存的范围，而储存至字符串就不会有这样的问题，另一方面是因为数组可以用下标来访问每一个位，这样再进行计算的时候可以方便的对每一位进行加减乘的运算。**
2. **用string来储存整数的话，理论上来说取决与我的内存大小，只要有足够的内存，那string中可以存储无限大的数并进行计算**

**4、类与对象初步-基础题：（10分）**

**根据自己的掌握情况重做课堂例题：游泳池造价计算 + 停车收费系统**

* **注意练习和熟悉多文件装配模型：**

**一个工程项目包含：自定义类库 xxx.hpp+ xxx.cpp 以及测试的主程序 maintest.cpp**

**如 游泳池造价计算 包含 pool.hpp , pool.cpp , pooltest.cpp**

**停车场收费系统 包含 time.hpp ,time.cpp , ParkingCost.cpp**

**5、停车场计费系统的升级思考：（40分+5分）**

**以课堂材料给出的停车场系统为基础考虑(见学习通课件材料)，如果 停车收费规则改为：**

* **1小时内免费；**
* **1小时以上：**

**超过15分钟不到30分钟按半小时收费；**

**超过30分钟不到1小时按1小时收费；**

**超过1小时按每小时4元收费；**

* **超过1天，每天按30元收费；停车超过1个月的，停车费打9折；超过1年的停车费打8折。**
* **实验要求：**

**❶考虑修改Time的设计，改为DateTime，补充数据成员年、月、日，补充或者调整类内成员函数的设计，并实现它。采用你实现的新的DateTime进行停车收费的测试。并将完整的源代码和测试截图黏贴在下面。（20分）**

**源代码只需要黏贴类定义部分即可（DateTime的声明和实现）**

**❷考虑另一种程序调整模型，补充Date类，数据成员部分为：年、月、日；成员函数部分可以仿照原来Time类的功能设计。采用补充的新类Date和原来的Time类一起工作完成停车收费程序。（20分）**

**\*并思考：在新的收费规则下，两种程序调整的策略差异导致调整的工作量不同，你更倾向哪一种？给出你的理由（附加5分）**

**❶**

* **源代码粘贴处：**

**(1)hpp部分**

class DateTime

{

public:

    void show\_time();

    void set\_time();

    long normalize1()  const;//以小时

    long normalize2() const;//天数

    double diff1(const DateTime& T);//以小时为单位来储存的diff

    int diff2(DateTime& T);//以天为单位来储存的diff

private:

    int year;

    int month;

    int date;

    int hour;

    int minute;

};

**(2)cpp部分**

#include "DateTime.h"

#include <iostream>

using namespace std;

void DateTime::set\_time(){

    do{

        cin >> year >> month >> date >> hour >> minute ;

    }while(month < 0 or month > 31 or hour < 0 or hour > 24);

}

void DateTime::show\_time(){

    cout << year << ':' << month << ':' << date << ':' << hour << ':' << minute << endl;

}

long DateTime::normalize1() const {

    return hour \* 60 + minute;

}

long DateTime::normalize2() const {

    return (year - 2020) \* 365 + month \* 30 + date;

}

double DateTime::diff1(const DateTime& T){

    long d = normalize1() - T.normalize1() - 60;

    int h = d / 60,m = d % 60;

    if(d < 0)return 0;

    if(m < 15) return h;

    if(m >= 15 and m <= 30) return h + 0.5;

    if(m > 30) return h + 1;

}

int DateTime::diff2(DateTime& T){

    return normalize2() - T.normalize2();

}

**(3)main.cpp部分**

#include "DateTime.h"

#include <iostream>

using namespace std;

double charge(double,double);

int main(){

    DateTime arrive,leave;

    arrive.set\_time();

    leave.set\_time();

    cout << "arrive time:" ;

    arrive.show\_time();

    cout << "leave time:" ;

    leave.show\_time();

    double Fee = charge(leave.diff1(arrive),leave.diff2(arrive));

    cout << "The toltal Fee is: " << Fee << endl;

    return 0;

}

double charge(double Time1,double Time2){

    double sum = 0;

    sum = Time1 \* 4 + Time2 \* 30;

    if(Time2 > 365){

        return sum \* 0.8;

    }else if(Time2 > 30){

        return sum \* 0.9;

    }else{

        return sum;

    }

}

* **程序测试截图：**

****

**❷**

* **源代码粘贴处：**

1. **hpp code:**

class Time

{

public:

    void show\_time();

    void set\_time();

    long normalize()  const;

    double diff(const Time& T);

private:

    int hour;

    int minute;

};

class Date

{

public:

    void set\_date();

    void show\_date();

    long normalize() const;

    double diff(const Date& T);

private:

    int year;

    int month;

    int date;

};

1. **cpp code:**

#include "DateTime1 .h"

#include <iostream>

 using namespace std;

void Time::set\_time(){

    do{

        cin >> hour >> minute ;

    }while( hour < 0 or hour > 24 or minute > 60 or minute < 0);

}

void Time::show\_time(){

    cout << hour << ':' << minute << endl;

}

//time normalize

long Time::normalize() const {

    return hour \* 60 + minute;

}

double Time::diff(const Time& T){

    long d = normalize() - T.normalize() - 60;

    int h = d / 60,m = d % 60;

    if(d < 0)return 0;

    if(m < 15) return h;

    if(m >= 15 and m <= 30) return h + 0.5;

    if(m > 30) return h + 1;

}

void Date::set\_date(){

    cin >> year >> month >> date;

}

void Date::show\_date(){

    cout << year << ';' << month << ';' << date << endl;

}

//date normalize

long Date::normalize() const{

    return (year - 2020) + month \* 30 + date;

}

double Date::diff(const Date& T){

    return normalize() - T.normalize();

}

1. (3)**main code:**

#include "DateTime1 .h"

#include <iostream>

using namespace std;

double Fee(double a,double b){

    double sum = 0;

    sum = b \* 4 + a \* 30;

    if(a > 365)return sum \* 0.8;

    if(a > 30) return sum \* 0.9;

    return sum;

}

int main(){

    Time arrive1,leave1;

    Date arrive2,leave2;

    arrive2.set\_date();

    arrive1.set\_time();

    leave2.set\_date();

    leave1.set\_time();

    cout << "arrive date：";

    arrive2.show\_date();

    cout << endl <<"arrive time: ";

    arrive1.show\_time();

    cout << "leave date: ";

    leave2.show\_date();

    cout << endl << "leave time: ";

    leave1.show\_time();

    cout << "leavedate:" << leave2.diff(arrive2) << "days  and  leavetime:" << leave1.diff(arrive1) << "hours" << endl;

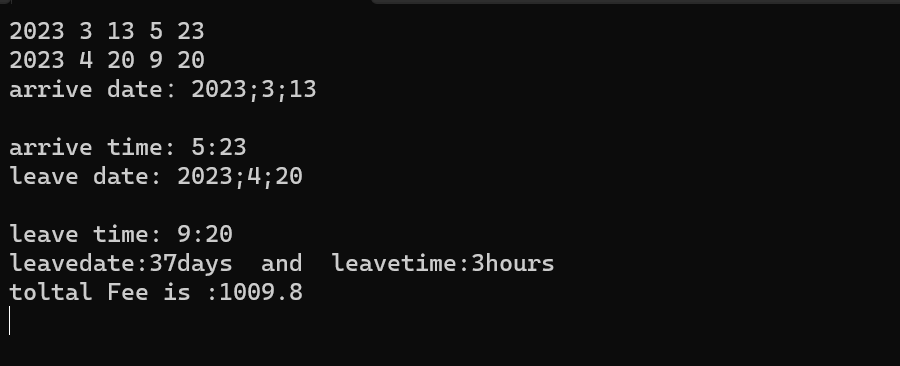
    double F = Fee(leave2.diff(arrive2) , leave1.diff(arrive1));

    cout << "toltal Fee is :" << F << endl;

    return 0;

}

* **程序测试截图：**

****

**\*附加思考：**

**我更倾向于前面一种，因为前面那一种从程序员的角度来说将与时间有关的所有变量都放入了一个类当中。这样在编辑main函数时，只需要定义一个时间有关的变量即可，而不用把time和date两个变量分开来定义，在实现时也要分开来算，比较麻烦。**