НИЖЕГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ

УНИВЕРСИТЕТ

ИМ. Р.Е. АЛЕКСЕЕВА

Кафедра "Вычислительные системы и технологии"

**ПРОГРАММИРОВАНИЕ**

**Отчёт**

**по лабораторной работе № 4**

Наследование, перегрузка операций

Вариант № 4

Выполнил студент группы 19-ИВТ-3

Сухоруков Валерий Алексеевич

«15» апреля 2020 г.

Провел ст. Преподаватель кафедры ВСТ

Мартынов Д.С.

«\_\_\_» \_\_\_\_\_\_\_ 2020 г.

Нижний Новгород 2020

Код на языке С++

#include<iostream>

#include<iomanip>

#include<fstream>

#include <stdlib.h>

using std::cout;

using std::cin;

using std::endl;

using std::ifstream;

using std::ofstream;

using std::setw;

using std::ios;

char logo[]=

"\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n"

" \_\_\_\_\_\_ \_\_\_\_\_\_ \n"

" \\ / /\\ | | | | \\ / \n"

" \\ / / \\ | | |\_\_\_\_\_| \\ / \n"

" \\ / / \\ | |\_\_\_\_\_ | \\ \\ / \n"

" \\ / /------\\ | | | \\ / \n"

" \\/ / \\ |\_\_\_\_\_ |\_\_\_\_\_ | \\ / \n"

" \n"

"\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n";

int myLen(const char\* str) {

int res = 0;

for (res; str[res] != '\0'; res++);

return res;

}

int myCmp(const char\* str1, const char\* str2) {

int res=0;

int len=myLen(str1);

if(myLen(str2)<len){len=myLen(str2);}

for(int i=0;i<len;i++){

if(str1[i]>str2[i]){res=1;break;}

if(str1[i]<str2[i]){res=-1;break;}

}

if((res==0) && (myLen(str1)>myLen(str2))){res=1;}

if((res==0) && (myLen(str1)<myLen(str2))){res=-1;}

return res;

}

class TCharArray {

protected:

char\* buf;

int size;

public:

TCharArray(void){

size=10;

buf=new char [size];

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

buf[i]='0';}

}

TCharArray(char\* str) {

size = myLen(str);

buf = new char[size + 1];

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

buf[i] = str[i];

buf[size] = '\0';

return;

}

TCharArray(int len){

size=len;

buf=new char [size];

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

buf[i]=0;}

}

~TCharArray(void) {

delete[] buf;

buf = nullptr;

size = 0;

return;

}

char& at(int index) {

if (index >= 0 && index < size)

return buf[index];

else {

cout << "Appeal to a non-existent element" << "/n";

exit(-1);

}

}

void setBuf(char\* str) {

if (buf == nullptr) {

size = myLen(str);

buf = new char[size + 1];

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

buf[i] = str[i];

buf[size] = '\0';

}

else {

delete[] buf;

size = myLen(str);

buf = new char[size + 1];

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

buf[i] = str[i];

buf[size] = '\0';

}

return;

}

int length(void) {

return size;

}

char& operator[](int index) {

return buf[index];

}

};

class String : public TCharArray {

public:

String(void) : TCharArray() {};

String(char\* str) : TCharArray(str) {};

~String(void) {

delete[] buf;

buf = nullptr;

size = 0;

return;

}

char\* getStr(void) {

return buf;

}

String operator+(String& str) {

String tmp;

tmp.size = size + str.size;

tmp.buf = new char[size + str.size + 1];

int i = 0;

int j = size;

for (i; i < size; i++) tmp[i] = buf[i];

for (i = 0; i < str.size; i++, j++) tmp[j] = str[i];

tmp[size + str.size] = '\0';

return tmp;

}

bool operator==(String& str) {

if(myCmp(buf,str.buf)==0){

return true;

}else{

return false;

}

}

bool operator!=(String& str) {

if(myCmp(buf,str.buf)==0){

return false;

}else{

return true;

}

}

bool operator>(String& str) {

if(myCmp(buf,str.buf)==1){

return true;

}else{

return false;

}

}

bool operator<(String& str) {

if(myCmp(buf, str.buf) == -1){

return true;

}else{

return false;

}

}

};

class flight {

private:

int Number;

String Name;

String Type;

double cost;

double length;

public:

flight() {

Number = 0;

cost= 0.0;

length = 0.0;

return;

}

~flight() { };

void setName(char\* nm) {

Name.setBuf(nm);

return;

}

void setType(char\* tp) {

Type.setBuf(tp);

}

void setNumber(int \_Number) {

Number = \_Number;

return;

}

void setCost(double \_Cost) {

cost = \_Cost;

return;

}

void setLength(double \_Length) {

length = \_Length;

return;

}

char\* getName(void) {

return Name.getStr();

}

char\* getType(void) {

return Type.getStr();

}

int getNumber(void) {

return Number;

}

double getCost(void) {

return cost;

}

double getLength(void) {

return length;

}

};

int main(int argc, char\*\* argv)

{

cout<<"\x1B[36m"<<logo<<endl;

if(((argc==2)&&((myCmp(argv[1],"-h")==0)||(myCmp(argv[1],"--help")==0)))||(argc==1)){

cout<<"\x1B[32m"<<"Аргументы:"<<endl;

cout<<"-h or --help для вызова подсказки"<<"\n";

cout<<"-c <с> [file\_name] для создания таблицы и записи её в файл (N строк)."<<endl;

cout<<"-r <N> [file\_name] для вывода таблицы из файла на экран (не более N строк)."<<endl;

}

if((argc==4) &&(myCmp(argv[1],"-c")==0)&&(atoi(argv[2])!=0)){

ofstream fp(argv[3],ios::binary);

if(fp.is\_open()) {

int N=atoi(argv[2]);

if(N<=0){cout<<"Eror, n<=0"<<endl;}

flight\* s=new flight [N];

int number;

char name[20];

char type[20];

double co;

double len;

for(int i=0;i<N;i++){

cout<<"\x1B[35m"<<"Рейс "<<i+1<<endl;

cout<<"Введите номер рейса"<<endl;

cin>>number;

cout<<"Введите название рейса"<<endl;

cin>>name;

cout<<"Введите тип самолёта"<<endl;

cin>>type;

cout<<"Введите стоимость билета"<<endl;

cin>>co;

cout<<"Введите протяжённость рейса"<<endl;

cin>>len;

s[i].setNumber(number);

s[i].setName(name);

s[i].setType(type);

s[i].setCost(co);

s[i].setLength(len);

fp<<setw(20)<<s[i].getNumber()<<setw(20)<<s[i].getName()<<setw(20)<<s[i].getType()<<setw(20)<<s[i].getCost()<<setw(20)<<s[i].getLength()<<endl;

}

fp.close();

delete []s;

}

}

if((argc==4) &&(myCmp(argv[1],"-r")==0)&&(atoi(argv[2])!=0)){

ifstream fp(argv[3],ios::binary);

if(fp.is\_open()) {

int N=atoi(argv[2]);

if(N<=0){cout<<"Eror, n<=0"<<endl;}

flight\* s=new flight [N];

int number;

char name[20];

char type[20];

double co;

double len;

cout<<"\x1B[35m"<<setw(15)<<"number"<<setw(15)<<"name"<<setw(15)<<"type"<<setw(15)<<"cost"<<setw(15)<<"length"<<endl;

for(int i=0;i<N;i++){

fp>>number>>name>>type>>co>>len;

s[i].setNumber(number);

s[i].setName(name);

s[i].setType(type);

s[i].setCost(co);

s[i].setLength(len); cout<<setw(15)<<s[i].getNumber()<<setw(15)<<s[i].getName()<<setw(15)<<s[i].getType()<<setw(15)<<s[i].getCost()<<setw(15)<<s[i].getLength()<<endl;

}

fp.close();

delete []s;

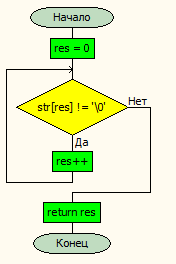
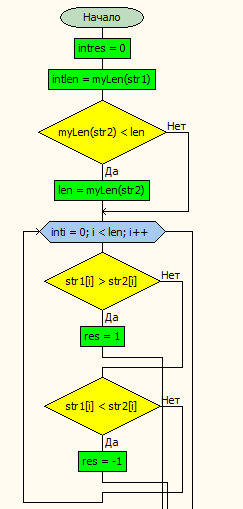
}

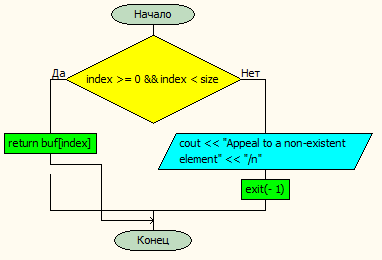
}

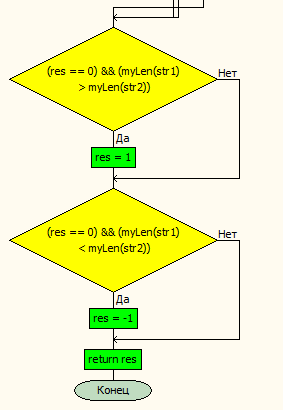
return 0;

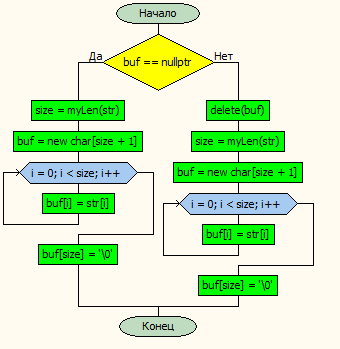
}

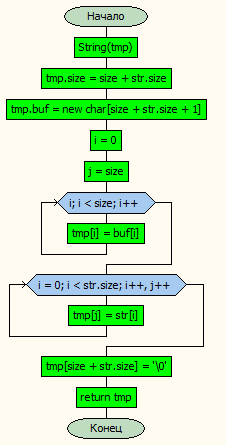
Блок-схема

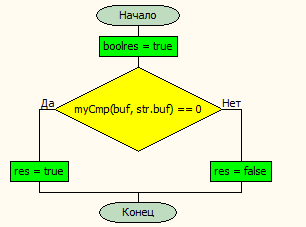
Функция myLen Функция myCmp

Функция at



Функция setBuf Оператор +



Оператор == (аналогично операторы != > < )

Алгоритм самой программы аналогичен коду в 3 лабораторной работе, поэтому его блок-схемы и псевдокод описаны не будут.

Псевдокод

Алгоритм myLen (Аргументы: str: Символьный массив, Результат: res: Целое)

Начало

|res=0

|Для (res; str [res]!=’\0’; res++)

|Всё-цикл

|Вернуть res

Конец

Алгоритм myCmp (Аргументы: str1, str2: Символьный массив, Результат: res: Целое)

Начало

|res=0

| Целое: len,i

|len=mylen(str1)

|Если (myLen (str2)<len) то

| |len=myLen (str2)

|Всё-если

|Для <i=0;i<len;i++>

| |Если (str1 [i]>str2 [i]) то

| | | res=-1

| | |Всё-цикл

| |Всё-если

| |Если (str1 [i]<str2 [i]) то

| | | res=1

| | |Всё-цикл

| |Всё-если

|Всё-цикл

|Если (res=0 и myLen (str1)>myLen (str2)) то

| | res=1

|Всё-если

|Если (res=0 и myLen (str1) <myLen (str2)) то

| | res=-1

|Всё-если

|Вернуть res

Конец

Алгоритм at (Аргументы: index, size: Целое, buf: Символьный массив; Результат buf[index]: Символ)

Начало

|Если (index>=0 и index<size) то

| |Вернуть buf [index]

|Иначе

| |Вывод: Обращение к неверному элементу

| |Завершить

|Всё-если

Конец

Функция setBuf (Аргументы: str, buf: Символьный массив, size: Целое)

Начало

|size=myLen (str)

|Целое: i

|Для <i=0;i<size;i++>

| |buf [i]=str[i]

|Всё-цикл

|buf[size]=’\0’

Конец

Функция == (Аргументы: buf, str: Строка, Результат: res: Логический тип)

Начало

|Если (myCmp (buf, str )=0) то

| |res=true

|Иначе

| |res=false

|Всё-если

|Вернуть res

Конец

Функция + (Аргументы: buf, str: Строка, size1, size2: Целое, Результат: tmp: Строка)

Начало

| Целое: res3, i, j

|res3=res1+res2

|Длина (tmp)=res3

|Для <i=0; i<size1; i++>

| |tmp[i]=buf[i]

|Всё-цикл

|j=size1

|Для <i=0; i<size2; i++; j++>

| |tmp[j]=str[i]

|Всё-цикл

|Вернуть tmp

Конец