**МИНОБРНАУКИ РОССИИ**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ  
ВЫСШЕГО ОБРАЗОВАНИЯ  
**«БЕЛГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ  
ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ им. В.Г.ШУХОВА»  
(БГТУ им. В.Г.Шухова)**

Кафедра программного обеспечения вычислительной техники и автоматизированных систем

Лабораторная работа №7

Дисциплина: Обьектно-ориентированное программирование

по теме «Исключительные ситуации в С++»

Выполнил: ст. группы ВТ-22  
Ковалев Павел ВТ-22

Проверил: Буханов Д.Г.

**Белгород 2019**

**Цель работы:** Получение теоретических знаний об исключительных ситуациях в С++. Получение практических навыков при работе с исключениями в С++.

Задания к лабораторной работе

1. Изучить теоретические сведения об исключениях в С++.
2. Изучить самостоятельно стандартные классы для исключений предусмотренных в С++.
3. Разработать программу в соответствии с заданным вариантом задания.
4. Оформить отчет.

#include <QCoreApplication>

#include <QTextCodec>

//cout << QString::fromUtf8("").toLocal8Bit().data();

#include <iostream>

#include <iostream>

using namespace std;

#define MAX\_STRING\_SIZE 256

class **\_String**{

protected:

unsigned size;

char\* str;

public:

**\_String**(unsigned size):size(size){

this->str=(char\*)malloc(sizeof(char)\*size);

}

~**\_String**(){

free(str);

}

};

class **Exception**:public exception{

public:

virtual void ***show***()=0;

};

class **OutOfMemory**:public Exception{

unsigned size;

public:

**OutOfMemory**(unsigned size):size(size){

this->*show*();

};

void ***show***(){

cout<<"Exeption: out of memory. Current str size - "<<this->size<<endl;

};

};

class **EmptyStr**:public Exception{

public:

**EmptyStr**(){

this->*show*();

};

void ***show***(){

cout<<"Exeption: empty str.\n";

};

};

class **OutOfRange**:public Exception{

unsigned called\_char\_index;

public:

**OutOfRange**(unsigned called\_str):called\_char\_index(called\_str){

this->*show*();

};

void ***show***(){

cout<<"Exeption: out of range. Called char - "<<this->called\_char\_index<<endl;

};

};

class **String**:public \_String{

unsigned current\_size;

public:

**String**(unsigned size);

**String**(char\* init\_str);

~**String**();

void **print**();

char operator[](int i);

String operator =(char\* origin);

void **del\_char**(int index);

void **add\_char**(int index,char c);

};

String::**String**(unsigned size):\_String(size){

this->current\_size=0;

this->str=(char\*)malloc(sizeof(char)\*size);

}

String::**String**(char\* init\_str):\_String(MAX\_STRING\_SIZE){

int i = 0;

while(init\_str[i]!='\0'){

i++;

}

this->str=(char\*)malloc(sizeof(char)\*i+2);

this->current\_size=i+1;

this->str[i+1]='\0';

while(i!=0){

this->str[i]=init\_str[i];

i--;

}

}

String::~**String**(){

free(str);

}

void String::**print**(){

try{

if(this->current\_size==0) throw EmptyStr();

cout<<this->str<<endl;

}

catch(EmptyStr){}

}

char String::operator[](int i){

try{

if(i>this->current\_size) throw OutOfRange(i);

if(this->current\_size==0) throw EmptyStr();

return this->str[i];

}

catch(OutOfRange){}

catch(EmptyStr){}

}

String String::operator =(char\* origin){

try{

int i=0,str\_size;

while(origin[i]!='\0') i++;

str\_size=i;

i=0;

if(str\_size>MAX\_STRING\_SIZE) throw OutOfMemory(str\_size);

this->str=(char\*)realloc(this->str,str\_size+1);

while(i<str\_size){

this->str[i]=origin[i];

i++;

}

this->current\_size=str\_size+1;

return \*this;

}

catch(OutOfMemory){}

}

void String::**del\_char**(int index){

try{

if(index>current\_size) throw OutOfRange(index);

if(current\_size==0) throw EmptyStr();

for(int i=index;i<current\_size;i++){

this->str[i]=str[i+1];

}

current\_size=current\_size-1;

}

catch(EmptyStr){}

catch(OutOfRange){}

};

void String::**add\_char**(int index,char c){

try{

if(index>current\_size) throw OutOfRange(index);

if(current\_size+1>MAX\_STRING\_SIZE) throw OutOfMemory(current\_size);

int i=index+1;

while(i<this->current\_size+1){

this->str[i]=this->str[i-1];

}

this->str[index]=c;

}

catch(OutOfRange){}

catch(OutOfMemory){}

};

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

{

QCoreApplication a(*argc*, argv);

#ifdef Q\_OS\_WIN32

QTextCodec::setCodecForLocale(QTextCodec::codecForName("IBM 866"));

#endif

#ifdef Q\_OS\_LINUX

QTextCodec::setCodecForLocale(QTextCodec::codecForName("UTF-8"));

#endif

char\* temp\_str;

printf("Введите строку");

scanf("%s",temp\_str);

String str(temp\_str);

str.print();

return a.exec();

}