Министерство образования и науки РФ

Федеральное государственное автономное

образовательное учреждение высшего образования

«Санкт-Петербургский национальный исследовательский университет

ИТМО»

**факультет безопасности информационных технологий**

**КУРСОВАЯ РАБОТА**

Разработка и иерархия классов

**Руководитель темы** Безруков В.А.

**Исполнитель темы** Шарифуллин И.А.



Санкт-Петербург, 2021

Оглавление

[Введение 3](#_Toc484291547)

[Цель: 3](#_Toc484291548)

[Задачи: 3](#_Toc484291549)

[Техническое задание: 4](#_Toc484291550)

[Теория: 7](#_Toc484291551)

[1. Класс Строка 7](#_Toc484291552)

[1.1 Базовый класс Строка 7](#_Toc484291553)

[1.2 Строка идентификатор 8](#_Toc484291554)

[1.3 Десятичная строка 9](#_Toc484291555)

[Заключение 10](#_Toc484291556)

[Список литературы 11](#_Toc484291557)

[Приложение: 12](#_Toc484291558)

[Файл String.h 12](#_Toc484291559)

[Файл IdentStr.h 13](#_Toc484291560)

[Файл Decstr.h 14](#_Toc484291560)

[Файл String.cpp 15](#_Toc484291561)

[Файл IdentStr.cpp 16](#_Toc484291561)

[Файл Decstr.cpp 19](#_Toc484291562)

[Файл main.cpp 22](#_Toc484291564)

# Введение

Цель: Основной целью данной работы является создание базового класса «строка», 2 производных класса от него, а также проработать систему тестирования написанных классов.

## Задачи:

Для достижения поставленной цели была выполнена следующая задача:

1. Разработка классов: базовая строка, строка идентификатор, десятичная строка согласно ТЗ.
2. Разработка методов согласно ТЗ для написанных классов.
3. Разработка консольного меню для удобного тестирования написанных классов, включая их методы.

# Техническое задание:

Описать базовый класс Строка.

1. Обязательные члены класса:
2. указатель на char – хранит адрес динамически выделенной памяти для размещения символов строки;
3. значение типа int – хранит длину строки в байтах.

Обязательные методы должны выполнять следующие действия:

1. конструктор без параметров;
2. конструктор, принимающий в качестве параметра Си-строку (заканчивается нулевым байтом);
3. конструктор, принимающий в качестве параметра символ (char);
4. конструктор копирования;
5. деструктор.
6. чтение символа из заданной позиции;

Производный от Строка класс Строка-идентификатор.

Строки данного класса строятся по правилам записи идентификаторов в СИ, и могут включать в себя только те символы, которые могут входить в состав Си-идентификаторов. Если исходные данные противоречат правилам записи идентификатора, то создается пустая СТРОКА ИДЕНТИФИКАТОР.

1. Обязательные методы:
2. Конструктор без параметров;
3. Конструктор, принимающий в качестве параметра Си-строку (заканчивается нулевым байтом);
4. Конструктор копирования;
5. Деструктор;
6. Поиск последнего вхождения символа в строку;
7. Переопределить следующие операции:

* присваивание (=);
* сложение (+) – операция конкатенации строк;
* индексное выражение [];

Производный от класса «Строка» класс «Десятичная строка».

Строки данного класса могут содержать только символы десятичных цифр и символы – и +, задающие знак числа. Символы – и + могут находиться только в первой позиции числа, причем символ + может отсутствовать, в том случае число считается положительным. Если в составе инициализирующей строки будут встречены любые символы, отличные от допустимых, десятичная строка принимает нулевое значение. Содержимое данных строк рассматривается как десятичное число.

Обязательные методы:

* конструктор без параметров;
* конструктор, принимающий в качестве параметра Си-строку (заканчивается нулевым байтом);
* конструктор копирования;
* деструктор;

Переопределить следующие операции :

\* присваивание (=);

\* операция сложения (+);

\* операция (<) – сравнение “меньше, чем”.

Разработчик вправе вводить любое (с обоснованием необходимости) число дополнительных членов и методов.

1. Разработать иерархию классов по следующей схеме:

СТРОКА

СТРОКА ИДЕНТИФИКАТОР БИТОВАЯ СТРОКА

или

ДЕСЯТИЧНАЯ СТРОКА

или

КОМПЛЕКСНОЕ ЧИСЛО

с обязательной поддержкой заданных членов и методов.

1. Написать тестовую программу, которая:

* динамически выделяет массив указателей на базовый класс (4-6 шт.);
* в режиме диалога заполняет этот массив указателями на производные классы,
* при этом экземпляры производных классов создаются динамически с заданием
* начальных значений;
* для созданных экземпляров производных классов выполняется
* проверка всех разработанных методов (в соответствии с вариантом задания),
* с выводом исходных данных и результатов на дисплей.
* для конструкторов копирования каждого класса предусмотреть
* диагностическую печать количества его ввызовов в заданное место дисплея,
* (рекомендуется использовать статические члены класса).

Режим диалога обеспечивается с помощью иерархического меню.

1. Основные пункты:
2. "Инициализация".

Подпункты:

* 1. "Число элементов" - задает число элементов в массиве указателей на базовый класс. Запрещается после задания числа элементов пользоваться этим пунктом меню.
  2. "Начальное значение" - с помощью этого пункта меню можно задать номер элемента, его тип и начальное значение. Задавать начальные значения и работать с другими пунктами меню запрещается до тех пор, пока не будет задано число элементов. Допускается задать новое начальное значение несколько раз.

1. "Тестирование"

В качестве подпунктов указываются типы строк из варианта задания и обязательные для всех подпункты "строка" и "задание операнда", например:

1. "Строка"
2. "СТРОКА ИДЕНТИФИКАТОР"
3. "КОМПЛЕКСНОЕ ЧИСЛО"
4. "Задать операнд"

После выбора одного из этих пунктов меню предлагается выбрать один методов из списка всех обязательных методов (кроме конструкторов и деструкторов), связанных с выбранным подпунктом. Результат выводится на экран.

1. "Выход".

Допускается одновременный вывод всех пунктов меню на экран. Переход между пунктами меню осуществляется клавишами со стрелками. Выбор пункта меню осуществляется нажатием клавиши Enter.

Значения экземпляров классов по мере их инициализации выводятся на дисплей и остаются на все время выполнения программы. После выбора операндов они подсвечиваются цветом, отличным от остальных экземпляров.

# Теория:

Далее рассмотрим основные функции, операции, данные-члены классов, а также методы используемые в данной работе. Для этого разделим основной класс «Строка» на подпункты.

## 1. Класс Строка

## 1.1 Базовый класс Строка

Название класса: base\_string. Является базовым классов для классов комплексное число и строка идентификатор.

Данные-члены класса, объявленные как protected:

1) char \* ar – указатель на массив символов;

2) int len – переменная, хранящая длину строки.

Методы класса base\_string:

1) base\_string () – конструктор без параметров, создает строку нулевой длины;

2) base\_string (const char\*) – конструктор, в качестве параметра принимающий Си-строку.;

3) base\_string (char) – конструктор, принимающий в качестве параметра один символ.;

4) base\_string (const base\_string&) – конструктор копирования;

5) ~ base\_string ( ) – деструктор;

6) int Length ( ) const – функция получения длины строки;

7) char\* value ( ) – функция получения значения строки;

## 1.2 Строка идентификатор

Название класса: IdentStr. Данный класс является производным классом класса base\_string.

Методы класса IdentStr:

1) IdentStr ( ) – конструктор без пораметров;

2) IdentStr (char) – конструктор, принимающий в качестве параметра символ

3) IdentStr (const char\*) – конструктор, принимающий в качестве параметра си-строку.

4) IdentStr (const IdentStr&) – конструктор копирования;

5) ~ IdentStr ( ) – деструктор;

6) char& operator [](int i) - перегрузка оператора индексации;

7) IdentStr& operator =(const IdentStr&); - перезагрузка оператора присваивания;

8) friend const IdentStr operator+(const IdentStr&, const IdentStr&) - функция конкатенации строк.

9) int Search(char c) const – функция поиска последнего вхождения символа в строку

## 1.3 Десятичная строка

Наименование класса: DecStr. Данный класс является производным классом класса base\_string.

Методы данного класса:

1) DecStr ( ) – конструктор;

2) DecStr (const char\*) – конструктор, принимающий в качестве параметра Си-строку;

3) DecStr (const DecStr&) – конструктор копирования;

4) ~ DecStr ( ) – деструктор;

5) DecStr operator=(const DecStr &);

- перегрузка операторов присваивания;

6) friend DecStr operator+(const DecStr&, const DecStr&)

– дружественная функция, перегрузка оператора сложения;

6) friend const DecStr operator-(const DecStr&, const DecStr&) – дружественная функция, перегрузка оператора вычитания;

7) friend bool operator <(const DecStr&) – дружественная функция, оператор сравнения “меньше, чем”.

# Заключение

В курсовой работе передо мной была поставлена цель, которая в процессе была достигнута. Мною были созданы такие тестируемые классы, как базовый класс «строка», «строка-идентификатор», «десятичная строка». Так же были реализованы и протестированы различные методы созданных классов. При проверке было выявлено, что тестируемые классы работают корректно. Вывод работы программы можно увидеть в консольном приложении.

# Список литературы

* + 1. Б.И. Березин. Начальный курс С и С++. – М.:Издательство Диалог-МИФИ, 2005 г. – 248 с.

2. Р. Джонс, Я. Стюарт. Программируем на Си. – М.: Издательское объединение «ЮНИТИ». «Компьютер», 1994 г. – 236 с.

# Приложение:

## Файл String.h

#pragma once

#ifndef \_\_BASE\_STRING\_\_

#define \_\_BASE\_STRING\_\_

#include <iostream>

using namespace std;

class base\_string

{

protected:

int len;

char\* ar;

public:

explicit base\_string(char newAr[], int length = -1);

base\_string();

base\_string(const base\_string&);

base\_string(char c);

base\_string(const char\*);

~base\_string();

int Length() const;

char\* value() const;

};

#endif

## Файл IdentStr.h

#pragma once

#ifndef \_\_IDENTSTR\_\_

#define \_\_IDENTSTR\_\_

#include "String.h"

class IdentStr : public base\_string

{

public:

IdentStr();

IdentStr(int = 0);

IdentStr(char);

IdentStr(const char\*);

IdentStr(const IdentStr&);

~IdentStr();

IdentStr& operator=(const IdentStr&);

IdentStr operator ~();

char& operator [](int i);

int Search(char c) const;

friend IdentStr operator + (const IdentStr&, const IdentStr&);

};

#endif

## Файл Decstr.h

#pragma once

#ifndef \_\_DECSTR\_\_

#define \_\_DECSTR\_\_

#include "String.h"

class DecStr : public base\_string

{

public:

DecStr(int);

DecStr();

DecStr(const char\*);

~DecStr();

DecStr& operator = (const DecStr&);

DecStr(const DecStr&);

friend DecStr operator + (const DecStr&, const DecStr&);

friend DecStr operator - (const DecStr&, const DecStr&);

friend bool operator < (const DecStr&, const DecStr&);

};

#endif

## Файл String.cpp

#include "String.h"

#include <iostream>

#include <string>

using namespace std;

base\_string::base\_string(char newAr[], int length){

this->len = length > -1 ? length : sizeof(newAr) / sizeof(char);

ar = new char[len + 1];

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

ar[i] = newAr[i];

}

ar[len] = '\0';

}

base\_string::base\_string(){

len = 1;

ar = new char[1]{ '\0' };

}

int base\_string::Length() const{

auto a = len;

return a;

}

base\_string::base\_string(const char\* S) : len(strlen(S)), ar(new char[len + 1]) {

strcpy\_s(ar, len + 1, S);

}

base\_string::base\_string(char ch) : len(1), ar(new char[len + 1]) {

ar[0] = ch;

ar[1] = '\0';

}

base\_string::base\_string(const base\_string& from) : len(strlen(from.ar)), ar(new char[from.len + 1]) {

strcpy\_s(ar, from.len + 1, from.ar);

}

base\_string::~base\_string() {

if (ar) delete[]ar;

}

char\* base\_string::value() const {

return ar;

}

## Файл IdentStr.cpp

#include "String.h"

#include "IdenStr.h"

#include <iostream>

#include <string.h>

using namespace std;

IdentStr::IdentStr()

{

len = 1;

ar = new char[1]{};

}

IdentStr::IdentStr(int val) : base\_string(val){}

IdentStr::IdentStr(char ch) : base\_string(ch)

{

// проверка символ(строка) начинается с цифры или нет

if (!((ar[0] >= 'a' && ar[0] <= 'z') || (ar[0] >= 'A' && ar[0] <= 'Z')))

{

if (ar) delete[] ar;

len = 0;

ar = new char[len + 1];

ar[0] = '\0';

return;

}

}

IdentStr::IdentStr(const char\* Str) : base\_string(Str)

{

// определяем массив указателей на ключевые слова

const char\* keyword[] = { "while", "for", "switch", "break", "return", "and", "and\_eq", "asm", "auto", "break", "catch", "case", "char", "class", "const", "continue", "default", "delete", "do", "double", "else", "enum", "extern", "float", "for", "friend", "goto", "if", "inline", "int", "long", "new", "operator", "private", "protected", "public", "register", "return", "short", "signed", "sizeof", "static", "struct", "template", "this", "throw", "try", "typedef", "union", "unsigned", "virtual", "void", "volatile" };

for (int i = 0; i < 6; i++)

{

if (strcmp(ar, keyword[i]) == 0)

{

if (ar) delete[] ar;

len = 0;

ar = new char[len + 1];

ar[0] = '\0';

return;

}

}

// проверка первого символа

if (!((ar[0] >= 'a' && ar[0] <= 'z') || (ar[0] >= 'A' && ar[0] <= 'Z') || (ar[0] == '\_')))

{

if (ar) delete[] ar;

len = 0;

ar = new char[len + 1];

ar[0] = '\0';

return;

}

for (int i = 1; i < len; i++)

{

if (!((ar[i] >= 'a' && ar[i] <= 'z') || (ar[i] >= 'A' && ar[i] <= 'Z') || (ar[i] == '\_') || (ar[i] >= '0' && ar[i] <= '9')))

{

if (ar) delete[] ar;

len = 0;

ar = new char[len + 1];

ar[0] = '\0';

return;

}

}

}

IdentStr::IdentStr(const IdentStr& from) : base\_string(from){}

IdentStr:: ~IdentStr(){}

IdentStr& IdentStr:: operator=(const IdentStr& S)

{

if (&S != this)

{

delete[] ar;

len = strlen(S.ar);

ar = new char[len + 1];

strcpy\_s(ar, len + 1, S.ar);

}

return \*this;

}

IdentStr IdentStr :: operator ~()

{

int i, j;

char tmp;

for (i = 0, j = len - 1; i < len / 2; i++, j--){

tmp = ar[i];

ar[i] = ar[j];

ar[j] = tmp;

}

return\*this;

}

IdentStr operator+(const IdentStr& a, const IdentStr& b){

char\* temp = new char[a.len + b.len + 1];

int i = 0;

while (true)

{

if (i < a.len)

{

temp[i] = a.ar[i];

}

if (i < b.len)

{

temp[i + a.len] = b.ar[i];

}

if (i >= a.len && i >= b.len) break;

i++;

}

temp[a.len + b.len] = '\0';

return IdentStr(temp);

}

int IdentStr::Search(char c) const{

for (int i = len - 1; i >= 0; i--)

if (c == ar[i])

return i;

return -1;

}

char& IdentStr::operator[](int i){

return ar[i];

}

## Файл Decstr.cpp

#include "String.h"

#include "Decstr.h"

#include <iostream>

#include <string.h>

using namespace std;

DecStr::DecStr(int val) : base\_string(val){}

DecStr::DecStr() : base\_string() {}

DecStr::~DecStr(){}

DecStr::DecStr(const char\* Str) : base\_string(Str){

if (!((ar[0] >= '1' && ar[0] <= '9') || (ar[0] == '+' || ar[0] == '-'))){

cout << "Bad Symbol, ar[0]=" << ar[0] << endl;

if (ar) delete[] ar;

len = 0;

ar = new char[len + 1];

ar[0] = '\0';

return;

}

for (int i = 1; i < len; i++){

if (!(ar[i] >= '0' && ar[i] <= '9')){

cout << "Bad Symbol, ar[" << i << "]=" << ar[i] << endl;

if (ar) delete[] ar;

len = 0;

ar = new char[len + 1];

ar[0] = '\0';

return;

}

}

}

DecStr::DecStr(const DecStr& from) : base\_string(from){}

DecStr& DecStr::operator=(const DecStr& DS){

if (&DS != this){

delete[] ar;

len = strlen(DS.value());

ar = new char[len + 1];

strcpy\_s(ar, len + 1, DS.value());

}

return \*this;

}

DecStr operator + (const DecStr& pobj1, const DecStr& pobj2){

char str[12];

char\* ptr = str;

int a = 0, b = 0, ans = 0;

a = atoi(pobj1.value());

b = atoi(pobj2.value());

\_asm {

mov eax, a;

mov ebx, b;

add eax, ebx;

mov ans, eax;

}

\_itoa(ans, ptr, 10);

return DecStr(str);

}

DecStr operator - (const DecStr& pobj1, const DecStr& pobj2){

char str[12];

char\* ptr = str;

int a = 0, b = 0, ans = 0;

a = atoi(pobj1.value());

b = atoi(pobj2.value());

\_asm {

mov eax, a;

mov ebx, b;

sub eax, ebx;

mov ans, eax;

}

\_itoa(ans, ptr, 10);

return DecStr(str);

}

bool operator < (const DecStr& pobj1, const DecStr& pobj2){

if (atoi(pobj1.value()) < atoi(pobj2.value()))

return true;

else

return false; }

## Файл main.cpp

#include "String.h"

#include "IdenStr.h"

#include "Decstr.h"

#include <iostream>

#include <locale.h>

using namespace std;

void numElem();

void intialElem();

void getStrLen();

void idenSearch();

void movStrId();

void sumStrId();

void indStrId();

void invStrDec();

void movStrDec();

void sumStrDec();

void subStrDec();

void lessStrDec();

bool print();

struct CHECKPOINT {

int mainMenu;

int subMenu;

};

int N = 0;

base\_string\*\* testObj;

int\* testObjtype;

int main() {

setlocale(LC\_ALL, "rus");

CHECKPOINT pos = { 0,0 };

while (true) {

system("cls");

switch (pos.mainMenu) {

case 0: //вывод меню

cout << "(1)Иннициализация" << endl;

cout << "(2)Tестирование" << endl;

cout << "(3)Печать" << endl;

cout << "(4)Выход" << endl;

break;

case 1: // иннициализация

cout << "(1)Число элементов" << endl;

cout << "(2)Иннициализация элементов" << endl;

break;

case 2: // тестирование

switch (pos.subMenu) {

case 0: // меню тестирование

cout << "1. Строка" << endl;

cout << "2. Строка Идентификатор" << endl;

cout << "3. Десятичная строка" << endl;

break;

case 1: // тестирование класса строка

cout << "1. Получение длины строки" << endl;

break;

case 2: // тестирование класса строка идентификатор

cout << "1. Поиск символа в строке" << endl;

cout << "2. Оператор присваивания" << endl;

cout << "3. Сумма {A = B + C}" << endl;

cout << "4. Индексное выражение IdenStr[?]" << endl;

break;

case 3: // тестирование класса Десятичная строка

cout << "1. Оператор присваивания" << endl;

cout << "2. Сумма {A = B + C}" << endl;

cout << "3. Сравнение(меньше)" << endl;

cout << "4. Инвертирование" << endl;

cout << "5. Разница {A = B + C}" << endl;

break;

}

break;

case 3: // печать

pos = { 0,0 };

print();

continue;

break;

case 4: // выход

return 0;

break;

}

char choise;

cin >> choise;

switch (pos.mainMenu) {

case 0: //вывод меню

switch (choise) {

case '1':

pos.mainMenu = 1;

break;

case '2':

pos.mainMenu = 2;

break;

case '3':

pos.mainMenu = 3;

break;

case '4':

pos.mainMenu = 4;

break;

}

break;

case 1: //иннициализация

switch (choise) {

case '1':

numElem();

break;

case '2':

intialElem();

break;

default:

pos = { 0,0 };

}

break;

case 2: //тестирование

switch (pos.subMenu) {

case 0:

switch (choise) {

case '1':

pos.subMenu = 1;

break;

case '2':

pos.subMenu = 2;

break;

case '3':

pos.subMenu = 3;

break;

default:

pos = { 0,0 };

}

break;

case 1: // меню тестирование класса строка

switch (choise) {

case '1':

getStrLen();

break;

default:

pos = { 0,0 };

}

break;

case 2: // тестирование класса Строка\_идентификатор

switch (choise) {

case '1':

idenSearch();

break;

case '2':

movStrId();

break;

case '3':

sumStrId();

break;

case '4':

indStrId();

break;

default:

pos = { 0,0 };

}

break;

case 3: // тестирование класса Десятичная строка

switch (choise) {

case '1':

movStrDec();

break;

case '2':

sumStrDec();

break;

case '3':

lessStrDec();

break;

case '4':

invStrDec();

break;

case '5':

subStrDec();

break;

default:

pos = { 0,0 };

}

break;

}

break;

}

}

return 0;

}

void numElem() {

system("cls");

if (N != 0) {

cout << "Число элементов уже определено!" << endl;

system("pause");

return;

}

cout << "Введите число элеметнов(1 <= N <= 6): ";

cin >> N;

if (N < 1 || N > 6) {

N = 0;

cout << "Некорректное N!" << endl;

system("pause");

return;

}

testObj = new base\_string \* [N] {0};

testObjtype = new int[N] {0};

}

void intialElem() {

if (!print())

return;

int n;

cout << "Введите номер объекта: ";

cin >> n;

if (n < 1 || n > N) {

cout << "Некорректный номер объекта!" << endl;

system("pause");

return;

}

n--;

char choiseClass;

cout << "1. Cтрока-идентификатор" << endl;

cout << "2. Десятичная строка" << endl;

cin >> choiseClass;

switch (choiseClass) {

case '1':

cout << "Введите значение Строки-идентификатора: " << endl;

char buf[256];

cin.ignore();

cin.getline(buf, 256);

testObj[n] = new IdentStr(buf);

testObjtype[n] = 1;

break;

case '2':

cout << "Введите значение Десятичной строки: " << endl;

cin.ignore();

cin.getline(buf, 256);

testObj[n] = new DecStr(buf);

testObjtype[n] = 2;

break;

default:

cout << "Ошибка ввода!" << endl;

}

system("pause");

}

void getStrLen() {

if (!print())

return;

int n;

cout << "Введите номер объекта: ";

cin >> n;

if (n < 1 || n > N) {

cout << "Некорректный номер объекта!" << endl;

system("pause");

return;

}

n--;

if (testObj[n] == 0) {

cout << "Объект не иннициализирован!" << endl;

system("pause");

return;

}

cout << "Длина строки объекта №" << n + 1 << " = " << testObj[n] -> Length();

system("pause");

}

void idenSearch() {

if (!print())

return;

int n;

cout << "Введите номер объекта: ";

cin >> n;

if (n < 1 || n > N) {

cout << "Некорректный номер объекта!" << endl;

system("pause");

return;

}

n--;

if (testObj[n] == 0) {

cout << "Объект не иннициализирован!" << endl;

system("pause");

return;

}

if (testObjtype[n] != 1) {

cout << "Объект не строка-идентификатор!" << endl;

system("pause");

return;

}

cout << "Введите искомый символ: ";

char c;

cin >> c;

cout << ((IdentStr\*)testObj[n])->Search(c) << endl;

system("pause");

}

void movStrId() {

if (!print())

return;

int n1, n2;

cout << "Введите номер операнда 1: ";

cin >> n1;

if (n1 < 1 || n1 > N) {

cout << "Некорректный номер объекта!" << endl;

system("pause");

return;

}

n1--;

if (testObj[n1] == 0) {

cout << "Объект не иннициализирован!" << endl;

system("pause");

return;

}

if (testObjtype[n1] != 1) {

cout << "Объект не строка-идентификатор!" << endl;

system("pause");

return;

}

cout << "Введите номер операнда 2: ";

cin >> n2;

if (n2 < 1 || n2 > N) {

cout << "Некорректный номер объекта!" << endl;

system("pause");

return;

}

n2--;

if (testObj[n2] == 0) {

cout << "Объект не иннициализирован!" << endl;

system("pause");

return;

}

if (testObjtype[n2] != 1) {

cout << "Объект не строка-идентификатор!" << endl;

system("pause");

return;

}

testObj[n1] = testObj[n2];

cout << "Присвоение выполнено успешно!" << endl;

system("pause");

}

void sumStrId() {

if (!print())

return;

int n1, n2, n3;

cout << "Введите номер операнда1: ";

cin >> n1;

if (n1 < 1 || n1 > N) {

cout << "Некорректный номер объекта!" << endl;

system("pause");

return;

}

n1--;

if (testObj[n1] == 0) {

cout << "Объект не иннициализирован!" << endl;

system("pause");

return;

}

if (testObjtype[n1] != 1) {

cout << "Объект не строка-идентификатор!" << endl;

system("pause");

return;

}

cout << "Введите номер операнда2: ";

cin >> n2;

if (n2 < 1 || n2 > N) {

cout << "Некорректный номер объекта!" << endl;

system("pause");

return;

}

n2--;

if (testObj[n2] == 0) {

cout << "Объект не иннициализирован!" << endl;

system("pause");

return;

}

if (testObjtype[n2] != 1) {

cout << "Объект не строка-идентификатор!" << endl;

system("pause");

return;

}

cout << "Введите номер операнда3: ";

cin >> n3;

if (n3 < 1 || n3 > N) {

cout << "Некорректный номер объекта!" << endl;

system("pause");

return;

}

n3--;

if (testObj[n3] == 0) {

cout << "Объект не иннициализирован!" << endl;

system("pause");

return;

}

if (testObjtype[n3] != 1) {

cout << "Объект не строка-идентификатор!" << endl;

system("pause");

return;

}

\*(IdentStr\*)testObj[n1] = (\*(IdentStr\*)testObj[n2] + \*(IdentStr\*)testObj[n3]);

cout << "Сложение выполнено успешно!" << endl;

system("pause");

}

void indStrId() {

if (!print())

return;

int n1, n2;

cout << "Введите номер операнда1: ";

cin >> n1;

if (n1 < 1 || n1 > N) {

cout << "Некорректный номер объекта!" << endl;

system("pause");

return;

}

n1--;

if (testObj[n1] == 0) {

cout << "Объект не иннициализирован!" << endl;

system("pause");

return;

}

if (testObjtype[n1] != 1) {

cout << "Объект не строка-идентификатор!" << endl;

system("pause");

return;

}

cout << "Введите номер операнда2: ";

cin >> n2;

if (n2 > (testObj[n1]->Length() - 1) || n2 < 0) {

cout << "Некорректный индекс!" << endl;

system("pause");

return;

}

IdentStr wow(testObj[n1]->value());

cout << wow[n2] << endl;

system("pause");

}

void invStrDec() {

if (!print())

return;

int n1;

DecStr tmp1("-1");

DecStr tmp2("-1");

cout << "Введите номер операнда 1: ";

cin >> n1;

if (n1 < 1 || n1 > N) {

cout << "Некорректный номер объекта!" << endl;

system("pause");

return;

}

n1--;

if (testObj[n1] == 0) {

cout << "Объект не иннициализирован!" << endl;

system("pause");

return;

}

if (testObjtype[n1] != 2) {

cout << "Объект не Десятичная строка!" << endl;

system("pause");

return;

}

tmp2 = \*((DecStr\*)testObj[n1]);

\*(DecStr\*)testObj[n1] = (tmp1 - tmp2 + 1);

cout << "Инвертирование выполнено успешно!" << endl;

system("pause");

}

void movStrDec() {

if (!print())

return;

int n1, n2;

cout << "Введите номер операнда 1: ";

cin >> n1;

if (n1 < 1 || n1 > N) {

cout << "Некорректный номер объекта!" << endl;

system("pause");

return;

}

n1--;

if (testObj[n1] == 0) {

cout << "Объект не иннициализирован!" << endl;

system("pause");

return;

}

if (testObjtype[n1] != 2) {

cout << "Объект не Десятичная строка!" << endl;

system("pause");

return;

}

cout << "Введите номер операнда 2: ";

cin >> n2;

if (n2 < 1 || n2 > N) {

cout << "Некорректный номер объекта!" << endl;

system("pause");

return;

}

n2--;

if (testObj[n2] == 0) {

cout << "Объект не иннициализирован!" << endl;

system("pause");

return;

}

if (testObjtype[n2] != 2) {

cout << "Объект не Десятичная строка!" << endl;

system("pause");

return;

}

testObj[n1] = testObj[n2];

cout << "Присвоение выполнено успешно!" << endl;

system("pause");

}

void sumStrDec() {

if (!print())

return;

int n1, n2, n3;

cout << "Введите номер операнда 1: ";

cin >> n1;

if (n1 < 1 || n1 > N) {

cout << "Некорректный номер объекта!" << endl;

system("pause");

return;

}

n1--;

if (testObj[n1] == 0) {

cout << "Объект не иннициализирован!" << endl;

system("pause");

return;

}

if (testObjtype[n1] != 2) {

cout << "Объект не Десятичная строка!" << endl;

system("pause");

return;

}

cout << "Введите номер операнда 2: ";

cin >> n2;

if (n2 < 1 || n2 > N) {

cout << "Некорректный номер объекта!" << endl;

system("pause");

return;

}

n2--;

if (testObj[n2] == 0) {

cout << "Объект не иннициализирован!" << endl;

system("pause");

return;

}

if (testObjtype[n2] != 2) {

cout << "Объект не Десятичная строка!" << endl;

system("pause");

return;

}

cout << "Введите номер операнда 3: ";

cin >> n3;

if (n3 < 1 || n3 > N) {

cout << "Некорректный номер объекта!" << endl;

system("pause");

return;

}

n3--;

if (testObj[n3] == 0) {

cout << "Объект не иннициализирован!" << endl;

system("pause");

return;

}

if (testObjtype[n3] != 2) {

cout << "Объект не Десятичная строка!" << endl;

system("pause");

return;

}

\*(DecStr\*)testObj[n1] = (\*(DecStr\*)testObj[n2] + \*(DecStr\*)testObj[n3]);

cout << "Сложение выполнено успешно!" << endl;

system("pause");

}

void subStrDec() {

if (!print())

return;

int n1, n2, n3;

cout << "Введите номер операнда 1: ";

cin >> n1;

if (n1 < 1 || n1 > N) {

cout << "Некорректный номер объекта!" << endl;

system("pause");

return;

}

n1--;

if (testObj[n1] == 0) {

cout << "Объект не иннициализирован!" << endl;

system("pause");

return;

}

if (testObjtype[n1] != 2) {

cout << "Объект не Десятичная строка!" << endl;

system("pause");

return;

}

cout << "Введите номер операнда 2: ";

cin >> n2;

if (n2 < 1 || n2 > N) {

cout << "Некорректный номер объекта!" << endl;

system("pause");

return;

}

n2--;

if (testObj[n2] == 0) {

cout << "Объект не иннициализирован!" << endl;

system("pause");

return;

}

if (testObjtype[n2] != 2) {

cout << "Объект не Десятичная строка!" << endl;

system("pause");

return;

}

cout << "Введите номер операнда 3: ";

cin >> n3;

if (n3 < 1 || n3 > N) {

cout << "Некорректный номер объекта!" << endl;

system("pause");

return;

}

n3--;

if (testObj[n3] == 0) {

cout << "Объект не иннициализирован!" << endl;

system("pause");

return;

}

if (testObjtype[n3] != 2) {

cout << "Объект не Десятичная строка!" << endl;

system("pause");

return;

}

\*(DecStr\*)testObj[n1] = (\*(DecStr\*)testObj[n2] - \*(DecStr\*)testObj[n3]);

cout << "Вычитание выполнено успешно!" << endl;

system("pause");

}

void lessStrDec() {

int n1, n2;

if (!print())

return;

cout << "Введите номер операнда 1: ";

cin >> n1;

if (n1 < 1 || n1 > N) {

cout << "Некорректный номер объекта!" << endl;

system("pause");

return;

}

n1--;

if (testObj[n1] == 0) {

cout << "Объект не иннициализирован!" << endl;

system("pause");

return;

}

if (testObjtype[n1] != 2) {

cout << "Объект не Десятичная строка!" << endl;

system("pause");

return;

}

cout << "Введите номер операнда 2: ";

cin >> n2;

if (n2 < 1 || n2 > N) {

cout << "Некорректный номер объекта!" << endl;

system("pause");

return;

}

n2--;

if (testObj[n2] == 0) {

cout << "Объект не иннициализирован!" << endl;

system("pause");

return;

}

if (testObjtype[n2] != 2) {

cout << "Объект не Десятичная строка!" << endl;

system("pause");

return;

}

cout << "Результат сравнения: " << ((\*(DecStr\*)testObj[n1] < (\*(DecStr\*)testObj[n2]))) << endl;

system("pause");

}

bool print() {

if (N == 0) {

cout << "Число элементов не определено!" << endl;

system("pause");

return false;

}

system("cls");

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

cout << i + 1 << ". ";

if (testObjtype[i] != 0) {

cout << testObj[i]->value();

if (testObjtype[i] == 1)

cout << "\t(IdentStr)";

else

cout << "\t(DecStr)";

}

cout << endl;

}

system("pause");

return true;

}