ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО»

Факультет безопасности информационных технологий

Дисциплина:

«Программирование»

ОТЧЕТ ПО ОБОБЩЁННОЙ ЛАБОРАТОРНОЙ РАБОТЕ

Разработка и иерархия классов

Выполнил:
Студент группы N3153
Пастухова А. А.

Проверил:
Кандидат технических наук
Безруков В.А.

Санкт-Петербург

2021г.

Оглавление

1 Введение

Цель: обучение основам ООП на примере языка C++ при помощи создания базового класса и его производных классов.

Задача: разработать базовый класс (строка) и наследственные от него (строка идентификатор, десятичная строка) согласно техническому заданию, а также разработать отображение на экран результатов программы и тестирования всех методов созданных классов.

На сегодняшний день существует множество различных парадигм программирования, каждая из которых имеет свои достоинства и недостатки. Однако основным стандартом при разработке программного обеспечения в наше время стало объектно-ориентированное программирование (далее «ООП»).

ООП - методология программирования, основанная на представлении программы в виде совокупности объектов, каждый из которых является экземпляром определенного класса, а классы образуют иерархию наследования. Данная парадигма представляет разработчику возможность создания очень гибких приложений, имеющих понятную структуру и возможность быстрой доработки, которая не затрагивает все приложение целиком.

2 Техническое задание

2.1 Базовый класс СТРОКА

Обязательные члены класса:

указатель на char - хранит адрес динамически выделенной памяти для размещения символов строки;

значение типа int - хранит длину строки в байтах.

Обязательные методы должны выполнять следующие действия:

конструктор без параметров;

конструктор, принимающий в качестве параметра Си-строку (заканчивается нулевым байтом);

конструктор, принимающий в качестве параметра символ (char);

конструктор копирования;

деструктор;

проверка, пуста ли строка.

2.2 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.

Обязательные методы:

конструктор без параметров;

конструктор, принимающий в качестве параметра Си-строку (заканчивается нулевым байтом);

конструктор, принимающий в качестве параметра символ (char);

конструктор копирования;

деструктор;

поиск последнего вхождения символа в строку.

Переопределить следующие операции:

присваивание (=);

оператор == - проверка на равенство. Строка считается равной другой, если длины строк совпадают, и код каждого символа первой строки в i-й позиции (i изменяется от 0 до n-1, где n - длина строки) равен коду символа в той же позиции во второй строке.

2.3 Производный от СТРОКА класс ДЕСЯТИЧНАЯ_СТРОКА

Строки данного класса могут содержать только символы десятичных цифр и символы - и +, задающие знак числа. Символы - или + могут находиться только в первой позиции числа, причем символ + может отсутствовать, в этом 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;

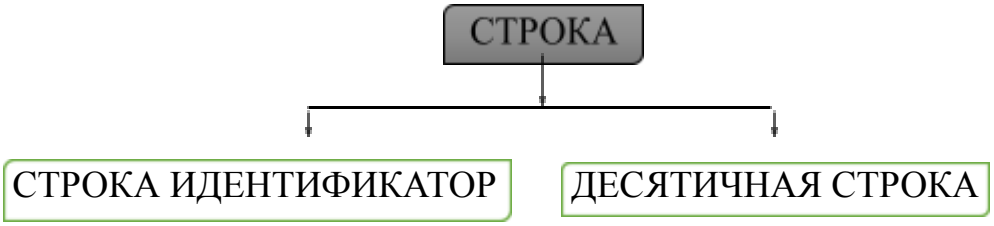
Переопределить следующие операции:

присваивание (=);

вычитание - - арифметическая разность строк;

2.4 Структура программы

Разработать иерархию классов по следующей схеме:



с обязательной поддержкой заданных членов и методов. 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 вызовов в заданное место дисплея, (рекомендуется использовать статические члены класса).

Режим диалога обеспечивается с помощью иерархического меню.

Основные пункты:

1. "Инициализация".

   Подпункты:

   1.1. "Число элементов" - задает число элементов в массиве указателей на базовый класс. Запрещается после задания числа элементов пользоваться этим пунктом меню.

1.2. "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.

2. "Тестирование"

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

   2.1. "Строка"

   2.2. "СТРОКА ИДЕНТИФИКАТОР"

   2.3. "ДЕСЯТИЧНАЯ СТРОКА"

   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.

3. "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.

3 Теория

Название класса: Stroka. Является базовым классом для классов БИТОВАЯ_СТРОКА и СТРОКА_ИДЕНТИФИКАТОР. [1] Все методы прокомментированы в исходном коде и, более того, понять их предназначение можно по названию. Код, в котором описывается класс, разбит на два файла. Первый – Stroka.h – заголовочный файл, в нем объявлены данные-члены, методы и конструкторы класса, представленные ниже. Второй – Stroka.cpp – содержит тело функций-методов и конструкторов, которые были объявлены в заголовочном файле.

3.1 Базовый класс СТРОКА

Данные-члены класса, объявленные как protected: [2]

1) char * pCh – указатель на динамически выделенную память для размещения символов строки;
2) int len – переменная, хранящая длину строки.

Методы класса stroka, объявленные как public:

1) Stroka (int = 0) – конструктор без параметров, создает строку нулевой длины или с параметром задающий длину строки (для переменной pCh выделяется память под символ конца строки ‘\0’).
2) Stroka (ch) – конструктор, принимающий в качестве параметра один символ. Точно известно, что длина строки = 1, значит len = 2, pCh - содержит принятый символ и символ ‘\0;
3) Stroka (conat char*) – конструктор, в качестве параметра принимающий Си-строку. Конструктор с помощью функции Getlen () определяет длину строки и присвает это значение переменной len. После этого выделяется память нужной длины +1 (здесь учитывается символ конца строки ‘\0’), а pCh теперь содержит строку, которую только что принял конструктор;
4) Stroka (const Stroka &) – конструктор копирования. Конструктор с помощью функции Getlen() определяет длину строки и присвает это значение переменной len. После этого выделяется память длины len +1 (здесь

учитывается символ конца строки ‘\0’), а в pCh теперь хранится та строка, которую он скопировал из принятой;

5) ~ Stroka ( ) – деструктор. Очищает память из переменной pCh с помощью функции delete[];
6) char* Getstr(void)const – функция, возвращающая Си-строку;
7) int Getlen(void)const – функция получения длины строки
8) void Show(void) – функция вывода значения и длины строки;

3.2 СТРОКА_ИДЕНТИФИКАТОР

Название класса: IdentStr. Данный класс является производным классом класса Stroka.

Методы класса: [3]

1) IdentStr (int = 0) – конструктор без параметров, создает строку нулевой длины или с параметром задающий длину строки, унаследован от базового класса.
2) IdentStr (ch) – конструктор, принимающий в качестве параметра один символ;
3) IdentStr (const char*) – конструктор, в качестве параметра принимающий Си-строку, проверяет, удовлетворяет ли принятая строка условиям построения идентификатора, а именно:
   - Может содержать латинские буквы верхнего и нижнего регистра
   - Может содержать цифры
   - Может содержать символ подчеркивания
   - Не может начинаться с цифры

   Если не выполнено хотя бы одно условие, то создается строка длины len = 0, содержащая только нулевой элемент pCh [0] = ‘\0’.;
4) IdentStr (const IdentStr &) – конструктор копирования;
5) ~ IdentStr ( ) – деструктор;
6) IdentStr operator = (const IdentStr &) – перегрузка оператора присваивания типа «объект» = «объект». Очищает строку pCh с помощью функции delete[]. Далее

подсчитывается, используя функцию Getlen() класса Stroka, длина той строки, которую необходимо присвоить. Под нее выделяется память с учетом символа конца строки, после чего в pCh помещается принятая строка;

7) operator ~() – реверс строки;
8) char& operator [](int) – перегрузка оператора индексации;
9) friend stroka_id operator + (const stroka_id&, const stroka_id&);
   friend stroka_id operator + (const stroka_id&, const char*);
   friend stroka_id operator + (const char*, const stroka_id&);
   – дружественные функции, перегрузки оператора вычитания. Функция подсчитывает длину введенных строк и выделяет память под новую строку + символ конца строки. Далее в цикле заполняется первая часть итоговой строки, а с помощью указателя мы контролируем конец первой строки. Как только она полностью скопируется, функция начинает копировать правую часть в таком же цикле, только с места остановки. В конце добавляется символ ‘\0’. Данная операция реализована с помощью функций Getlen() и strlen() базового класса Stroka.

3.3 ДЕСЯТИЧНАЯ_СТРОКА

Наименование класса: DecStr. Данный класс является производным классом класса Stroka. [4]

Методы данного класса:

1) DecStr (int = 0) – конструктор без параметров, создает строку нулевой длины или с параметром задающий длину строки.
2) DecStr (char) – конструктор, принимающий в качестве параметра один символ;
3) DecStr (const char*) – конструктор, в качестве параметра принимающий Си-строку;
4) DecStr (const DecStr &) – конструктор копирования;
5) ~ DecStr ( ) – деструктор;

6) DecStr& operator = (const DecStr&) – перегрузка оператора присваивания;
7) friend DecStr operator - (const DecStr&, const DecStr&) – дружественная функция, перегрузка оператора вычитания;
8) DecStr operator + (const DecStr&, const int) - дружественная функция, перегрузка оператора сложения типов «строка» + «число»
9) DecStr operator + (const DecStr&, const DecStr&) - дружественная функция, перегрузка оператора сложения типов «строка» + «строка»
10) bool operator == (const DecStr&, const DecStr&) – оператор равенства.

3.4 Ассемблерная вставка

Согласно техническому заданию, при выполнении данной работы необходимо создать подключаемый к основной программе модуль на ассемблере. Выбор действий, реализуемых в данном модуле, предоставлен исполнителю работы. При выполнении поставленной задачи, был реализован модуль на ассемблере, выполняющий сложение двух целых числе типа int (dword).

3.5 Тестирование

Все методы протестированы в консольном приложении. Функции тестирования методов запускается в main(). Данная функция позволяет продемонстрировать работу всех методов с выводом результатов в консоль.

4 Список литературы

1. Безруков, В. А. Win32 API. Программирование / учебное пособие. - СПб: СПбГУ ИТМО, 2009.
2. Березин, Б. И. Начальный курс С и С++ / Б. И. Березин, С. Б. Березин. - М.: ДИАЛОГ-МИФИ, 2004.

3. Литвиненко, Н. А. Технология программирования на C++. Win32 API-приложения. - СПб.: БХВ-Петербург, 2010.
4. Финогенов, К.Г. Win32. Основы программирования - М.: Диалог-МИФИ, 2006.

5 Заключение

Цель курсовой работы достигнута. Программа разработана в соответствии с техническим заданием на языке C++. Были созданы тестируемые классы: базовый класс Строка, Строка-идентификатор, Десятичная строка. Также были реализованы и протестированы различные методы созданных классов. Вывод работы программы можно увидеть в консольном приложении.

6 Приложение

6.1 Файл main.cpp

```
#include "Stroka.h"
#include "IdentStr.h"
#include "DecStr.h"
#include <iostream>


using namespace std;

int main()
{
	Stroka obj1(0);
	obj1.Show();
	Stroka obj2('5');
	obj2.Show();
	Stroka obj3("itmo");
	obj3.Show();
	Stroka obj4 = obj3;
	obj4.Show();
	cout << " ---------------" << endl << endl;

	IdentStr obj5(0);
	obj5.Show();
	IdentStr obj6('f');
	obj6.Show();
	IdentStr obj7("world");
	obj7.Show();
	IdentStr obj8 = obj7;
	obj8.Show();
	IdentStr obj9 = ~obj8;
	obj9.Show();
	IdentStr obj10 = ("ITMO_");
	IdentStr obj11 = ("UNIVERSITY");
	obj10 = obj10 + obj11;
	obj10.Show();
	obj11 = obj6 + obj11;
	obj11.Show();
	IdentStr obj12 = obj10 + obj6;
	obj12.Show();
	bool obj15 = obj11 < obj10;
	cout << " ---------------" << endl << endl;

	DecStr obj16(0);
	obj16.Show();
	DecStr obj17('p');
	obj17.Show();
	DecStr obj18("9+7");
	obj18.Show();
	DecStr obj19("787");
	DecStr obj20 = obj19;
	obj20.Show();
	DecStr obj21("13");
	obj21 = obj21 + obj19;
	obj21.Show();
	DecStr obj22('3');
	obj22 = obj22 + 100;
	obj22.Show();
	DecStr obj23 = obj19 - obj21;
```

```
        obj23.Show();
        bool obj24 = (obj21 == obj19);
        cout << " ---------------" << endl << endl;


        //создание объекта через указатель
        // выделение памяти динамически
        Stroka** ptrs = new Stroka* [6];

        ptrs[0] = new Stroka("FBIT");
        ptrs[1] = new Stroka("constructurs");
        ptrs[2] = new IdentStr("GoodMorning");
        ptrs[3] = new IdentStr("Good_Night");
        ptrs[4] = new DecStr("123");
        ptrs[5] = new DecStr("80");

        ptrs[0]->Show();
        Stroka* obj100 = dynamic_cast <Stroka*>(ptrs[0]); //динамическое приведение типа
        obj100->Show();
        Stroka* obj200 = dynamic_cast <Stroka*>(ptrs[1]);
        *(obj100) = *(obj200);
        obj100->Show();

        ((IdentStr*)ptrs[2])->Show();
        ((IdentStr*)ptrs[3])->Show();

        ((DecStr*)ptrs[4])->Show();
        ((DecStr*)ptrs[5])->Show();

        delete[] ptrs; //очистка ptrs

        return 0;
}
```

6.2 Файл Stroka.h

```
#ifndef __STROKA__
#define __STROKA__
class Stroka
{
protected:
	int len;
	char* pCh;
public:
	Stroka(int = 0);
	Stroka(char);
	Stroka(const char*);
	Stroka(const Stroka&);
	~Stroka();
	char* Getstr(void)const
	{
		return pCh;
	}
	int Getlen(void)const
	{
		return len;
	}
	void Show(void);
};
#endif // !__STROKA__
```

6.3 Файл Stroka.cpp

```
#include "Stroka.h"
#include <iostream>

using namespace std;

Stroka::Stroka(int val) :len(val), pCh(new char[len + 1])
{
	if (val == 0) pCh[0] = '\0';
	cout << "Zero string was created SUCCESSFULLY!" << endl;
}
Stroka::Stroka(char ch) :len(1), pCh(new char[len + 1])
{
	pCh[0] = ch, pCh[1] = '\0';
	cout << "A memory for character was allocated SUCCESSFULLY!" << endl;
}
Stroka::Stroka(const char* S) :len(strlen(S)), pCh(new char[len + 1])
{
	strcpy_s(pCh, len + 1, S);
	cout << "pCh has a string S right now" << endl;
}
Stroka::Stroka(const Stroka& from) : len(strlen(from.pCh)), pCh(new char[from.len + 1])
{
	strcpy_s(pCh, from.len + 1, from.pCh);
	cout << "Copy string constructure" << endl;
}
Stroka::~Stroka()
{
	if (pCh) delete[]pCh;
	cout << "Destructure for string " << &pCh << endl;
}

void Stroka::Show(void)
{
	cout << "pCh = " << pCh << endl;
	cout << "len = " << len << endl;
}
```

6.4 Файл IdentStr.h

```
#ifndef __IdentStr__
#define __IdentStr__
#include "Stroka.h"
class IdentStr :public Stroka
{
public:
	IdentStr(int = 0);
	IdentStr(char);
	IdentStr(const char*);
	IdentStr(const IdentStr&);
	~IdentStr();
	IdentStr& operator=(const IdentStr&);
	char&operator[](int);
	IdentStr operator~();
	friend IdentStr operator +(const IdentStr&, const IdentStr&);
	friend IdentStr operator +(const char*, const IdentStr&);
	friend IdentStr operator +(const IdentStr&, const char*);
	friend bool operator < (const IdentStr&, const IdentStr&);
};
#endif // !__IdentStr__
```

6.5 Файл IdentStr.cpp

```
#include "Stroka.h"
#include "IdentStr.h"
#include <iostream>

using namespace std;


IdentStr::IdentStr(int val) :Stroka(val)
{
	cout << "Identify a zero string SUCCESSFULLY!" << endl;
}

IdentStr::IdentStr(char ch) : Stroka(ch)
{
	if (!((pCh[0] >= 'a' && pCh[0] <= 'z') || (pCh[0] >= 'A' && pCh[0] <= 'Z')))
	{
		cout << "Bad symbol pCh[0]=" << pCh[0] << endl;
		if (pCh) delete[] pCh;
		len = 0;
		pCh = new char[len + 1];
		pCh[0] = '\0';
		return;
	}
	cout << "Correct symbol" << endl;
}
IdentStr::IdentStr(const char* str) :Stroka(str)
{
	const char* keyword[] = { "alignas", "alignof", "and", "and_eq","asm",
"auto","bitand","bitor","bool", "break", "case", "catch", "char", "char16_t", "har32_t", "class",
"compl", "const", "constexpr","const_cast", "continue", "decltype", "default", "delete", "do",
"double", "dynamic_cast", "else", "enum", "explicit", "export","extern", "false", "float", "for",
"friend", "if", "inline", "int", "long", "mutable", "namespace", "new", "noexcept", "not",
"not_eq","nullptr", "operator", "or", "or_eq", "private", "protected", "public", "register",
"reinterpret_cast", "return", "short", "signed", "sizeof","static", "static_assert",
"static_cast", "struct", "switch", "template", "this", "thread_local", "throw", "true", "try",
"typedef", "typeid", "typename","union", "unsigned", "using", "virtual", "void", "volatile",
"wchar_t", "while", "xor", "xor_eq" };
	for (int i = 0; i < 83; i++) {
		if (strcmp(pCh, keyword[i]) == 0) {
			if (pCh) delete[] pCh;
			len = 0;
			pCh = new char[len + 1];
			pCh[0] = '\0';
			cout << "Bad stroka because of word" << endl;
			return;
		}
	}
	for (int i = 0; i < len; i++)
	{
		if (!((pCh[i] >= 'a' && pCh[i] <= 'z') || (pCh[i] >= 'A' && pCh[i] <= 'Z') ||
(pCh[i] == '_')))
		{
			if (!(pCh[0] >= '0' && pCh[0] <= '9'))
			{
				cout << "Bad stroka because of symbol pCh[i]=" << pCh[i] << endl;
				if (pCh) delete[]pCh;
				len = 0, pCh = new char[len + 1];
				pCh[0] = '\0';
				return;
```

```
                        }
                        cout << "Bad stroka because of symbol pCh[i]=" << pCh[i] << endl;
                        return;
                }
        }
        cout << "Correct string" << endl;
}
IdentStr::IdentStr(const IdentStr& from) :Stroka(from)
{
        cout << "Copy string by :Stroka(from)" << endl;
}
IdentStr::~IdentStr()
{
        cout << "A string < " << pCh << " > was deleted SUCCESSFULLY!" << endl;
}
IdentStr& IdentStr:: operator = (const IdentStr& S)
{
        if (&S != this)
        {
                delete[]pCh;
                len = strlen(S.pCh);
                pCh = new char[len + 1];
                strcpy_s(pCh, len + 1, S.pCh);
        }
        cout << "Operator = was overloaded  SUCCESSFULLY for IdentStr& S" << endl;
        return *this;
}

IdentStr IdentStr::operator ~()
{
        char tmp;
        int i, j;
        for (i = 0, j = (len - 1); i < len / 2; i++, j--)
        {
                tmp = pCh[i];
                pCh[i] = pCh[j];
                pCh[j] = tmp;
        }
        cout << "Reverse string was done SUCCESSFULLY!" << endl;
        return *this;
}
IdentStr operator + (const IdentStr& pobj1, const IdentStr& pobj2)
{
        IdentStr tmp(pobj1.Getlen() + pobj2.Getlen());
        int i = 0, j = 0;
        while (tmp.pCh[i++] = pobj1.pCh[j++]);
        --i;
        j = 0;
        while (tmp.pCh[i++] = pobj2.pCh[j++]);
        cout << "Operator + (const IdentStr& pobj1, const IdentStr& pobj2)  was overloaded
SUCCESSFULLY!" << endl;
        return tmp;
}
IdentStr operator + (const char* str, const IdentStr& pobj2)
{
        IdentStr tmp((int)strlen(str) + pobj2.Getlen());
        int i = 0, j = 0;
        while (tmp.pCh[i++] = str[j++]);
        --i;
        j = 0;
        while (tmp.pCh[i++] = pobj2.pCh[j++]);
```

```
	cout << "Operator + (const char* str, const IdentStr& pobj2)  was overloaded
SUCCESSFULLY!" << endl;
	return tmp;
}
IdentStr operator + (const IdentStr& pobj1, const char* str)
{
	IdentStr tmp1(str);
	IdentStr tmp(pobj1.Getlen() + tmp1.Getlen());
	int i = 0, j = 0;
	while (tmp.pCh[i++] = pobj1.pCh[j++]);
	--i;
	j = 0;
	while (tmp.pCh[i++] = tmp1.pCh[j++]);
	cout << "Operator + (const IdentStr& pobj1, const char* str)  was overloaded
SUCCESSFULLY!" << endl;
	return tmp;
}
char& IdentStr::operator[](int index)
{
	if (index >= 0 && index < len)
	{
		cout << "Operator[](int index) was overloaded  SUCCESSFULLY!" << endl;
		return pCh[index];
	}
	else return pCh[0];
}
bool operator < (const IdentStr& pobj1, const IdentStr& pobj2)
{
	if (&pobj1 == &pobj2)
	{
		return false;
	}
	for (int i = 0; (i < pobj1.Getlen()) && (i < pobj2.Getlen()); i++)
	{
		if (pobj1.pCh[i] < pobj2.pCh[i])
		{
			return true;
		}
		else return false;
	}
	return false;
}
```

6.6 Файл DecStr.h

```
#ifndef __DECSTR__
#define __DECSTR__
#include "Stroka.h"
class DecStr : public Stroka
{
public:
	DecStr(int = 0);
	DecStr(char);
	DecStr(const char*);
	DecStr(const DecStr&);
	~DecStr();
	DecStr& operator = (const DecStr&);
	friend DecStr operator + (const DecStr&, const DecStr&);
	friend DecStr operator + (const DecStr&, const int);
	friend DecStr operator - (const DecStr&, const DecStr&);
	friend bool operator == (const DecStr&, const DecStr&);
};
#endif // !__DECSTR__
```

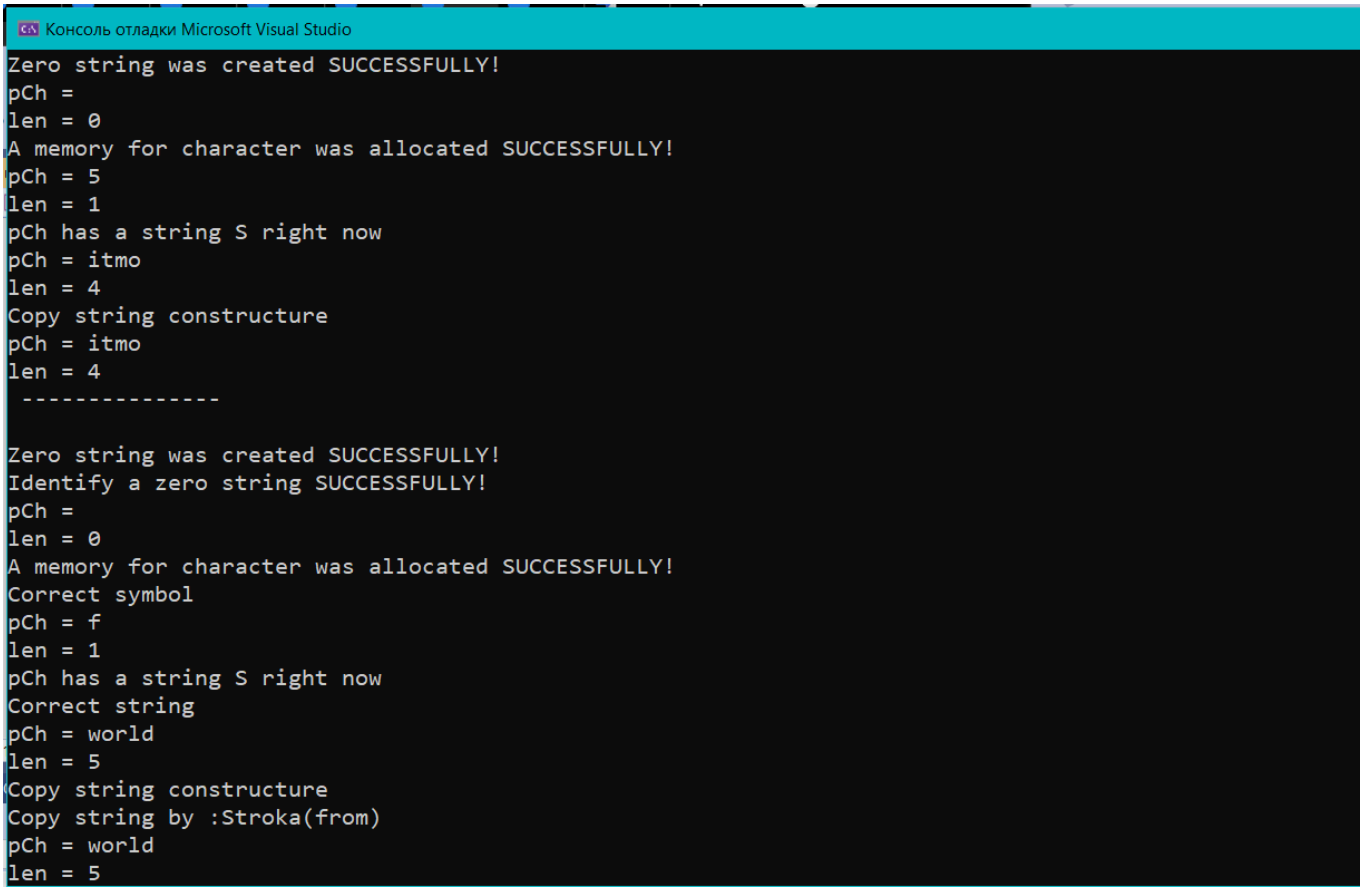
6.7 Файл DecStr.cpp

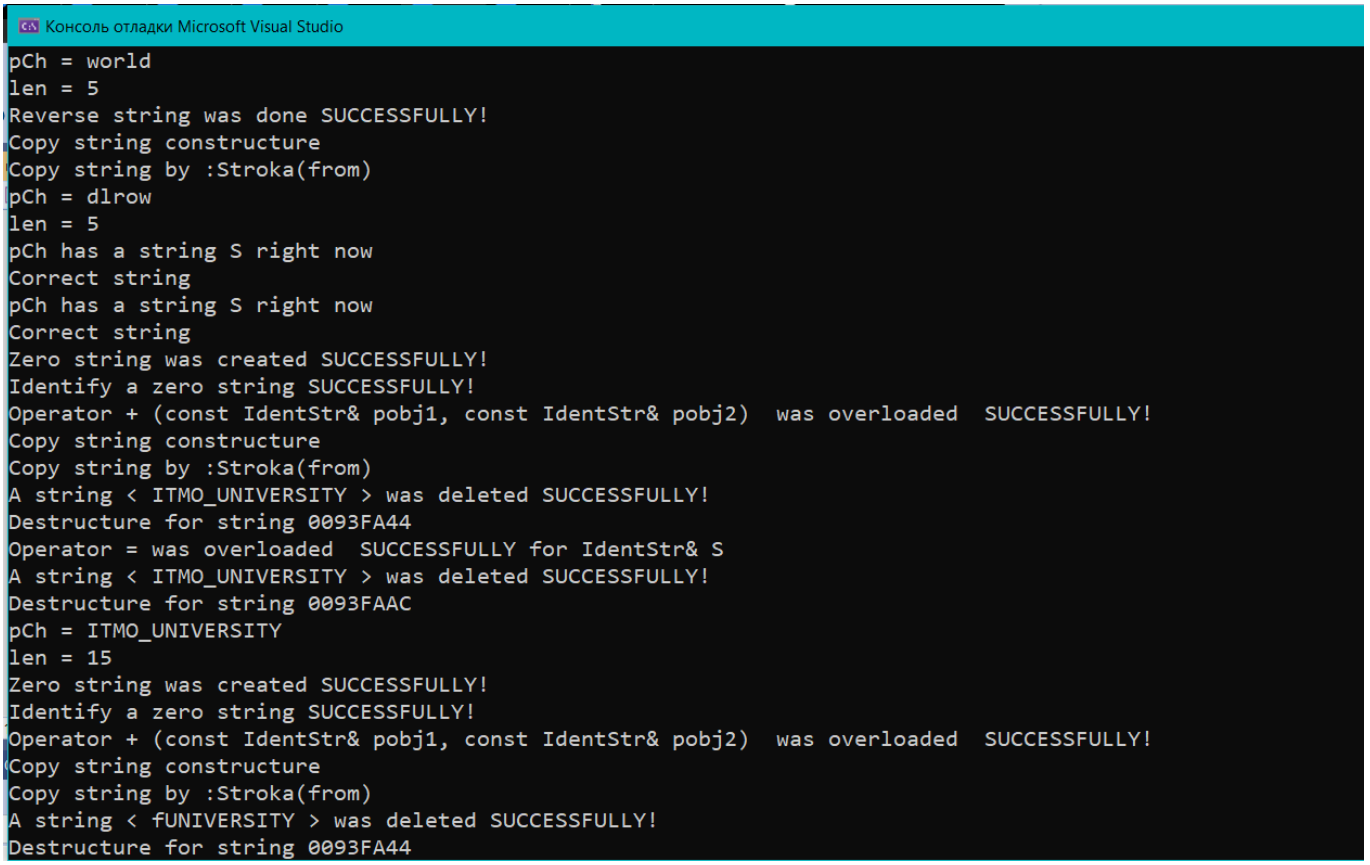
```
#include "Stroka.h"
#include "IdentStr.h"
#include "DecStr.h"
#include <iostream>

using namespace std;


DecStr::DecStr(int val) :Stroka(val)
{
	cout << "Zero DecStr" << endl;
}

DecStr::DecStr(char ch) : Stroka(ch)
{
	cout << "One Dec char" << endl;
}

DecStr::~DecStr()
{
	cout << "Dec string < " << pCh << " > was deleted SUCCESSFULLY!" << endl;
}

DecStr::DecStr(const char* str) : Stroka(str)
{
	if (!((pCh[0] >= '1' && pCh[0] <= '9') || (pCh[0] == '-') || (pCh[0] == '+')))
	{
		cout << "Bad Dec stirng because of 1 symbol" << endl;
		if (pCh) delete[]pCh;
		len = 0;
		pCh = new char[len + 1];
		pCh[0] = '\0';
		return;
	}
	for (int i = 1; i < len; i++)
	{
		if (!(pCh[i] >= '0' && pCh[0] <= '9'))
		{
			cout << "Bad stroka pCh[" << i << "] = " << pCh[i] << endl;
			{
				if (pCh) delete[]pCh;
				len = 0;
				pCh = new char[len + 1];
				pCh[0] = '\0';
				return;
			}
		}
	}
	cout << "Good Dec string" << endl;
}

DecStr::DecStr(const DecStr& from) :Stroka(from)
{
	cout << "Copy Dec string" << endl;
}

DecStr& DecStr::operator = (const DecStr& Ds)
{
	if (&Ds != this)
```

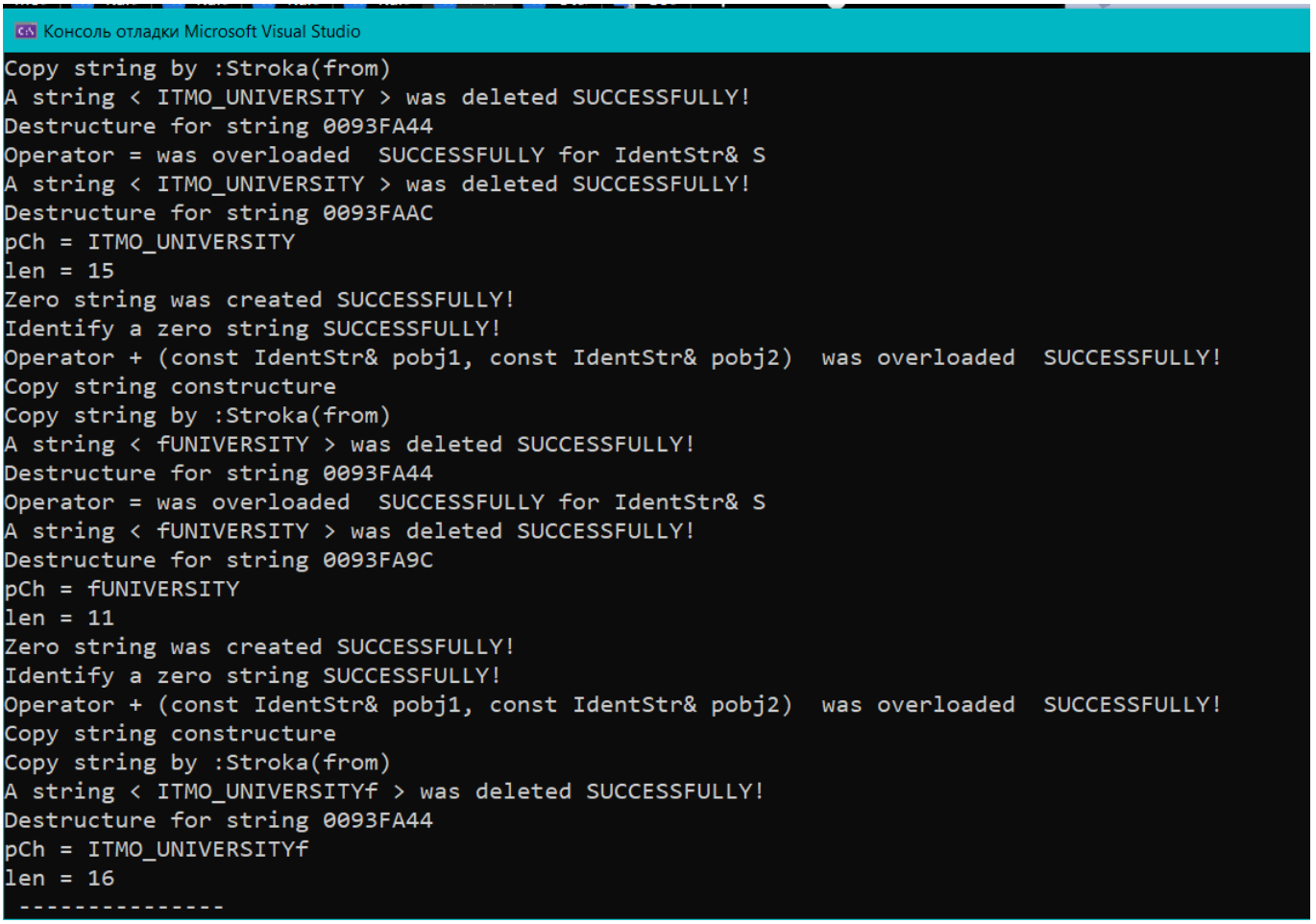
```
	{
		delete[]pCh;
		len = strlen(Ds.pCh);
		pCh = new char[len + 1];
		strcpy_s(pCh, len + 1, Ds.pCh);
	}
	cout << "Operator = (const DecStr& Ds)  was overloaded  SUCCESSFULLY!" << endl;
	return *this;
}

DecStr operator + (const DecStr& pobj1, const DecStr& pobj2)
{
	char str[12];
	char* ptr = str;
	int a = 0, b = 0, ans = 0;
	a = atoi(pobj1.Getstr());
	b = atoi(pobj2.Getstr());

	_asm {
		mov eax, a;
		mov ebx, b;
		add eax, ebx;
		mov ans, eax;
	}
	_itoa(ans, ptr, 10);
	cout << "DecStr operator + for pobj " << endl;
	return DecStr(str);
}

DecStr operator + (const DecStr& pobj, const int obj)
{
	int num1, num2;
	DecStr tmp(pobj);
	num1 = atoi(tmp.Getstr());
	num2 = obj;
	char* pMasCh;
	int A = num1 + num2;
	if (abs(num1) >= abs(num2))
	{
		pMasCh = new char[tmp.len + 1];
		_itoa_s(num2, pMasCh, tmp.len + 1, 10);
	}
	else
	{
		pMasCh = new char[tmp.len + 8];
		_itoa_s(num2, pMasCh, tmp.len + 8, 10);
	}
	char* pTmpCh;
	int len2 = strlen(pMasCh);
	if (tmp.len >= len2)
	{
		pTmpCh = new char[tmp.len + 1];
		_itoa_s(A, pTmpCh, tmp.len + 1, 10);
	}
	else
	{
		pTmpCh = new char[len2 + 1];
		_itoa_s(A, pTmpCh, len2 + 1, 10);
	}

	if (tmp.pCh) delete[]tmp.pCh;
```

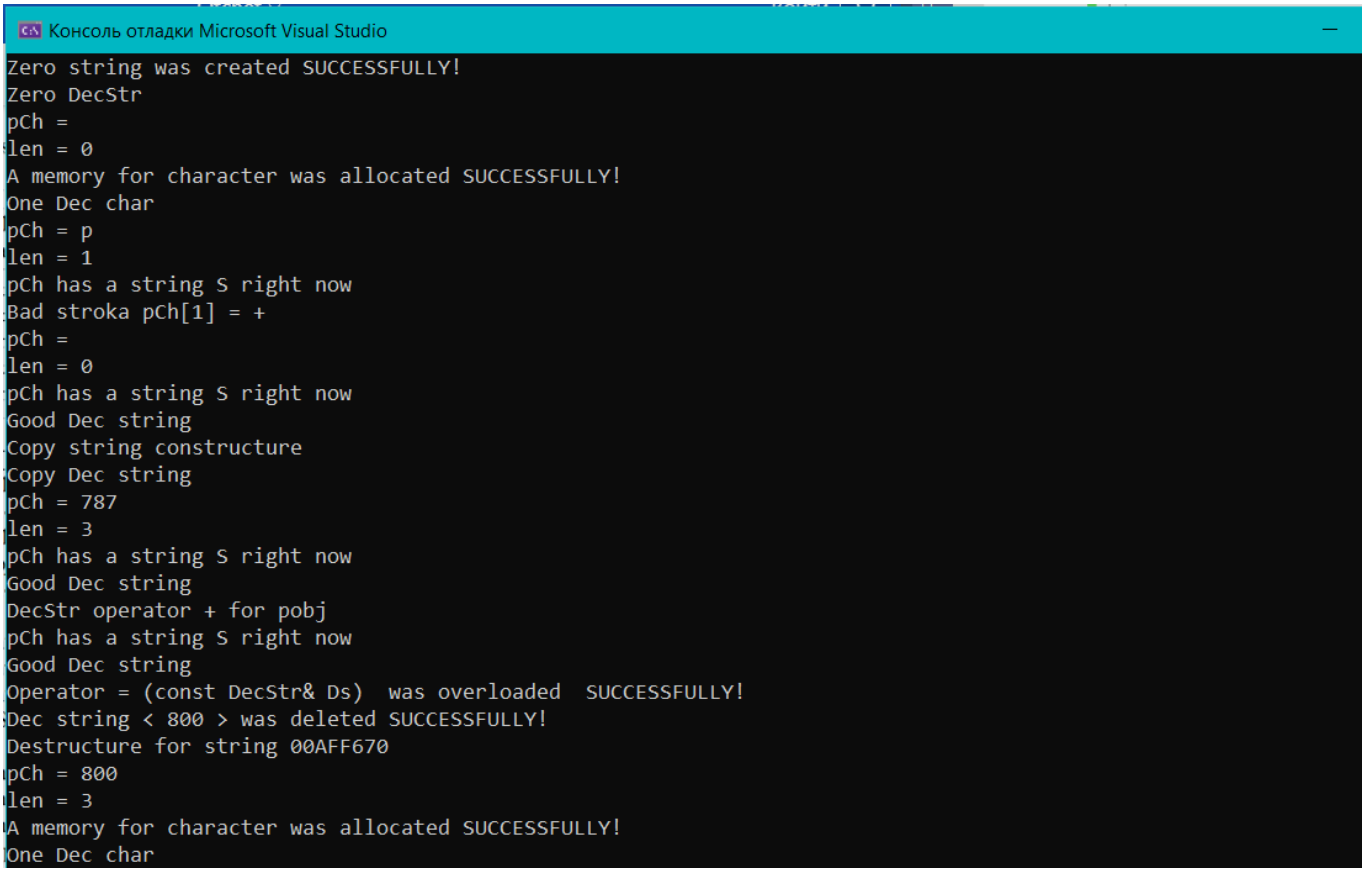
```
	tmp.pCh = pTmpCh;
	tmp.len = strlen(pTmpCh);

	cout << "DecStr operator + for pobj and obj)" << endl;
	return tmp;
}

DecStr operator - (const DecStr& pobj1, const DecStr& pobj2)
{
	int num1, num2;
	DecStr tmp(pobj1);
	num1 = atoi(tmp.Getstr());
	num2 = atoi(pobj2.Getstr());
	char* pTmpCh;
	int A = num1 - num2;
	if (pobj1.len >= pobj2.len)
	{
		pTmpCh = new char[tmp.len + 2];
		_itoa_s(A, pTmpCh, tmp.len + 2, 10);
	}
	else
	{
		pTmpCh = new char[pobj2.len + 2];
		_itoa_s(A, pTmpCh, pobj2.len + 2, 10);
	}
	if (tmp.pCh) delete[]tmp.pCh;
	tmp.pCh = pTmpCh;
	tmp.len = strlen(tmp.pCh);
	cout << "DecStr operator - for pobj)" << endl;
	return tmp;
}

bool operator == (const DecStr& num1, const DecStr& num2)
{
	cout << "bool operator == was done" << endl;
	return (atoi(num1.Getstr()) == atoi(num2.Getstr()));
}
```

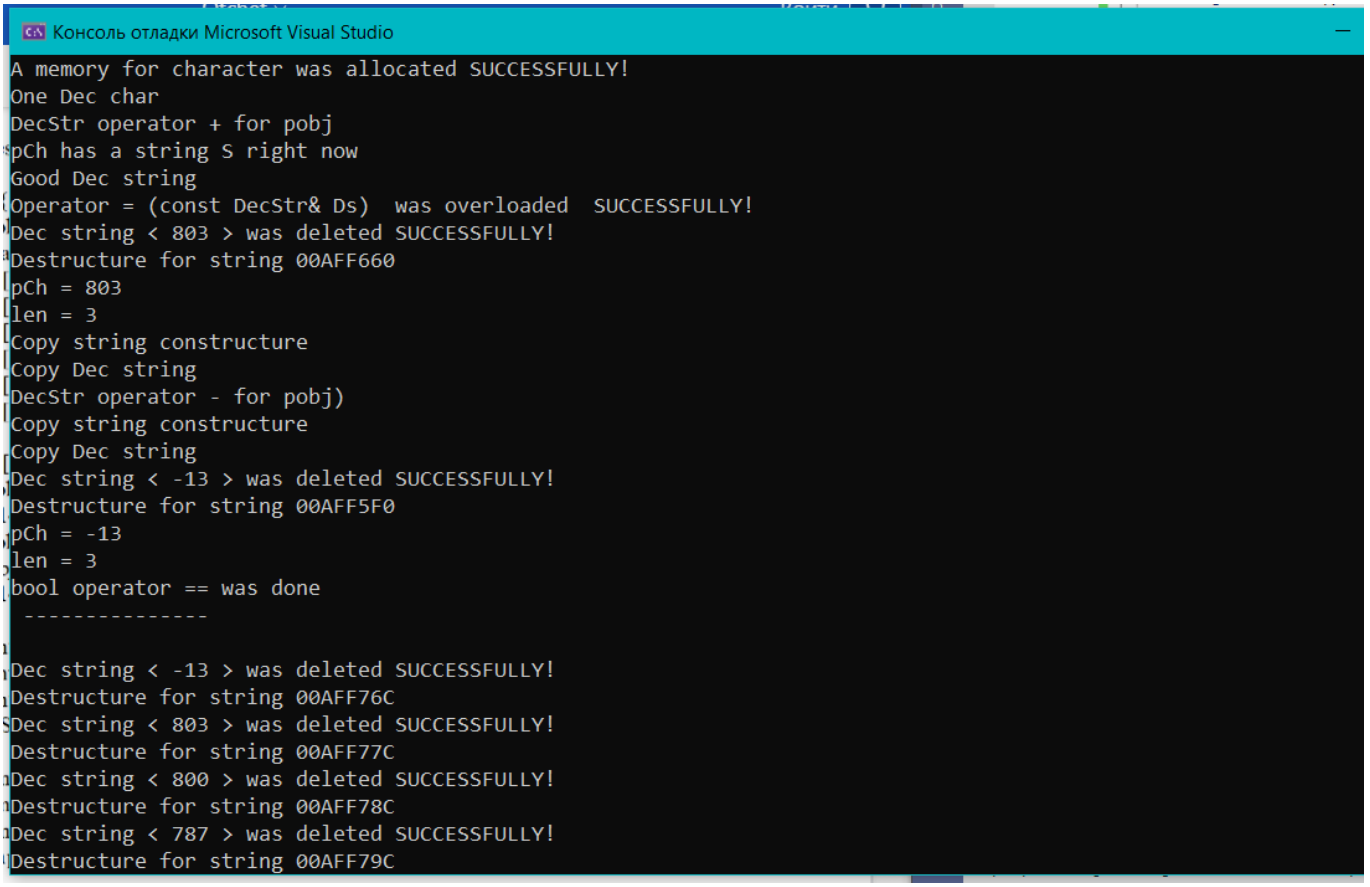
6.8 Скриншоты работы программы



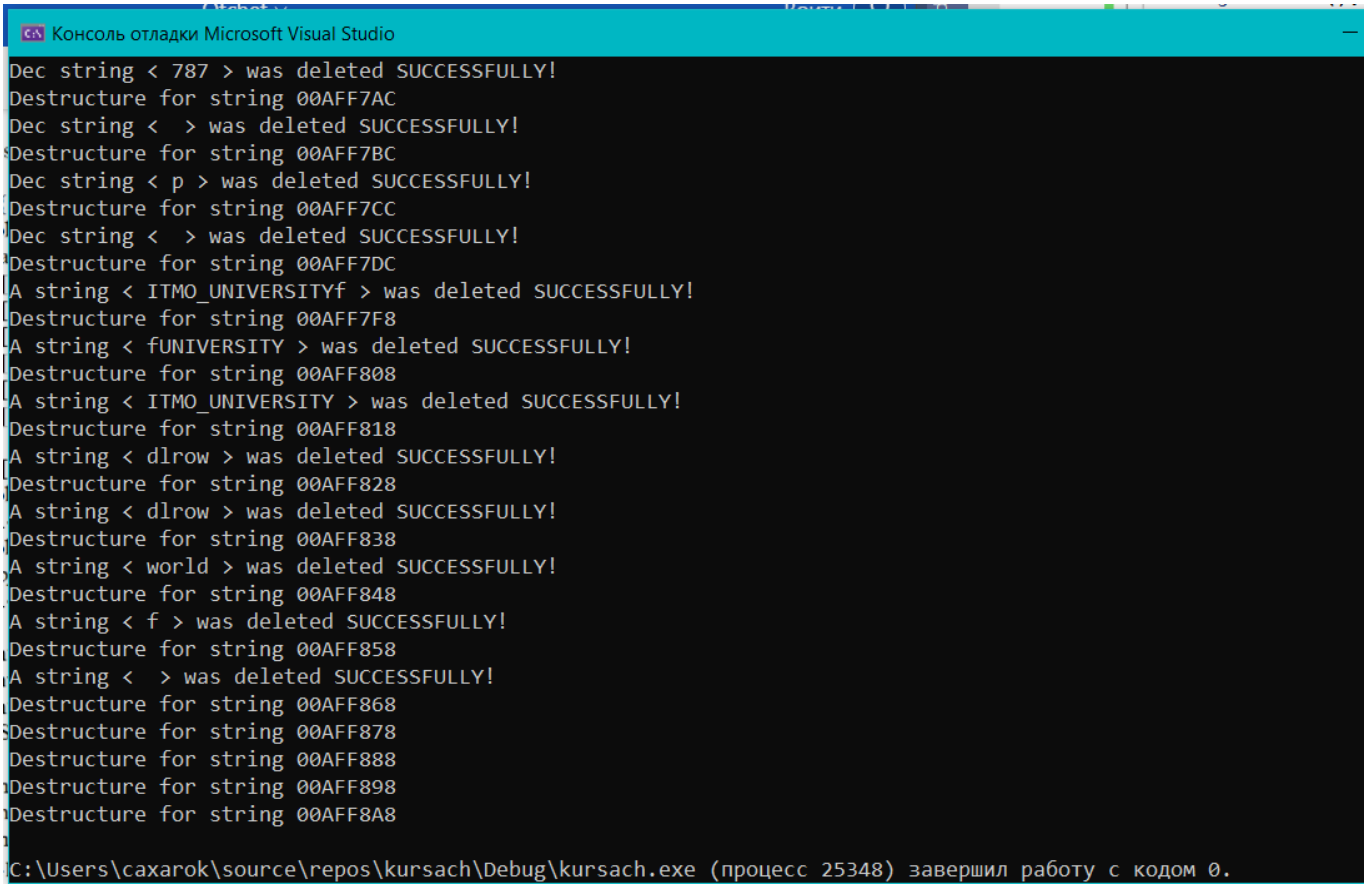



Консоль отладки Microsoft Visual Studio
Copy string by :Stroka(from)
A string < ITMO_UNIVERSITY > was deleted SUCCESSFULLY!
Destructure for string 0093FA44
Operator = was overloaded SUCCESSFULLY for IdentStr& S
A string < ITMO_UNIVERSITY > was deleted SUCCESSFULLY!
Destructure for string 0093FAAC
pCh = ITMO_UNIVERSITY
len = 15
Zero string was created SUCCESSFULLY!
Identify a zero string SUCCESSFULLY!
Operator + (const IdentStr& pobj1, const IdentStr& pobj2) was overloaded SUCCESSFULLY!
Copy string constructure
Copy string by :Stroka(from)
A string < fUNIVERSITY > was deleted SUCCESSFULLY!
Destructure for string 0093FA44
Operator = was overloaded SUCCESSFULLY for IdentStr& S
A string < fUNIVERSITY > was deleted SUCCESSFULLY!
Destructure for string 0093FA9C
pCh = fUNIVERSITY
len = 11
Zero string was created SUCCESSFULLY!
Identify a zero string SUCCESSFULLY!
Operator + (const IdentStr& pobj1, const IdentStr& pobj2) was overloaded SUCCESSFULLY!
Copy string constructure
Copy string by :Stroka(from)
A string < ITMO_UNIVERSITYf > was deleted SUCCESSFULLY!
Destructure for string 0093FA44
pCh = ITMO_UNIVERSITYf
len = 16


Консоль отладки Microsoft Visual Studio
Zero string was created SUCCESSFULLY!
Zero DecStr
pCh =
len = 0
A memory for character was allocated SUCCESSFULLY!
One Dec char
pCh = p
len = 1
pCh has a string S right now
Bad stroka pCh[1] = +
pCh =
len = 0
pCh has a string S right now
Good Dec string
Copy string constructure
Copy Dec string
pCh = 787
len = 3
pCh has a string S right now
Good Dec string
DecStr operator + for pobj
pCh has a string S right now
Good Dec string
Operator = (const DecStr& Ds) was overloaded SUCCESSFULLY!
Dec string < 800 > was deleted SUCCESSFULLY!
Destructure for string 00AFF670
pCh = 800
len = 3
A memory for character was allocated SUCCESSFULLY!
One Dec char


Консоль отладки Microsoft Visual Studio
A memory for character was allocated SUCCESSFULLY!
One Dec char
DecStr operator + for pobj
pCh has a string S right now
Good Dec string
Operator = (const DecStr& Ds) was overloaded SUCCESSFULLY!
Dec string < 803 > was deleted SUCCESSFULLY!
Destructure for string 00AFF660
pCh = 803
len = 3
Copy string constructure
Copy Dec string
DecStr operator - for pobj)
Copy string constructure
Copy Dec string
Dec string < -13 > was deleted SUCCESSFULLY!
Destructure for string 00AFF5F0
pCh = -13
len = 3
bool operator == was done

Dec string < -13 > was deleted SUCCESSFULLY!
Destructure for string 00AFF76C
Dec string < 803 > was deleted SUCCESSFULLY!
Destructure for string 00AFF77C
Dec string < 800 > was deleted SUCCESSFULLY!
Destructure for string 00AFF78C
Dec string < 787 > was deleted SUCCESSFULLY!
Destructure for string 00AFF79C


Консоль отладки Microsoft Visual Studio
Dec string < 787 > was deleted SUCCESSFULLY!
Destructure for string 00AFF7AC
Dec string < > was deleted SUCCESSFULLY!
Destructure for string 00AFF7BC
Dec string < p > was deleted SUCCESSFULLY!
Destructure for string 00AFF7CC
Dec string < > was deleted SUCCESSFULLY!
Destructure for string 00AFF7DC
A string < ITMO_UNIVERSITYf > was deleted SUCCESSFULLY!
Destructure for string 00AFF7F8
A string < fUNIVERSITY > was deleted SUCCESSFULLY!
Destructure for string 00AFF808
A string < ITMO_UNIVERSITY > was deleted SUCCESSFULLY!
Destructure for string 00AFF818
A string < dlrow > was deleted SUCCESSFULLY!
Destructure for string 00AFF828
A string < dlrow > was deleted SUCCESSFULLY!
Destructure for string 00AFF838
A string < world > was deleted SUCCESSFULLY!
Destructure for string 00AFF848
A string < f > was deleted SUCCESSFULLY!
Destructure for string 00AFF858
A string < > was deleted SUCCESSFULLY!
Destructure for string 00AFF868
Destructure for string 00AFF878
Destructure for string 00AFF888
Destructure for string 00AFF898
Destructure for string 00AFF8A8
C:\Users\caxarok\source\repos\kursach\Debug\kursach.exe (процесс 25348) завершил работу с кодом 0.