**Новосибирский Государственный Технический Университет**

***Факультет автоматики и вычислительной техники***

**Лабораторная работа № 2  
по курсу «Программирование»**

**«Переопределение операций»**

**Вариант №2**

Студентов 2-го курса  
факультета АВТФ группы АБ-521  
Рудковского А.А.

Крошакова Е.В.

Проверил Бычков М.И.

Новосибирск-2016г

**Цель работы**: Для класса из лаб. работы №1 реализовать набор операций для работы с объектами класса: по заданию. Изменить демонстpационную пpогpамму, продемонстрировав все перегруженные операции. Операции перегрузить методами класса и дружественными функциями.

**Задание**:

1. Переопределить следующие операторы: +, -, (), =.
2. Для оператора ‘+’ определим следующие две операции: сложение со строкой типа char\*, сложение со строкой типа Str.
3. Для оператора ‘-‘ определить операцию удаления подстроки из строки.
4. Для оператора ‘()’ определить операцию получения подстроки по индексу начала и длине подстроки.
5. Для оператора ‘=’ определить операцию присваивания.

**Ход работы**:

1. Для реализации оператора, надо объявить функцию с ключевым словом operator@, где @ сам перегружаемый оператор.
2. Для перегрузок будем использовать методы класса. Их параметром является правый операнд в операции. От использование дружественных методов откажемся, так как в задании имеется перегрузка операции присваивания, которая не допускает перегрузки через дружественную функцию.
3. Для оператора + нам понадобиться два перегруженных метода. Один для работы со строками, другой для работы с объектами.
4. Для оператора – нам понадобиться один метод. Его мы реализуем как удаление подстроки из строки.
5. Для оператора () нам понадобиться один метод класса.

**Вывод**: В ходе работы предполагается получение практических навыков по переопределению операций.

Приложение 1. Заголовочный файл string.h  
#pragma once

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <list>

#include <ctime>

#pragma once

#define LEN 3

class Str

{ public:

char\* arr;

int maxlen;

time\_t create;

char str[256];

int len;

static std::list<Str\*> allstr;

static void PrintAll();

Str();

Str(const char\*);

Str(int max, char\* strarr);

Str(const Str& copy);

~Str();

void SetString(const char\*);

int FindSubstring(const char\*);

int Length();

char\* Substring(char\* sub);

void Print();

void getStr();

Str operator +(char\*);

Str operator +(Str);

Str operator-(const char\*);

Str operator()(int, int);

Str&operator=(const Str&);

};

Приложение 2. Файл исходного кода string.cpp

#define \_CRT\_SECURE\_NO\_WARNINGS

#include "String.h"

#include <iostream>

#include <string.h>

std::list<Str\*> Str::allstr = std::list<Str\*>();

void Str::PrintAll()

{

std::cout << "Total amount of objects: " << allstr.size() << "\nNow print all objects:\n";

for (auto it = allstr.begin(); it != allstr.end(); it++)

std::cout << it.\_Ptr->\_Myval->arr + 3 << "\n";

}

void Str::SetString(const char \*s)

{

if (arr) {

free(arr);

arr = nullptr;

}

arr = \_strdup(s);

len = strlen(arr);

}

int Str::FindSubstring(const char\*s) {

char\*f = strstr(arr, s);

if (f)

return f - arr;

return -1;

}

Str::Str()

{

arr = "0\0";

create = time(NULL);

allstr.insert(allstr.end(), this);

}

Str::Str(const char \*s)

{

arr = nullptr;

SetString(s);

}

char\* Str::Substring(char\* substr)

{

return strstr(arr+LEN, substr);

}

Str::Str(int max, char\* strarr)

{

arr = new char[max + LEN];

maxlen = max;

strcpy\_s(arr + 3, max, strarr);

int real\_len = 0;

while (strarr[real\_len] != '\0')

real\_len++;

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

{

arr[i] = (real\_len % 10) + '0';

real\_len /= 10;

}

create = time(NULL);

allstr.insert(allstr.end(), this);

}

Str::Str(const Str& copy)

{

maxlen = copy.maxlen;

arr = new char[maxlen + LEN];

memcpy(arr, copy.arr, maxlen);

time(&create);

allstr.insert(allstr.end(), this);

}

Str::~Str()

{

}

int Str::Length()

{

int len = 0;

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

{

len \*= 10;

len += (int)(arr[i] - '0');

}

return len;

}

void Str::Print()

{

std::cout << arr+LEN;

}

Str Str::operator +(char \*s)

{

strcat(str, s);

return(\*this);

}

Str Str::operator+(Str par)

{

strcat(str, par.str);

return(\*this);

}

Str Str::operator-(const char \*s)

{

char\* p = 0, \*p1 = 0;

size\_t size = strlen(s);

for (; p = strstr(arr, s); )

{

p1 = p + size;

while (\*p++ = \*p1++);

}

std::cout << arr;

return Str(\*this);

}

Str Str::operator()(int offs, int len )

{

char\*temp = (char\*)calloc(1, len + 1);

memcpy(temp, &arr[offs], len);

Str ret(temp);

std::cout << temp;

free(temp);

return ret;

}

Str & Str::operator=(const Str &s)

{

if (this == &s)return \*this;

if (arr)free(arr);

arr = \_strdup(s.arr);

len = s.len;

std::cout << "First is now:" << arr;

return \*this;

}

void Str::getStr()

{

std::cout << str << "\n";

}

Приложение 3. Файл исходного кода main.cpp, демонстрация работы класса

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <iostream>

#include <conio.h>

#include "String.h"

#include <string.h>

#include "Interface.h"

using namespace std;

void testconstructor()

{

char\*s = (char\*)calloc(256, 1);

Str::allstr.insert(Str::allstr.end(), new Str());

Str::allstr.clear();

cout << "Enter string: ";

char\* input = new char[80];

cin.ignore();

cin.getline(input, 80);

Str\* test = new Str(50, input);

cout << "Real length: " << test->Length() << "\n";

cout << "Enter substring: ";

cin.getline(input, 80);

cout << "Substring by input: " << test->Substring(input) << "\n";

cout << "Full string: ";

test->Print();

cout << "\n";

Str\* default = new Str();

Str\* copy = new Str(\*test);

cout << "\n\n\nStatic: \n";

Str::PrintAll();

\_getch();

}

void testplusstr() {

cout << "Overwrite operator + strings:" << endl;

char \*str1 = new char[80];

char \*str2 = new char[80];

cin.ignore();

cout << "Enter 1 string: ";

cin.getline(str1, 80);

cout << "Enter 2 string: ";

cin.getline(str2, 80);

strcpy(str1, str1);

strcpy(str2, str2);

Str plus = Str();

plus + str1;

plus + str2;

plus.getStr();

}

void testplusobj()

{

cout << "Overwrite operator + objects:" << endl;

char \*str3 = new char[80];

char \*str4 = new char[80];

cin.ignore();

cout << "Enter 1 string: ";

cin.getline(str3, 80);

cout << "Enter 2 string: ";

cin.getline(str4, 80);

strcpy(str3, str3);

strcpy(str4, str4);

Str PlusObj = Str();

PlusObj + str3;

PlusObj + str4;

PlusObj.getStr();

}

void testskobki()

{

char\*s = (char\*)calloc(256, 1);

Str str(s);

cout << ":operator() test:" << endl;

cout << "Enter string: ";

cin.ignore();

cin.getline(s, 256);

str.SetString(s);

cout << "Substring (2, 2) = ";

cout << endl << endl;

}

void testminus()

{

char\*s = (char\*)calloc(256, 1);

Str str(s);

cout << ":operator- test:" << endl;

cout << "Enter string: ";

cin.ignore();

cin.getline(s, 256);

str.SetString(s);

cout << "Enter string to delete: ";

cin.getline(s, 256);

cout << "Result: ";

str = str - s;

}

void testravno()

{

char\*s = (char\*)calloc(256, 1);

Str str(s);

cout << "Enter first string: ";

cin.ignore();

cin.getline(s, 256);

str.SetString(s);

cout << "Enter second string: ";

cin.getline(s, 256);

str = Str(s);

}

int main()

{

Interface items[] =

{

{1, &testconstructor, "Test constructor" },

{2, &testplusstr, "Test overload + strings"},

{3, &testplusobj , "Test overload + objects"},

{4, &testminus, "Test overload - substring"},

{5, &testskobki, "Test overload ()"},

{6, &testravno, "Test overload ="},

};

while (true)

{

int choice;

for each (Interface i in items)

cout << i.id << ". " << i.comment << "\n";

cout << "Choose menu item: ";

cin >> choice;

if (choice < 1 || choice > sizeof(items) / sizeof(Interface))

return 0;

items[choice - 1].invoke();

\_getch();

system("cls");

}

\_getch();

return 0;

}

Приложение 4. Заголовочный файл Interface.h, класс демонстрационного меню

#pragma once

class Interface

{

public:

int id;

void(\*invoke)();

char\* comment;

};