МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

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 №15

по дисциплине: Основы программирования
тема: «Создание библиотеки для обработки строк»

Выполнил: студент группы ПВ-223
Мелехов Артём Дмитриевич

Проверили:
ст. преп. Притчин Иван Сергеевич
асс. Черников Сергей Викторович

Белгород 2023 г.

Лабораторная работа №15 «Создание библиотеки для обработки строк»

Цель работы: получение навыков работы со строками в стиле C.

Содержание отчета:

Тема лабораторной работы

Цель лабораторной работы

Решения задач. Для каждой задачи указаны:

– Текст задачи.

– Исходный код (в том числе и тестов).

Вывод.

Список всех импортируемых библиотек:

```
#include <memory.h>
#include <ctype.h>
```

Задача №3.

Реализуем функцию `strlen`. Она возвращает количество символов в строке (не считая ноль-символ)

Код программы:

```
// Возвращает длину строки
long long strlen(char* begin)
{
	char *end = begin;
	while (*end != '\0')
		end++;

	return end - begin;
}
```

Задача №4.

Реализуем функции поиска:

(a) `char* find(char* begin, char* end, int ch)` – возвращает указатель на первый элемент с кодом `ch`, расположенным на ленте памяти между адресами `begin` и `end` не включая `end`. Если символ не найден, возвращается значение `end`.
(b) `char* find_non_space(char* begin)` – возвращает указатель на первый символ, отличный от пробельных, расположенный на ленте памяти, начиная с `begin` и заканчивая ноль-символом. Если символ не найден, возвращается адрес первого ноль-символа.
(c) `char* find_space(char* begin)` – возвращает указатель на первый пробельный символ, расположенный на ленте памяти начиная с адреса `begin` или на первый ноль-символ
(d) `char* find_non_space_reverse(char* rbegin, const char* rend)` – возвращает указатель на первый справа символ, отличный от пробельных, расположенный на ленте памяти, начиная с `rbegin` (последний символ строки, за которым следует ноль-символ) и заканчивая `rend` (адрес символа перед началом строки). Если символ не найден, возвращается адрес `rend`.
(e) `char* find_space_reverse(char* rbegin, const char* rend)` – возвращает указатель на первый пробельный символ справа, расположенный на ленте памяти, начиная с `rbegin` и заканчивая `rend`. Если символ не найден, возвращается адрес `rend`.
(f)

Код программы:

```
// Возвращает указатель на первый элемент с кодом ch расположенным на ленте памяти
между адресами
// begin и end не включая end
char* find(char* begin, char* end, int ch)
{
	while (begin != end && *begin != ch)
		begin++;

	return begin;
}
```

```
// Возвращает указатель на первый символ, отличный от пробельных, расположенный на ленте памяти,
// начиная с begin и заканчивая ноль-символом. Если символ не найден, возвращается адрес первого
// ноль-символа
char* find_non_space(char* begin)
{
	while (*begin != '\0' && isspace(*begin))
		begin++;

	return begin;
}

// Возвращает указатель на первый пробельный символ, расположенный на ленте памяти начиная с адреса
// begin или на первый ноль-символ
char* find_space(char* begin)
{
	while (*begin != '\0' && isspace(*begin))
		begin++;

	return begin;
}

// Возвращает указатель на первый справа символ, отличный от пробельных, расположенный на ленте
// памяти, начиная с rbegin (последний символ строки, за которым следует ноль-символ) и заканчивая
// rend (адрес символа перед началом строки). Если символ не найден, возвращается адрес rend
char* find_non_space_reverse(char* rbegin, const char* rend)
{
	while (rbegin != rend && isspace(*rbegin))
		rbegin--;

	return rbegin;
}

// Возвращает указатель на первый пробельный символ справа, расположенный на ленте памяти, начиная с
// rbegin и заканчивая rend. Если символ не найден, возвращается адрес rend
char* find_space_reverse(char* rbegin, const char* rend)
{
	while (rbegin != rend && !isspace(*rbegin))
		rbegin--;

	return rbegin;
}
```

Задача №5.

Опишем функцию, которая часто используется для проверки строк на равенство. Функция возвращает отрицательное значение, если `lhs` располагается до `rhs` в лексикографическом порядке (как в словаре), значение 0, если `lhs` и `rhs` равны, иначе – положительное значение.

Код программы:

```
// Сравнивает 2 строки. Возвращает отрицательное значение, если lhs располагается до rhs в
// лексикографическом порядке, значение 0, если lhs и rhs равны, иначе – положительное значение
int strcmp(const char *lhs, const char *rhs)
{
	while (*lhs && (*lhs == *rhs))
		lhs++, rhs++;

	return (*lhs - *rhs);
}
```

Задача №6.

Функции для копирования:

(a) `char* copy(const char* begin_source, const char* end_source, char* begin_destination)` – записывает по адресу `begin_destination` фрагмент памяти, начиная с адреса `begin_source` до `end_source`. Возвращает указатель на следующий свободный фрагмент памяти в `begin_destination`.
По окончании работы функции ноль-символ не записывается.

(b) `char* copy_if(char* begin_source, const char* end_source, char* begin_destination, long long (*f)(long long))` – записывает по адресу `begin_destination` элементы из фрагмента памяти начиная с `begin_source` заканчивая `end_source`, удовлетворяющие функции-предикату `f`. Функция возвращает указатель на следующий свободный для записи фрагмент в памяти.
По окончании работы функции ноль-символ не записывается.

(c) `char* copy_if_reverse(char* rbegin_source, const char* rend_source, char* begin_destination, long long (*f)(long long))` – записывает по адресу `begin_destination` элементы из фрагмента памяти начиная с `rbegin_source` заканчивая `rend_source`, удовлетворяющие функции-предикату `f`. Функция возвращает значение `begin_destination` по окончании работы функции.
По окончании работы функции ноль-символ не записывается.

Код программы:

```
// Записывает по адресу begin_destination фрагмент памяти, начиная с адреса begin_source до
// end_source. Возвращает указатель на следующий свободный фрагмент памяти в destination
char* copy(const char* begin_source, const char* end_source, char* begin_destination)
{
	memcpy(begin_destination, begin_source, end_source - begin_source);

	return begin_destination;
}
```

```
// Записывает по адресу begin_destination элементы из фрагмента памяти начиная с
begin_source
// заканчивая end_source, удовлетворяющие функции-предикату f. Функция возвращает
указатель на
// следующий свободный для записи фрагмент в памяти
char* copy_if(char* begin_source, const char* end_source, char* begin_destination,
	long long (*f)(long long))
{
	while (begin_source != end_source)
	{
		if (!f(*begin_source))
			*begin_destination++ = *begin_source;

		*begin_source++;
	}

	return begin_destination;
}

// Записывает по адресу begin_destination элементы из фрагмента памяти начиная с
rbegin_source
// заканчивая rend_source, удовлетворяющие функции-предикату f. Функция возвращает
значение
// begin_destination по окончании работы функции
char* copy_if_reverse(char* rbegin_source, const char* rend_source, char*
begin_destination,
	long long (*f)(long long))
{
	while (rbegin_source != rend_source)
	{
		if (!f(*rbegin_source))
			*begin_destination++ = *rbegin_source;

		*rbegin_source--;
	}

	return begin_destination;
}
```

Вывод: в ходе выполнения работы были получены навыки работы со строками в стиле C.