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 № 17**

по дисциплине: Основы программирования

тема: «Создание библиотеки для обработки строк»

Выполнил: ст.группы

Игнатьев Артур Олегович

Проверил:

Преподаватель Притчин Иван Сергеевич

Преподаватель Черников Сергей Викторович

Белгород 2022г.

**Лабораторная работа «Создание библиотеки для обработки строк»**

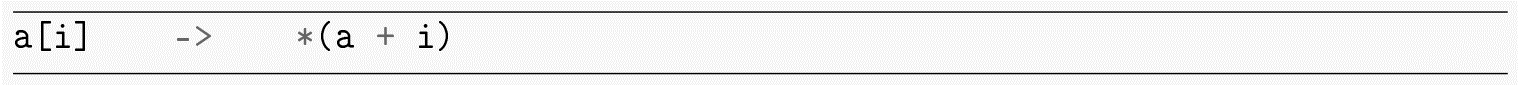
**Цель работы:** получение навыков работы со строками в стиле С.

**Содержание отчёта:**

* Тема лабораторной работы.
* Цель лабораторной работы.
* Исходный код string\_.h / string\_.c.
* Ссылка на открытый репозиторий с решением.

**Требования:**

* Запрещено использование string.h.
* Запрещены операции обращения к элементу по индексу, а также замена:



**Задания к лабораторной работе:**

**Исходный код string\_.h**

#ifndef LABS\_STRING\_H  
#define LABS\_STRING\_H  
  
#include <ctype.h>  
#include <memory.h>  
  
//возвращает количество символов в строке (не считая ноль-символ)  
size\_t strlen\_(const char \*begin);  
  
// 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  
char \*find(char \*begin, char \*end, int ch);  
  
//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 \*findNonSpace(char \*begin);  
  
//возвращает указатель на первый пробельный символ, расположенный на ленте  
// памяти начиная с адреса begin или на первый ноль-символ  
char \*findSpace(char \*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 \*findNonSpaceReverse(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  
char \*findSpaceReverse(char \*rbegin, const char \*rend);  
  
//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  
int strcmp(const char \*lhs, const char \*rhs);  
  
//записывает по адресу beginDestination  
//фрагмент памяти, начиная с адреса beginSource до endSource  
//Возвращает указатель на следующий свободный фрагмент памяти в destination  
char \*copy(const char \*beginSource, const char \*endSource,  
 char \*beginDestination);  
  
//записывает по адресу beginDestination элементы из фрагмента памяти начиная с beginSource  
//заканчивая endSource, удовлетворяющие функции-предикату f. Функция  
//возвращает указатель на следующий свободный для записи фрагмент в памяти  
char \*copyIf(char \*beginSource, const char \*endSource,  
 char \*beginDestination, int (\*f)(int));  
  
//записывает по адресу beginDestination элементы из фрагмента памяти начиная с rbeginSource  
//заканчивая rendSource, удовлетворяющие функции-предикату f. Функция возвращает значение  
//beginDestination по окончанию работы функции  
char \*copyIfReverse(char \*rbeginSource, const char \*rendSource,  
 char \*beginDestination, int (\*f)(int));  
  
#endif //LABS\_STRING\_H

**Исходный код string\_.c**

3. Реализуем функцию strlen.

size\_t strlen\_(const char \*begin) {  
 char \*end = begin;  
 while (\*end != '\0')  
 end++;  
  
 return end - begin;  
}

4. Реализуем функции поиска:

(a) char\* find(char \*begin, char \*end, int ch)

char \*find(char \*begin, char \*end, int ch) {  
 while (begin != end && \*begin != ch)  
 begin++;  
  
 return begin;  
}

(b) char\* findNonSpace(char \*begin)

char \*findNonSpace(char \*begin) {  
 while (\*begin != '\0' && isspace(\*begin))  
 begin++;  
  
 return begin;  
}

(c) char\* findSpace(char \*begin)

char \*findSpace(char \*begin) {  
 while (\*begin != '\0' && !isspace(\*begin))  
 begin++;  
  
 return begin;  
}

(d) char\* findNonSpaceReverse(char \*rbegin, const char \*rend)

char \*findNonSpaceReverse(char \*rbegin, const char \*rend) {  
 while (rbegin > rend && isspace(\*rbegin))  
 rbegin--;  
  
 return rbegin;  
}

(e) char\* findSpaceReverse(char \*rbegin, const char \*rend)

char \*findSpaceReverse(char \*rbegin, const char \*rend) {  
 while (rbegin > rend && !isspace(\*rbegin))  
 rbegin--;  
  
 return rbegin;  
}

5. Опишем функцию strcmp.

int strcmp(const char \*lhs, const char \*rhs) {  
 while (\*lhs == \*rhs && \*lhs != '\0') {  
 lhs++;  
 rhs++;  
 }  
  
 return \*lhs - \*rhs;  
}

Функции для копирования:

* char\* copy(const char \*beginSource, const char \*endSource, char \*beginDestination)

char \*copy(const char \*beginSource, const char \*endSource,  
 char \*beginDestination) {  
 memcpy(beginDestination, beginSource,  
 sizeof(char) \* (endSource - beginSource));  
  
 return beginDestination + (endSource - beginSource);  
}

* char\* copyIf(char \*beginSource, const char \*endSource, char \*beginDestination, int (\*f)(int))

char \*copyIf(char \*beginSource, const char \*endSource,  
 char \*beginDestination, int (\*f)(int)) {  
 while (endSource > beginSource) {  
 if (f(\*beginSource))  
 \*beginDestination++ = \*beginSource;  
  
 beginSource++;  
 }  
  
 return beginDestination;  
}

* char\* copyIfReverse(char \*rbeginSource, const char \*rendSource, char \*beginDestination, int (\*f)(int))

char \*copyIfReverse(char \*rbeginSource, const char \*rendSource,  
 char \*beginDestination, int (\*f)(int)) {  
 while (rbeginSource > rendSource) {  
 if (f(\*rbeginSource))  
 \*beginDestination++ = \*rbeginSource;  
 rbeginSource--;  
 }  
  
 return beginDestination;  
}

Ссылка на репозиторий с библиотекой string:

https://github.com/NTK-Hub/Labs/tree/master/libs/data\_structures/string