Лабораторная №10. Массивы символов. Строки null-terminated.

Темы, рассматриваемые в лабораторной работе:

Использование адресной арифметики при работе с массивами символов, представляющими null-terminated строки. Базовые алгоритмы для строк.

В задачах использовать строки статического размера с максимальной длиной 256 символов.

```
const int Max_Length = 256;
char str[Max_Length];
```

Пример функции копирования одной строки в другую

```
char * copy(char *p, const char *q) {
    char *pp=p;
    while (*pp++=*q++);
    return p;
}
```

Задания:

Реализовать следующие функции. Функция должна использовать адресную арифметику для работы со строками, завершение строки должно определяться символом ‘\0’ Написать для каждой функции набор тестов.

1. Функция сравнения строк.

```
// Входные параметры:
//     str1 - начало первой строки;    str2 - начало второй строки.
// Возвращаемое значение:
//      0 - строки равны;
//     >0 - первый различный символ строки 1 > строки 2;
//     <0 - первый различный символ строки 1 < строки 2.
int str_cmp(const char *str1, const char *str2);
```

2. Функция вычисления количества слов в строке.

```
// Входные параметры:
//     str - начало строки.
// Слово - последовательность не пробельных символов, разделенных пробелами.
// Возвращаемое значение:
// количество слов
int word_count(const char* str);
```

3. Поиск первого вхождения заданного символа в строке.

```
// Входные параметры:
//     str – начало строки;     c – искомый символ.
// Результат:
//     указатель на позицию искомого символа в строке,
//     nullptr, если символ не найден.
const char* str_chr(const char *str, char c);
```

4. Обрезать строку.

```
// Входные параметры:
//     str – начало строки;  c – на котором строка должна оканчиваться.
// Результат:
//     str не содержит символы, расположенные после символа  c
//     длина строки меньше или равна исходной длины
// Если в строке нет символа c, функция выбрасывает исключение
void trim_chr(char *str, char c);
```

5. Вставка заданного символа в конец строки.

```
// Входные параметры:
// str – начало строки; c – символ.
// Результат:
//     str содержит символ c в конце, длина str больше на 1.
void insert_str(char *str, char c);
```

В задачах, требующих изменение размера строки, как и в задачах с другими массивами, можно использовать динамическую память. При этом следует учитывать, что размер памяти, выделяемой под строку, должен быть на 1 больше, чем количество символов в ней.

Пример функции, дублирующей каждый символ в строке, и программы, использующей эту функцию.

```
#include <cstring>
#include <iostream>

using namespace std;

void repeat_chars(char* &str) {
	int len = strlen(str);
	if (len != 0) {
		char* new_str = new char[len*2 + 1];
		char* c = str;
		char* p = new_str;
		do {
			*p = *c;
			p++;
			*p = *c;
			p++;
			c++;
		} while (*c!='\0');
		*p = '\0';
		delete[] str;
		str = new_str;
	}
}

int main() {
	char* s = new char[5];
	copy(s, "1234");
	cout << s << endl;
	repeat_chars(s);
	cout << s << endl;
	delete[]s;
	return 0;
}
```

Следующие функции реализовать для случая использования динамических строк.

6. Удаление всех пробелов в строке.

```
// Входные параметры:
//     str – начало строки;
// Результат:
//     str не содержит пробелов.
void remove_space(char *&str);
```

7. Вставка заданной подстроки в строку после первого вхождения заданного символа.

```
// Входные параметры:
// str – начало исходной строки; substr – строка для вставки; c – символ.
// Результат:
//     str содержит substr после первого вхождения c.
//     Если вставка не возможна, выбрасывает исключение char*.
void insert_str(char *&str, const char *substr, char c);
```