**Белорусский государственный технологический университет**

**Факультет информационных технологий**

**Кафедра ПИ**

Лабораторная работа № 13

По дисциплине «Основы алгоритмизации и программирования»

На тему «Обработка символьной информации»

Выполнил:  
Студент 1 курса, 10 группа  
Короткевич Артём Сергеевич  
Преподаватель: асс. Харланович А.В

2024, Минск  
  
  
  
  
  
  
**Задание 1**Изучить способы преобразования символов, выполнив программы в правой части.  
Выполнить прокрутки программ и записать условия.  
Опробовать программы с различным текстом.  
  
**Код(1)**

#include <iostream>  
void main() {

int num = 5;

char symb, new\_symbol = ' ';

char\* pc;

pc = &symb;

\*pc = num + '0';

std::cout << \*pc << ' ';

if (symb >= '0' && symb <= '9')

num = symb - '0';

std::cout << num << ' ';

symb = 'b';

if (symb >= 'a' && symb <= 'z')

new\_symbol = symb - 'a' + 'A';

std::cout << new\_symbol << ' ';  
}

**Прокрутка(1)**  
  
1. Инициализация: num = 5;   
symb — неинициализированная переменная.  
 new\_symbol = ' '; (пробел)   
 pc — указатель на char, инициализированный нулевым значением.   
2. pc = &symb  
Указатель pc теперь указывает на переменную symb.   
3. \*pc = num + '0'  
 Вычисляется num + '0' (5 + 48 = 53).   
 48 — ASCII-код символа '0'.   
 Значение 53 (ASCII-код символа '5') записывается по адресу, на который указывает pc (в переменную symb).   
Теперь symb = '5'.   
4. cout << \*pc << ' ';   
 Выводится значение, на которое указывает pc (символ '5'), затем пробел.   
5. if (symb >= '0' && symb <= '9')   
num = symb - '0';   
 Условие истинно, так как symb = '5'.  
 symb - '0' (53 - 48 = 5) вычисляется.   
Значение 5 присваивается переменной num.   
Теперь num = 5.   
6. cout << num << ' ';   
 Выводится значение num (5), затем пробел.  
 7. symb = 'b';   
Значение 'b' присваивается переменной symb.  
 8. if (symb >= 'a' && symb <= 'z') new\_symbol = symb - 'a' + 'A';   
Условие истинно, так как symb = 'b'.   
 symb - 'a' + 'A' (98 - 97 + 65 = 66) вычисляется. 98, 97, и 65 - это ASCII-коды символов 'b', 'a', и 'A' соответственно.   
Значение 66 (ASCII-код символа 'B') присваивается переменной new\_symbol. Теперь new\_symbol = 'B'.   
9. cout << new\_symbol << ' ';  
Результат программы: Вывод программы: 5 5 B  
  
**Результат(1)**  
  
  
  
**Код(2)**

#include <iostream>  
void main()

{

char str[] = "Text";

int count = 0;

char\* pstr;

pstr = &str[0];

if (str)

while (\*pstr++)

++count;

std::cout << count;  
}

**Прокрутка(2)**  
  
Инициализация:   
str содержит "Text\0" (символ '\0' - нулевой символ, конец строки).   
 count = 0.  
 pstr указывает на начало строки str (на 'T').   
Цикл while:  
 Цикл выполняется до тех пор, пока \*pstr не равно '\0'.  
 Итерации:   
 Итерация 1: \*pstr ('T') не равно '\0', count увеличивается до 1, pstr сдвигается на следующий символ.   
 Итерация 2: \*pstr ('e') не равно '\0', count увеличивается до 2, pstr сдвигается. Итерация 3: \*pstr ('x') не равно '\0', count увеличивается до 3, pstr сдвигается. Итерация 4: \*pstr ('t') не равно '\0', count увеличивается до 4, pstr сдвигается. Итерация 5: \*pstr ('\0') равно '\0', цикл завершается.   
Вывод:  
cout << count; выводит значение count, которое равно 4.   
Результат: Программа выведет 4.  
 **Результат(2)  
  
**  
 **Задание 3**Изучить способы работы с символьными массивами, выполнив программу в правой части.  
  
**Код**

#include <iostream  
#include <cstring>

using namespace std;

void main()

{

char s, t[20];

int i, j, n;

cout << "Input string: ";

gets\_s(t);

cout << "Input symbol: ";

cin >> s;

n = strlen(t);

for (i = 0; i <= n; i++)

if (t[i] == s)

for (j = i; j <= n; j++)

t[i] = t[i + 1];

for (i = 0; i < n; i++)

cout << t[i];

}

**Результат  
  
**  
**Задание 4**  
Выполнить программу в правой части.  
Закомментировать строку **cin.ignore…**Объяснить возникшие проблемы при выполнении программы. **Код**

#include <iostream>  
#include <cstring>

using namespace std;

void main()

{

char t[3][20];

int i, n;

cout << "Input size (<=3): ";

cin >> n;

cin.ignore(cin.rdbuf()->in\_avail()); //очистка входного буфера

for (i = 0; i < n; i++)

{

cout << "Input string: ";

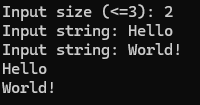
gets\_s(t[i]);

}

for (i = 0; i < n; i++)

puts(t[i]);

}

**Результат  
  
  
  
Комментарий  
  
cin.ignore(cin.rdbuf()->in\_avail())** удаляет все оставшиеся символы из буфера ввода, чтобы следующий ввод **gets\_s()** выполнялся корректно  
  
  
**Задание 5**В правой части записаны два варианта решения за-дачи. Выполнить программы и объяснить различия между ними. Внести изменения в программу с тем, чтобы проверялось не только количество скобок, но и правильность их расстановки (первой в тексте должна быть открывающая скобка).  
  
**Код(1)**

#include <stdio.h>  
void main()

{

char s[256];

int i, count;

puts("Enter string: ");

gets\_s(s);

for (count = i = 0; s[i] != 0; i++)

{

if (s[i] == '(') count++;

if (s[i] == ')') count--;

}

if (!count)

puts("Ok\n");

else

puts("Not Ok\n");

}

**Результат(1)**  
  
  
  
**Код(2)**

#include <stdio.h>  
void main()

{

char s[256];

int count;

char\* ps;

puts("Enter string: ");

gets\_s(s);

for (count = 0, ps = s; \*ps != 0; ps++)

{

if (\*ps == '(') count++;

if (\*ps == ')') count--;

}

if (count == 0)

puts("Ok\n");

else

puts("Not Ok\n");

}

**Результат(2)**  
  
  
  
**Комментарий**  
Оба кода выполняют одну и ту же задачу: проверяют, равны ли количества открывающих и закрывающих круглых скобок в строке.  
Разница в реализации решений.  
1) С использованием индекса i  
2) С использованием указателя ps **Исправленный код(1)**

#include <stdio.h>  
void main()

{

char s[256];

int i, count;

puts("Enter string: ");

gets\_s(s);

for (count = i = 0; s[i] != 0; i++)

{

if (s[i] == '(')

count++;

if (s[i] == ')')

count--;

if (count < 0) // если закрывающая скобка встречается раньше открывающей

{

puts("Not Ok\n");

return;

}

}

if (count == 0)

puts("Ok\n");

else

puts("Not Ok\n");

}

**Исправленный код(2)**

#include <stdio.h>  
void main()

{

char s[256];

int count;

char\* ps;

puts("Enter string: ");

gets\_s(s);

for (count = 0, ps = s; \*ps != 0; ps++)

{

if (\*ps == '(')

count++;

if (\*ps == ')')

count--;

if (count < 0) // если закрывающая скобка встречается раньше открывающей

{

puts("Not Ok\n");

return;

}

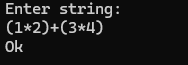
}

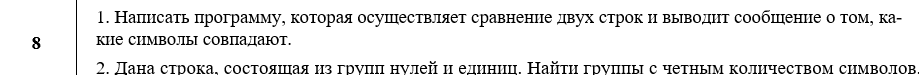
if (count == 0)

puts("Ok\n");

else

puts("Not Ok\n");  
}

**Результаты** ****  
  
**Задание 6**  
Выполнить задания из таблицы ниже, используя ***индексы*** для доступа к элементам массивов в первой программе и ***указатели*** − во второй программе.  
При написании программне использовать стандартные функции для строк символов.

**Вариант 8**  
  
 **  
  
  
Код(1)**=

#include <iostream>  
#include <string>

using namespace std;

int main() {

setlocale(LC\_ALL, "Russian");

string strok1, strok2, commonCharacter; // strok1 и strok2 — вводимые строки, commonCharcheter — строка для хранения совпадающих символов

cout << "Введите первую строку: ";

cin >> strok1;

cout << "Введите вторую строку: ";

cin >> strok2;

for (size\_t i = 0; i < strok1.length(); ++i) { // цикл по каждому символу из первой строки

bool alreadyAdded = false; // флаг для проверки, был ли символ уже добавлен в commonCharacter

for (size\_t j = 0; j < commonCharacter.length(); ++j) { // проверяем, добавлен ли уже символ в commonCharacters

if (strok1[i] == commonCharacter[j]) {

alreadyAdded = true; // символ уже добавлен, пропускаем его

break;

}

}

if (alreadyAdded) continue; // если символ уже добавлен, переходим к следующему символу

for (size\_t k = 0; k < strok2.length(); ++k) { // проверяем, есть ли символ str1[i] во второй строке

if (strok1[i] == strok2[k]) {

commonCharacter += strok1[i]; // если символ найден в обеих строках, добавляем его в commonCharacter

break;

}

}

}

if (commonCharacter.empty()) { // если совпадающих символов нет

cout << "Совпадающих символов нет." << endl;

}

else {

cout << "Совпадающие символы: ";

for (size\_t i = 0; i < commonCharacter.length(); ++i) {

cout << commonCharacter[i] << " ";

}

cout << endl;

}

return 0;  
}

**Результат(1)  
  
  
  
Код(2)**

#include <iostream>  
#include <string>

using namespace std;

int main() {

setlocale(LC\_ALL, "Russian");

string binarString, currentGroup; // строка для ввода и временная строка для текущей группы

bool hasGroups = false; // флаг, показывающий, найдены ли группы с четным количеством символов

cout << "Введите строку из нулей и единиц: ";

cin >> binarString; // ввод строки из нулей и единиц

cout << "Группы с четным количеством символов: " << endl; // ввод заголовка для групп с четным количеством символов

for (size\_t i = 0; i < binarString.length(); ++i) { // проходим по каждому символу строки

if (binarString[i] == '0' || binarString[i] == '1') { // если символ — это '0' или '1', добавляем его в текущую группу

currentGroup += binarString[i];

}

else {

if (!currentGroup.empty() && currentGroup.length() % 2 == 0) { // если встретился символ, не являющийся '0' или '1', обрабатываем текущую группу

cout << currentGroup << endl;

hasGroups = true; // устанавливаем флаг, что хотя бы одна группа найдена

}

currentGroup.clear(); // очищаем текущую группу

}

}

if (!currentGroup.empty() && currentGroup.length() % 2 == 0) { // проверяем последнюю группу

cout << currentGroup << endl;

hasGroups = true; // устанавливаем флаг, что хотя бы одна группа найдена

}

if (!hasGroups) // если не было найдено ни одной группы, сообщаем

{

cout << "Нет групп с четным количеством символов" << endl;

}

return 0;  
}

**Результат(2)  
  
  
  
  
  
  
  
  
  
Дополнительные задания**



**Код**

#include <iostream>

#include <string> // подключение библиотеки для хранения текстовых данных

using namespace std;

int main()

{

setlocale(LC\_ALL, "Russian");

string str; // объявляем строковую переменную

char a;

int n;

cout << "Ввести количество слов: " << endl;

cin >> n;

cout << "Ввести окончание слов: " << endl;

cin >> a;

for (int i = 0; i < n; i++) // создаем цикл, который будет проверять наши слова в строке в зависимости от их количества

{

cout << "Введите слово: ";

cin >> str;

if (str[str.size() - 1] == a) // если окончание введеного слова равно тому, которое ввели мы, то выводим:

{

cout << "В этом слове присутствует введенное окончание" << endl;

}

else

{

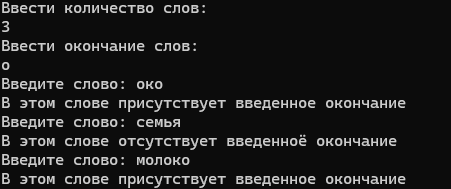
cout << "В этом слове отсутствует введенноё окончание" << endl;

}

}

return 0;  
}

**Результат**





**Код**

#include <iostream>

#include <string> // подключение библиотеки для работы со строками

#include <algorithm> // подключение библиотеки, которая будет искать анаграммы и будет сортировать слова

using namespace std;

int main()

{

setlocale(LC\_ALL, "Russian");

string words[] = { "керамит", "материк", "метрика", "равновесие", "своенравие" }; // массив из слов

int n = 5; // кол-во слов в массиве

for (int i = 0; i < n; i++) // цикл для сортировки слов

{

string firstword = words[i]; // выбор первого слова для сравнения

sort(firstword.begin(), firstword.end()); // сортировка каждой буквы слова с начала и до конца

for (int j = i + 1; j < n; j++) // второй цикл для сравнения с другими словами

{

string secondword = words[j]; // выбор второго слова для сравнения

sort(secondword.begin(), secondword.end()); // сортировка каждой буквы для сравнения между вторым и первым словом

if (firstword == secondword) // если отсортированные слова совпадают, значит они анаграммы

{

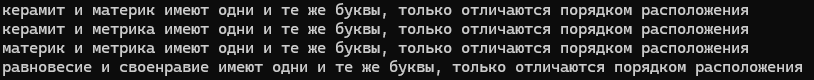
cout << words[i] << " и " << words[j] << " имеют одни и те же буквы, только отличаются порядком расположения" << endl;

}

}

}

return 0;  
}

**Результат  
  
**

**Код**

#include <iostream>

#include <string> // подключаем библиотеку для работы со строками

using namespace std;

int main()

{

setlocale(LC\_ALL, "Russian");

string sentence = "Сингапур – многонациональный многонациональный мировой мировой финансовый финансовый центр центр"; // строка с нашим предложением

string duplicateWords[15]; // объявляем переменную в которой может хранится 15 слов дубликатов

string word; // строка для хранения слова

int count = 0; // счетчик слов

bool duplicate = true; // флаг, который будет определять, является ли слово дубликатом или нет

for (int i = 0; i < sentence.length(); i++) // цикл, который проходит по всему нашему предложению

{

if (sentence[i] != ' ') // разбиваем предложение на слова с помощью пробела и сравниваем слова

{

word += sentence[i];

}

else {

for (int j = 0; j < count; j++) // если слово есть в массиве слов-дубликатов, то устанавливаем флаг false

{

if (duplicateWords[j] == word) {

duplicate = false;

break; // цикл прерван

}

}

if (duplicate) // если слово не является дубликатом, значит добавим его в массив слов-дубликатов

{

duplicateWords[count] = word;

count++; // увеличиваем счетчик на 1

cout << word << " "; // выводим каждое слово

}

word = ""; // очистка нынешнего слова

duplicate = true; // сброс флаг(дубликата)

}

}

return 0;  
}

**Результат  
  
**