Семинар #3: Строки. Домашнее задание.

Задача 1. Коды и символы

Что напечатает данная программа?

```c
#include <stdio.h>
int main()
{
    printf("%i\n", '>');
    printf("%i\n", '1');
    printf("%i\n", '\0');
    printf("%i\n", '\n');
    printf("%i\n", '4' * '2');
    printf("%i\n", '7' - '0');
    printf("%c\n", 't' - 32);
    printf("%c\n", 'X' + 'a' - 'A');
}
```

Для сдачи задачи создайте в текстовом редакторе файл `01.txt`, запишите в нем все ответы, а затем добавьте файл в ваш github-репозиторий.

Задача 2. Печать всех символов

Напишите программу, которая будет печатать на экран все символы с кодами от 32 до 126 в формате:

```
Symbol = A, Code = 65
```

Задача 3. Тип символа

Напишите программу, которая будет считывать символ и печатать:

- `Letter`, если этот символ – буква (`[A, Z]` и `[a, z]`)
- `Digit`, если этот символ – цифра
- `Other`, если это какой-то другой символ

Решите эту задачу в трёх вариантах:

(a) Без использования строковых литералов и библиотеки `ctype.h`

(b) Используя строковые литералы, но без использования библиотеки `ctype.h`

(c) Используя библиотеку `ctype.h`

Задача 4. Чередование

Считать 2 слова и печатать их чередуя по одному символу. То есть сначала напечатать первый символ первой строки, потом первый символ второй строки, потом второй символ первой строки, второй символ второй и т. д. Если какая-то из строк закончится, то нужно допечатать оставшуюся строку.

вход	выход
cat dog	cdaotg
cat elephant	cealtephant
elephant dog	edloegphant
aaaa bbbb	abababab
aaaa b	abaaa
a b	ab

Задача 5. Сумма цифр

На вход программе поступает число из диапазона от 0 до 10^{10^8}. Найти сумму цифр этого числа

вход	выход
0	0
123	6
999999999999999999999999	225

Задача 6. Палиндром

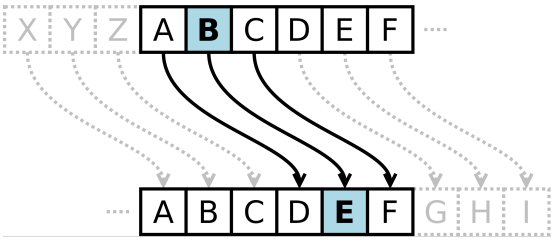
Написать функцию `is_palindrom`, которая будет принимать строку и проверять является ли эта строка палиндромом.

вход	выход
abba	Yes
aba	Yes
a	Yes
aa	Yes
ab	No

вход	выход
abcdedcba	Yes
abcdedcb	No
abcdedcbab	No
abcdedcbb	No
abcxedcba	No

Задача 7. Шифр Цезаря

Шифр Цезаря — это вид шифра подстановки, в котором каждый символ заменяется символом, находящимся на некотором постоянном числе позиций левее или правее него в алфавите.



Напишите функцию `void encrypt(char* str, int k)`, которая будет зашифровывать фразу шифром Цезаря.

вход	выход
1 ABCZ	BCDA
15 ZzZzZ	OoOoO
7 The Fox Jumps Over The Dog	Aol Mve Qbtwz Vcly Aol Kvn
13 Green Terra	Terra Green

Задача 8. Ошибка в считывании

Если в этой программе ввести число, нажать `Enter` и ввести строку, то произойдёт ошибка. Почему? Исправьте эту ошибку.

```c
#include <stdio.h>
int main()
{
    int a;
    scanf("%i", &a);
    printf("a = %i\n", a);

    char str[100];
    scanf("%[^\n]", str);
    printf("str = %s\n", str);
}
```

Для сдачи задачи создайте в текстовом редакторе файл `08.txt`, запишите в нем причину ошибки и способ ее исправления, затем добавьте файл в ваш github-репозиторий.

Задача 9. Укоротить строку

Напишите функцию `void trim_after_first_space(char str[])`, которая будет принимать на строку и укорачивать её до первого пробела. Протестируйте функцию с помощью следующего кода:

```
#include <stdio.h>
// Тут вам нужно написать функцию trim_after_first_space
int main()
{
    char a[] = "Cats and Dogs";
    printf("%s\n", a); // Должно напечатать Cats and Dogs
    trim_after_first_space(a);
    printf("%s\n", a); // Должно напечатать Cats
}
```

Задача 10. Сокровище

Вы находитесь на плоскости в начале координат (точке с координатами `(0, 0)`) и вам нужно найти закопанное сокровище. Путь до него задаётся последовательностью команд, состоящих из направления и расстояния, которое нужно пройти в этом направлении. Вам нужно найти координаты сокровища. Используйте функцию `strcmp`.

вход	выход
6 North 10 East 20 South 50 West 60 East 10 North 60	-30 20

Задача 11. Безопасный `strcpy`

Известно, что функция `strcpy` небезопасна, так как может выйти за границы массива, в который она записывает. Например, в следующем примере:

```
char a[10] = "Mouse";
char b[50] = "LargeElephant";
strcpy(a, b); // UB, выйдет за пределы массива a
```

Ваша задача заключается в том, чтобы написать функцию `safe_strcpy`, которая будет более безопасной, чем функция `strcpy` и не будет выходить за границы массива. Эта функция должна иметь 3 параметра:

- строка в которую мы будем записывать (тип `char[]`)
- размер массива в который мы будем записывать (тип `size_t`)
- строка из которой мы будем считывать (тип `const char[]`).

В случае если строка из которой мы считываем не будет помещаться в массив, нужно скопировать только часть строки, которая может поместиться в него. И обязательно поставить нулевой символ в конце строки.

```
char a[10] = "Mouse";
char b[50] = "LargeElephant";
safe_strcpy(a, 10, b); // OK, строка a будет равна "LargeElep\0"
```

Задача 12. Повторитель

Напишите программу `repeater`, которая будет принимать через аргументы командной строки некоторое слово и некоторое число. Эта программа должна печатать это слово столько раз, чему равно переданное число. Например, если мы вызовем эту программу так:

```
./repeater Hello 5
```

то программа должна напечатать:

```
Hello Hello Hello Hello Hello
```

Если вы программируете под Windows, то исполняемый файл будет называться `repeater.exe`, а не `repeater`.

Задача 13. Простой калькулятор

Напишите программу `calc`, которая будет выполнять базовые арифметические операции над двумя целыми числами. Программа должна принимать строго 3 аргумента командной строки (при этом `argc` будет иметь значение 4) в следующем порядке:

- Первый операнд
- Знак оператора (`+`, `-`, `*`, `/` или `%`)
- Второй операнд

Команда должна поддерживать: сложение, вычитание, умножение, целочисленное деление и нахождение остатка. Программа должна обрабатывать такие ошибки во входных данных, как неверное количество аргументов, неверный оператор, неверный формат операндов и ошибку деления на ноль.
Примеры использования программы:

```
$ ./calc 10 + 20
30

$ ./calc 500 / 9
55

$ ./calc 11 * 12
132

$ ./calc 20 % 7
6

$ ./calc 5
Error: Wrong number of arguments!
Usage: ./calc <number> <operator> <number>

$ ./calc 10 & 20
Error: Invalid operator!

$ ./calc 10.5 + abc
Error: Operands should be integers!

$ ./calc 10 / 0
Error: Division by zero!
```

Если вы программируете под Windows, то исполняемый файл будет называться `calc.exe`, а не `calc`.

Задача 14. Шифрование файла

Напишите программу `encrypt`, которая будет принимать через аргументы командной строки названия входного и выходного файлов, а также число-ключ шифра Цезаря. Программа должна считывать входной файл, зашифровывать его шифром Цезаря и записывать результат в выходной файл.
Например, если мы вызовем эту программу так:

```
./encrypt a.txt b.txt 7
```

то программа должна считывать файл `a.txt`, зашифровывать содержимое шифром Цезаря с ключом 7 и записывать результат в файл `b.txt`. В данной задаче необязательно обрабатывать ошибки во входных данных.
Проверьте работу программы на файлах `three_little_pigs.txt` и `invisible_man.txt`.
Если вы программируете под Windows, то исполняемый файл будет называться `encrypt.exe`, а не `encrypt`.

Задача 15. Извлечение строк

Напишите программу `line_extractor`, которая будет извлекать строки из файла. Программа должна принимать строго 3 аргумента командной строки:

- Имя входного файла
- Имя нового файла, который будет содержать определённые строки
- Или одно число – номер строки, или диапазон строк в формате `начало:конец`

Например, если вызвать программу так:

```
$ ./line_extractor a.txt b.txt 10
```

то она должна создать новый файл `b.txt` и поместить туда 10-ю строку из файла `a.txt`.

Если же вызвать программу так:

```
$ ./line_extractor a.txt b.txt 10:20
```

то программа должна создать новый файл `b.txt` и поместить туда строки из 10-й по 20-ю (не включительно) из файла `a.txt`.

Если аргументы заданы неверно, то программа должна напечатать сообщение о ошибке и завершиться. Программа должна обрабатывать следующие ошибки:

- Неверное количество аргументов
- Начальный файл не существует.
- Неверный формат задания номера строки или диапазона строк.

Примеры ошибочных вызовов программы:

```
$ ./line_extractor result.txt
Error: Wrong number of arguments!
Usage: ./line_extractor <input_file> <output_file> <lines

$ ./line_extractor qwertyuiop.txt b.txt 10
Error: File qwertyuiop.txt does not exist!

$ ./line_extractor a.txt b.txt 10:20abc
Error: Wrong lines format!

$ ./line_extractor a.txt b.txt 10:
Error: Wrong lines format!
```

Проверьте работу программы на файлах `three_little_pigs.txt` и `invisible_man.txt`.
Если вы программируете под Windows, то исполняемый файл будет называться `line_extractor.exe`, а не `line_extractor`.

Необязательные задачи (не входят в ДЗ, никак не учитываются)

Задача 1. Номер буквы

Считайте символ буквы латинского алфавита и напечатайте его номер в алфавите. Если на вход подаётся не буква, то нужно напечатать `Not a letter`.

вход	выход
P	16
b	2
B	2
#	Not a letter

Задача 2. Лесенка

Считать слово и напечатать лесенку из этого числа. Например, для слова `Hello` нужно напечатать лесенку:

вход	выход
Hello	H
	He
	Hel
	Hell
	Hello

Задача 3. Восклицание

На вход подаётся строка. Напечатать эту же строку, но ставя восклицательный знак после каждого слова.

вход	выход
Better late than never	Better! late! than! never!
cat dog elephant	cat! dog! elephant!
a	a!

Задача 4. Правильная скобочная последовательность

На вход подаётся скобочная последовательность(строка, состоящая из символов `'('` и `')'`). Нужно выяснить является ли эта скобочная последовательность допустимой или нет.

вход	выход
(()())	Yes
((()()()))	Yes
(()))()())	No
(((())))	Yes
(((())))	No
((((()))	No

вход	выход
()	Yes
)(	No
))	No
((	No
()()	Yes
(	No
)	No

Задача 5. Удаление символа

Напишите функцию `void delete_chars(char str[], char c)`, которая будет удалять все символы, равные `c` из строки `str`. Постарайтесь сделать эту функцию как можно более эффективной.

str, c	str после вызова delete_chars(str, c)
cat a	ct
elephant e	lphant
aaaa a	
a a	
abababababa a	bbbb

Задача 6. Самое длинное слово

Напишите функцию `int longest_word(const char str[], char result[])`, которая будет искать самое длинное слово в строке `str` и записывать его в строке `result`. Строка должна возвращать длину этого слова. Под словом тут понимается последовательность непробельных символов, ограниченная с двух сторон пробельными символами или границами строки. В случае если есть несколько слов с самой большой длиной в `result` нужно записать первое из них.

`str`	`result` после вызова `longest_word(str, result)`
`cats and dogs`	`cats`
`cat dog elephant mouse`	`elephant`
`cat   dog     elephant mouse`	`elephant`

Задача 7. Переворот слов в файле

Счмтайте файл `input.txt` и переверните все слова из этого файла и запишите результат в файл `output.txt`.

файл `input.txt`	файл `output.txt`
`Cat and Dog`	`taC dna goD`

Протестируйте программу на файле `invisible_man.txt`.

Задача 8. Сортировка аргументов

Напишите программу `sort`, которая будет принимать через аргументы командной строки произвольное число строк и сортировать эти строки лексикографически и печатать их.
Например, если мы вызовем эту программу так:

```
./sort cat elephant mouse axolotl lion
```

то программа должна напечатать:

```
axolotl cat elephant lion mouse
```

Задача 9. *Замена

Напишите программу `replacer`, которая будет принимать через аргументы командной строки названия входного и выходного файлов, а также две строки. Программа должна считывать входной файл, заменять все вхождения первой строки на вторую строку и записывать результат в выходной файл.
Например, если мы вызовем эту программу так:

```
./replacer three_little_pigs.txt result.txt pig elephant
```

то программа должна считывать файл `three_little_pigs.txt`, заменять все подстроки `"pig"` на `"elephant"` и записывать результат в файл `result.txt`. Постарайтесь написать эффективный алгоритм замены. Рассмотите случаи когда подстрока заменяется на более длинную строку и когда подстрока заменяется на более короткую строку.