Семинар #4: Строки. Классные задачи.

Таблица ASCII

Символ	Код	C	K	C	K	C	K	C	K	C	K	C	K	C	K	C	K	C	K
\0	0	&	38	0	48	:	58	D	68	N	78	X	88	b	98	l	108	v	118
\t	9	’	39	1	49	;	59	E	69	O	79	Y	89	c	99	m	109	w	119
\n	10	(	40	2	50	<	60	F	70	P	80	Z	90	d	100	n	110	x	120
		)	41	3	51	=	61	G	71	Q	81	[	91	e	101	o	111	y	121
(пробел)	32	*	42	4	52	>	62	H	72	R	82	\	92	f	102	p	112	z	122
!	33	+	43	5	53	?	63	I	73	S	83	]	93	g	103	q	113	{	123
"	34	,	44	6	54	@	64	J	74	T	84	^	94	h	104	r	114	|	124
#	35	-	45	7	55	A	65	K	75	U	85	_	95	i	105	s	115	}	125
$	36	.	46	8	56	B	66	L	76	V	86	‘	96	j	106	t	116	~	126
%	37	/	47	9	57	C	67	M	77	W	87	a	97	k	107	u	117		

Символы. Модификатор %c:

Тип `char` – это тип целочисленных чисел размером 1 байт (соответственно диапазон от `-128` до `127`).
Часто используется для хранения кодов символов (хотя его можно использовать и для хранения чисел).
Функция `printf` с модификатором %c принимает на вход число и печатает соответствующий символ.

- Спецификатор `%hhd` - печатать и считывание однобайтовых чисел (для печати можно использовать и `%d`).
- Спецификатор `%c` в `printf` - берёт число и печатает символ с соответствующим кодом.
- Спецификатор `%c` в `scanf` - читает 1 символ и записывает его код по переданному адресу.

1. **Символы это числа:** Что напечатает следующая программа? Проверьте себя

```c
#include <stdio.h>
int main()
{
        char x = 64;
        printf("%d\n", x);
        printf("%c\n", x);

        // Символьные константы - символы в одинарных кавычках - это просто числа
        printf("%d\n", '>');
        printf("%d\n", '5');
        printf("%d\n", 'a' + '0');
        printf("%d\n", '4'*'2');
        printf("%d\n", '7' - '0');
        printf("%c\n", 't' - 32);
        printf("%c\n", 'Z' + 'a' - 'A');
}
```

2. **ASCII:** Вывести на экран все символы таблицы ASCII с номерами от 32 до 126 в следующем формате:

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

3. **Считывание символов:** Напишите программу, которая будет считывать символы один за другим в цикле `while` и печатать код каждого символа. Используйте `scanf` с модификатором `%c`. Программа должна заканчиваться после ввода символа `q`.

Строки. Модификатор %s:

Строки - это массивы чисел типа `char`. Обычные массивы нельзя печатать одной командой `printf`, но специально для строк ввели модификатор `%s`, благодаря которому можно печатать и считывать строки одной командой.

```
int main()
{
        char a[20] = {77, 73, 80, 84, 0};
        char b[20] = {'M', 'I', 'P', 'T', '\0'};
        char c[20] = "MIPT"; // Лучше всего использовать эту инициализацию
        // Использовать = со строками можно только при создании, то есть такое:
        a = "FAKI";  // работать не будет

        printf("%s\n", a);
        printf("%s\n", b);
        printf("%s\n", c);
}
```

- **Инициализация строки:** Объявите и инициализируйте строку `str` с содержимым `"Hello 1!"` 3-мя разными способами и напечатайте её.
- **Изменение строки:** Измените созданную строку из прошлой задачи на `"Hello 2!"` с помощью команды `str[6] = '2'` и снова напечатайте её.
- **Считывание строки:** Считайте число `n` и строку `str` и напечатайте её `n` раз через пробел.
- **Удвоение:** Считайте строку и напечатайте её удвоив каждый символ. Для итерации используйте тот факт, что в конце строки всегда должен стоять нулевой символ (символ с кодом `0`).

вход	выход
Hello	HHeellllloo
MIPT	MMIIPPTT

- **Длина строки:** Напишите функцию `int get_length(char* str)`, которая будет возвращать длину строки. Стандартную функцию `strlen` в этой задаче использовать нельзя. Проверьте эту функцию в `main()`.
- **Переворот:** Напишите функцию `void reverse_string(char* str)`, которая будет переворачивать строку строку. Проверьте эту функцию в `main()`. Так как строка - это просто массив из чисел, то она передаётся в функцию по указателю (также как и обычный массив) и, соответственно, может меняться внутри.

вход	выход
Hello!	!olleH
live	evil
Madam	madaM

- **UPPERCASE:** Напишите функцию `void to_upper_case(char* str)`, которая будет переводить строку в верхний регистр. Проверьте эту функцию в `main()`.

вход	выход
mipt	MIPT
Hello!	HELLO!
Area51	AREA51

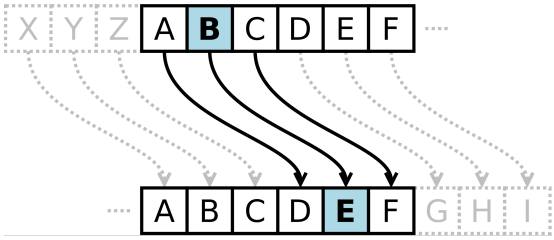
- **Усечение строки:** На вход подаётся строка. Усечь строку до первого символа точка ‘‘.’’. Можно использовать только один вызов функции `printf`.

вход	выход
judge.mipt.ru	judge
A.B.C.	A
.com	

Как вы могли заметить, использование scanf с модификатором `%s` считывает до первого пробельного символа. Чтобы считать всю строку (то есть до символа '\n'), следует использовать модификатор `%[^\n]`. Пример программы, которая считывает строку и меняет пробелы на переносы строк:

```
#include <stdio.h>
int main()
{
        char str[100];
        scanf("%[^\n]", str);
        for (int i = 0; str[i]; i++)
        {
                if (str[i] == ' ')
                        str[i] = '\n';
        }
        printf("%s\n", str);
}
```

- **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

- **Переворот слов:** Используйте решение задачи **Переворот**, чтобы перевернуть каждое слово в строке.

вход	выход
The Fox Jumps Over The Dog	ehT xoF spmuJ revO ehT goD

- **Сортировка символов:** Отсортируйте символы строки по их коду ASCII.

вход	выход
MIPT	IMPT
Majestic12	12Maceijst
The Fox Jumps Over The Dog	DFJOTTeeeghhmooprsuvx

- **Умножение на 3:** На вход передаётся целое положительное число $n < 10^{10000}$. Нужно напечатать это число, умноженное на 3.

вход	выход
123456789098765432123456789098765432 1	370370367296296296370370367296296296 3

Стандартные функции библиотеки `string.h`:

- `unsigned int strlen(char* str)` - возвращает длину строки
- `char* strcpy (char* a, char* b))` - копирует строку b в строку a, т.е. a = b.
- `int strcmp(const char* a, char* b)` - лексикографическое сравнение строк (возвращает 0, если строки одинаковые, положительное, если первая строка больше, и отрицательное, если меньше)
- `char* strcat(char* a, char* b)` - приклеивает копию строки b к строке a.
- `char* strstr(char* a, char* b)` - ищет строку `b` в строке `a`. Возвращает указатель на первый символ вхождения строки `b` или `0 (NULL)` если такой строки нет.

```
#include <stdio.h>
#include <string.h>
int main()
{
        char a[100] = "Dog";
        char b[100] = "Mice";
        // Строки это массивы, поэтому их нельзя просто присваивать
        //( можно только при инициализации )
        a = "Cat"; // Это не будет работать! Нужно использовать strcpy:
        strcpy(a, "Cat");

        // Строки это массивы, поэтому их нельзя просто сравнивать
        a == b; // Это не будет работать! Нужно использовать strcmp:
        printf("%d\n", strcmp(a, b));
        // Напечатает -1 так как a < b, то есть "Cat" < "Mice", тк.. 'C' < 'M'

        // Конкатенация ( склейка ) строк. Можно воспринимать как +=
        strcat(a, b);
        printf("%s\n", a); // Напечатает CatMice

        // Считывание слов из файла
        FILE* infile = fopen("words.txt", "r");
        char words[500][100];
        int number_of_words = 0;
        while (fscanf(infile, "%s", words[number_of_words]) != -1)
                number_of_words++;
        fclose(infile);
}
```

- **Обмен строк:** Напишите функцию `void swap_strings(char* a, char* b)`, которая будет обменивать значениями две строки. Используйте стандартную функцию `strcpy`. Предполагается, что размер каждой из строк ограничен 100 символами.
- **Поиск подстроки:** Считать 2 строки и проверить является ли вторая строка подстрокой первой строки. Вывести на экран YES или NO соответственно.
- **Чтение из файла:** Напишите программу, которая будет считывать слова из файла и записывать их в массив `char words[1000][100]`. Когда в файле слов для считывания не останется, функция `fscanf` будет возвращать `-1`. После этого все слова должны быть напечатаны на экран через пробел.
- **Сортировка слов:** Напишите программу, которая будет считывать слова из файла и записывать их в массив `words`. После этого все слова должны быть отсортированы по алфавиту и записаны в файл `sorted_words.txt`.