Семинар #5: Строки. Классные задачи. Решения.

Таблица ASCII

Символ	Код	С	К	С	К	С	К	С	К	С	К	С	К	С	К	С	К	С	К
\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		

Часть 1: Символы.

Тип `char` – однобайтовое целое число

Тип `char` – это тип целочисленных чисел размером 1 байт (соответственно диапазон от `-128` до `127`).
Для считывания и печати переменных типа `char` используется спецификатор `%hhi` (смотрите второй семинар).

```c
#include <stdio.h>
int main() {
    char a = 64;
    char b;
    scanf("%hhi", &b);
    printf("%hhi\n", a + b);
}
```

- Что напечатает эта программа, если на вход передать число `10`?
- Что напечатает эта программа, если на вход передать число `100`?
- Напишите программу, которая принимает на вход 2 числа, сохраняет их в переменных типа `char` и печатает результат произведения этих чисел.

Спецификатор `%c` в функции `printf`

`char` используется для хранения кодов символов. Функция `printf` со спецификатором `%c` принимает на вход число и печатает соответствующий символ по таблице `ASCII`. Что напечатает следующая программа?

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

- Напишите программу которая будет печатать символ `^`

```c
#include <stdio.h>
int main() {
    printf("%c\n", 94);
}
```

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

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

```
#include <stdio.h>
int main() {
    for (int i = 32; i <= 126; ++i) {
        printf("Symbol = %c, Code = %i\n", i, i);
    }
}
```

Спецификатор %c в функции scanf

Функция scanf со спецификатором %c считывает 1 символ и записывает код ASCII этого символа по соответствующему адресу.

```
#include <stdio.h>
int main() {
    char x;
    scanf("%c", &x);
    printf("%hhi\n", x);
}
```

- Измените программу выше так, чтобы она считывала 1 символ и печатала и этот символ и его код.

```
#include <stdio.h>
int main() {
    char x;
    scanf("%c", &x);
    printf("%c %hhi\n", x, x);
}
```

- Напишите программу, которая будет постоянно считывать символы в цикле while и печатать эти символы и их коды. Программа должна заканчиваться после ввода с клавиатуры символа q.

```
#include <stdio.h>
int main() {
    while (1) {
        char x;
        scanf("%c", &x);
        if (x == 'q') {
            break;
        }
        if (x != '\n') {
            printf("Symbol: %c   Code: %i\n", x, x);
        }
    }
}
```

Можно и без условия if (x != '\n'), но тогда будут печататься коды переносов строк.

- Напишите программу, которая будет считывать символ и печатать:

– `Uppercase Letter`, если этот символ – заглавная буква.
– `Lowercase Letter`, если этот символ – строчная буква.
– `Digit`, если этот символ – цифра.
– `Other`, если это какой-то другой символ.

```c
#include <stdio.h>
int main() {
    char x;
    scanf("%c", &x);
    if (x >= 'A' && x <= 'Z') {
        printf("Uppercase Letter\n");
    }
    else if (x >= 'a' && x <= 'z') {
        printf("Lowercase Letter\n");
    }
    else if (x >= '0' && x <= '9') {
        printf("Digit\n");
    }
    else {
        printf("Other\n");
    }
}
```

Символьные константы

Для удобства работы с символами с языке были введены символьные константы. В коде они выглядят как символы в одинарных кавычках, но являются просто числами, соответствующими коду символа.

```c
#include <stdio.h>
int main() {
    int a = '@'; // Теперь a равно 64
    int b = '5'; // Теперь b равно 53
    printf("%d %d\n", a, b);

    // Проверьте себя. Что напечатает следующий код?
    printf("%d\n", '>');
    printf("%d\n", '1');
    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');
}
```

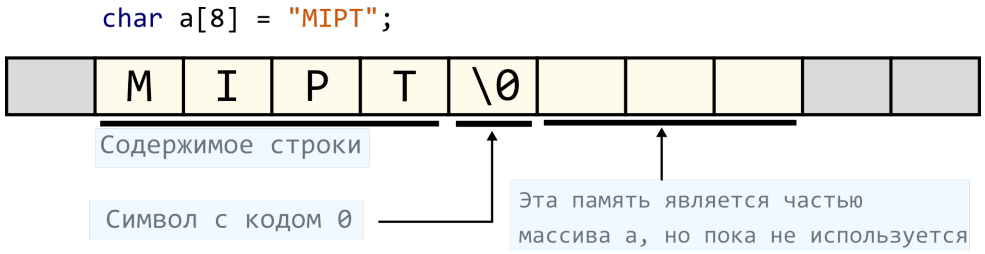
- Напишите программу, которая будет считывать символ и, если этот символ является строчной буквой, то делать эту букву заглавной и печатать её. Если символ – не строчная буква, то нужно просто напечатать его.

вход	выход
a	A
l	L
5	5

```c
#include <stdio.h>
int main() {
    char x;
    scanf("%c", &x);
    if (x >= 'a' && x <= 'z') {
        x -= ('a' - 'A');
    }
    printf("%c\n", x);
}
```

Часть 2: Строки:

Строки - это массивы чисел типа `char`, которые хранят коды символов. Самое значительное отличие строк от массивов это то, что конец строки задаётся как элемент массива символом с кодом 0.



Объявление, инициализация и изменение строк

Создавать строки можно также как и массивы, а можно и с помощью строки в двойных кавычках.

```c
int main() {
    char a[10] = {77, 73, 80, 84, 0};
    char b[10] = {'M', 'I', 'P', 'T', '\0'};
    char c[10] = "MIPT"; // Символ 0 поставится автоматически

    // Использовать = со строками можно только при создании, то есть это работать не будет:
    a = "FAKT";

    // Изменение элементов строк работает также как и у массивов
    a[1] = 'A';
}
```

Печать строк. Спецификатор %s

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

```c
#include <stdio.h>
int main() {
    char a[10] = "MIPT";
    // Печатаем каждый символ по отдельности ( идём циклом до нулевого символа )
    for (int i = 0; a[i] != 0; ++i) {
        printf("%c", a[i]);
    }
    printf("\n");
```

```
    // Печатаем всю строку целиком
    printf("%s\n", a);
}
```

- Создайте строку `str` с содержимым `"Cat"` 3-мя разными способами и напечатайте её.

```
#include <stdio.h>
int main() {
    char a[10] = {67, 97, 116, 0};
    char b[10] = {'C', 'a', 't', '\0'};
    char c[10] = "Cat"; // Символ 0 поставится автоматически
        printf("%s %s %s\n", a, b, c);
}
```

- Измените созданную строку из прошлой задачи на `"Dog"` и снова напечатайте её.

```
#include <stdio.h>
int main() {
    char a[10] = "Cat";
    printf("%s\n", a);
    a[0] = 'D';
    a[1] = 'o';
    a[2] = 'g';
    printf("%s\n", a);
}
```

Считывание строк

```
#include <stdio.h>
int main() {
    char a[100];
    // Считываем каждый символ по отдельности до пробела или переноса строки ( сложный способ )
    for (int i = 0; 1; ++i) {
        char x;
        scanf("%c", &x);
        if (x == ' ' || x == '\n') {
            a[i] = 0;
            break;
        }
        a[i] = x;
    }
    printf("%s\n", a);

    // То же самое с помощью спецификатора %s ( простой способ )
    scanf("%s", a);
    printf("%s\n", a);
}
```

- Считайте число `n` и строку `str` и напечатайте её `n` раз через пробел. Для считывания используйте `scanf` со спецификатором `%s`.

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

- **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

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

- **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	

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

- **Сумма цифр:** На вход передаётся целое положительное число $n < 10^{10000}$. Нужно сумму цифр этого числа. *Подсказка:* Считайте это число как строку.

вход	выход
97	16
123456789098765432123456789098765432 1	179

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

Строки и функции

Строки передаются в функции также как и массивы. То есть при изменении строки внутри функции она меняется и снаружи. Но только при передаче строки не обязательно передавать её размер, так как граница строки задаётся нулевым символом. Пример функции, которая заменяет один символ в строке на другой:

```
#include <stdio.h>
void change_letter(char str[], char from, char to) {
    int i = 0;
    while (str[i]) {
        if (str[i] == from) {
            str[i] = to;
        }
        i++;
    }
}
int main() {
    char a[100] = "Sapere aude";
    printf("%s\n", a);
    change_letter(a, 'e', '#');
    printf("%s\n", a);
}
```

- **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()`.

вход	выход
Hello!	!olleH
live	evil
Madam	madaM

```
#include <stdio.h>

int get_length(char str[]) {
    int i = 0;
    while (str[i]) {
        i++;
    }
    return i;
}

void reverse(char str[]) {
    int length = get_length(str);
    for (int i = 0; i < length / 2; ++i) {
        char temp = str[i];
        str[i] = str[length - 1 - i];
        str[length - 1 - i] = temp;
    }
}
```

```c
int main() {
    char str[100];
    scanf("%s", str);
    printf("str = %s\n", str);
    printf("length = %i\n", get_length(str));
    reverse(str);
    printf("reversed str = %s\n", str);
}
```

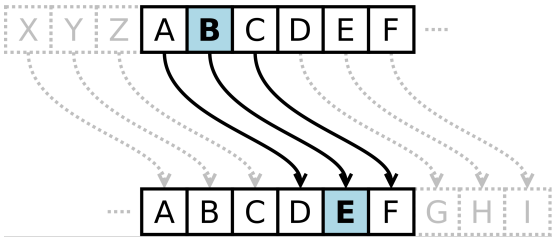
- **Uppercase:** Напишите функцию `void to_upper_case(char str[])`, которая будет переводить строку в верхний регистр. Проверьте эту функцию в `main()`.

вход	выход
mipt	MIPT
Hello!	HELLO!
Area51	AREA51

```c
#include <stdio.h>

void to_upper_case(char str[]) {
    for (int i = 0; str[i]; ++i) {
        if (str[i] >= 'a' && str[i] <= 'z') {
            str[i] -= 'a' - 'A';
        }
    }
}
int main() {
    char str[100];
    scanf("%s", str);
    to_upper_case(str);
    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

```
#include <stdio.h>

void encrypt(char str[], int k) {
    for (int i = 0; str[i]; ++i) {
        if (str[i] >= 'a' && str[i] <= 'z') {
            str[i] = 'a' + (str[i] - 'a' + k) % 26;
        }
        if (str[i] >= 'A' && str[i] <= 'Z') {
            str[i] = 'A' + (str[i] - 'A' + k) % 26;
        }
    }
}
int main() {
    int k;
    char str[100];
    scanf("%i %[^\n]", &k, str);
    encrypt(str, k);
    printf("%s\n", str);

}
```

Считывание до заданного символа

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);
}
```

- **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

```c
#include <stdio.h>

void reverse_substring(char str[], int l, int r) {
    for (int i = l; i < l + (r - l) / 2; ++i) {
        char temp = str[i];
        str[i] = str[r - 1 - i + l];
        str[r - 1 - i + l] = temp;
    }
}

int is_letter(char c) {
    return (c >= 'A' && c <= 'Z') || (c >= 'a' && c <= 'z');
}

void reverse_all_words(char str[]) {
    int word_start = 0;
    int is_prev_letter = is_letter(str[0]);
    for (int i = 1; str[i - 1]; ++i) {

        int is_curr_letter = is_letter(str[i]);

        if (!is_prev_letter && is_curr_letter) {
            word_start = i;
        }
        else if (is_prev_letter && !is_curr_letter) {
            reverse_substring(str, word_start, i);
        }
        is_prev_letter = is_curr_letter;
    }
}
int main() {
    char str[500];
    scanf("%[^\n]", str);
    reverse_all_words(str);
    printf("%s\n", str);

}
```

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

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

```c
#include <stdio.h>

void sort(char str[]) {
    for (int i = 0; str[i]; ++i) {

        int min_index = i;
        for (int j = i; str[j]; ++j)
            if (str[j] < str[min_index])
                min_index = j;

        char temp = str[i];
        str[i] = str[min_index];
        str[min_index] = temp;
    }
}
int main() {
    char str[500];
    scanf("%[^\n]", str);
    sort(str);
    printf("%s\n", str);

}
```

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

вход	выход
12345678909876543212345678909876543 21	37037036729629629637037036729629629 63

```c
#include <stdio.h>

int get_length(char str[]) {
    int i = 0;
    while (str[i]) {
        i++;
    }
    return i;
}

void multiply3(char str[]) {
    int length = get_length(str);
    for (int i = 0; i < length; ++i) {
        str[i] -= '0';
    }

    int carry = 0;
    for (int i = length - 1; i >= 0; --i) {
        str[i] = 3 * str[i] + carry;
        carry = str[i] / 10;
        str[i] = str[i] % 10;
    }

    if (carry != 0) {
        for (int i = length; i >= 0; --i) {
            str[i + 1] = str[i];
        }
        str[0] = carry;
        length += 1;
    }

    for (int i = 0; i < length; ++i) {
        str[i] += '0';
    }
}
int main() {
    char str[10000];
    scanf("%s", str);
    multiply3(str);
    printf("%s\n", str);
}
```

Часть 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. аналог `a = b`. Возвращает указатель на `a`.
- `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`, т.е. аналог `a += b`.
- `char* strstr(char a[], char b[])` - ищет строку `b` в строке `a`. Возвращает указатель на первый символ вхождения строки `b` или `0` если такой строки нет.

```c
#include <stdio.h>
#include <string.h>
int main() {
        char a[100] = "Dog";
        char b[100] = "Mice";

        // Строки это массивы, поэтому их нельзя просто присваивать
        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));

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

- Считайте строку и напечатайте её длину. Используйте функцию `strlen`.

```c
#include <stdio.h>
#include <string.h>
int main() {
    char str[100];
    scanf("%s", str);
    printf("%i\n", strlen(str));
}
```

- **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.

```c
#include <stdio.h>
#include <string.h>

void swap_strings(char a[], char b[]) {
    char temp[100];
    strcpy(temp, a);
    strcpy(a, b);
    strcpy(b, temp);
}

int main() {
    char a[100] = "Cat";
    char b[100] = "Elephant";
    printf("a = %s, b = %s\n", a, b);
    swap_strings(a, b);
    printf("a = %s, b = %s\n", a, b);
}
```

Надо отметить, что это далеко не самое эффективное решение этой задачи, так как, если строки будут очень длинными, то придётся перекопировать большое количество данных. Более оптимальный способ – это хранить указатель на строки и обменивать указатели.

- **Поиск подстроки:** Считать 2 строки и проверить является ли вторая строка подстрокой первой строки. Вывести на экран `YES` или `NO` соответственно.

```c
#include <stdio.h>
#include <string.h>

int main() {
    char a[100];
    char b[100];
    scanf("%s%s", a, b);

    if (strstr(a, b)) {
        printf("Yes\n");
    }
    else {
        printf("No\n");
    }
}
```

Чтение из файла

Пример программу, которая подсчитывает количество слов в файле.

```
#include <stdio.h>
#include <string.h>
int main() {
        // 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);
}
```

- **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`.

```
#include <stdio.h>
#include <string.h>

#define MAX_NUM_WORDS 1000
#define MAX_WORD_SIZE 100


// Функция, которая сортирует указатели на слова сортировкой выбора
// Это не самая эффективная сортировка
// Лучше использовать други сортировки, такие как быстрая, сортировка слиянием и другие
void sort_pointer(char* pointers[], int size) {
    for (int i = 0; i < size; ++i) {
        int min_index = i;
        for (int j = i + 1; j < size; ++j)
            if (strcmp(pointers[j], pointers[min_index]) < 0)
                min_index = j;


        char* temp = pointers[i];
        pointers[i] = pointers[min_index];
        pointers[min_index] = temp;
    }

}
```

```
int main() {
    // Создаём массив строк
    char words[MAX_NUM_WORDS][MAX_WORD_SIZE];

    // Считываем все слова из файла words.txt
    FILE* infile = fopen("words.txt", "r");
    int number_of_words = 0;
    while (fscanf(infile, "%s", words[number_of_words]) != -1) {
        number_of_words++;
    }
    fclose(infile);

    // Сохраняем все слова в файл без изменений
    FILE* outfile = fopen("copy_words.txt", "w");
    for (int i = 0; i < number_of_words; ++i) {
        fprintf(outfile, "%s ", words[i]);
    }
    fclose(outfile);


    // Для более эффективной сортировки слов создаём массив указателей
    // i - й указатель указывает на начало i - го слова
    char* pointers[MAX_NUM_WORDS];
    for (int i = 0; i < number_of_words; ++i) {
        pointers[i] = &words[i][0];
    }

    // Сортируем указатели
    sort_pointer(pointers, number_of_words);

    // Сохраняем отсортированные слова
    FILE* sortedfile = fopen("sorted_words.txt", "w");
    for (int i = 0; i < number_of_words; ++i) {
        fprintf(sortedfile, "%s ", pointers[i]);
    }
    fclose(sortedfile);
}
```