# Лекция 8.
# Си: Текстовый файл, двухмерный массив, строки

В языке Си есть два вида файлов: текстовый и бинарные. Файловая переменная – дескриптор файла – для любого из этих двух типов описывается с помощью типа *FILE* из модуля *stdio.h*, который необходимо подключить к своей программе.

До конца семестра для работы с файлами использовать только тип *FILE*, а не классы (*fstream, ifstream, ofstream*). А также не использовать классы *string, list, stack* и т.д. и вообще не использовать классы – не свои, не чужие; по плану дисциплины – изучаем процедурное программирование на двух языках высокого уровня.

Как и в *Delphi* в языке *C* имеются стандартные и нестандартные текстовые файлы (потоки). Поток стандартного ввода – *stdin* – связан по умолчанию с клавиатурой, а поток стандартного вывода – *stdout* – связан по умолчанию с экраном для консольного вывода. В *Delphi* эти стандартные имена имели имена *input* и *output* (см.Практическое занятие №2 в первой части курса). Для стандартных потоков можно использовать не только функции консольного ввода и вывода из модуля *conio.h*, но и функции для работы с любыми текстовыми файлами из *stdio.h*, указывая в них вместо файлов дескрипторов – имена *stdin* и *stdout*.

**Порядок работы с текстовыми файлами:**

| Способ / Действие | **Создание** нового **потока** ввода/вывода и определения дескриптора файла. Позволяет использовать несколько потоков ввода/вывода одновременно. | **Переопределение потоков** стандартного ввода (***stdin***) и вывода (***stdout***), которые по умолчанию связаны с клавиатурой и экраном для консольного ввода и вывода |
|---|---|---|
| Требуется подключить модуль | #include <stdio.h> | #include <stdio.h><br>#include <conio.h> |
| Описание переменной | FILE *f; | FILE *f; |
| Открытие для чтения (r/rt) * ** | f = fopen("File1.txt", "r"); | f = freopen("File1.txt","r", stdin); |
| Открытие для записи с созданием | f = fopen("d:\\File1.txt", "w"); | f = freopen("F1.txt","w", stdout); |
| Открытие для дозаписи в конец | f=fopen("d:\\Dir\\F1.txt", "a"); | f = freopen("F1.txt","a", stdout); |
| Чтение чисел (%i,%d,%u, %f,%lf), символов (%c) и токенов (без разделителей)(%s) | fscanf(f, "%d %f", &i, &r); | scanf("%d %f", &i, &r);<br>без указания файла<br>как для ввода с клавиатуры |
| Запись чисел | fprintf(f, “i=%d r=%f”, i, r); | printf(“i=%d r=%f”, i, r); |
| Чтение символов char ch | ch = getc(f);<br>или fscanf(f, "%c", &ch); | ch = getch();или scanf("%c",&ch);<br>с клавиатуры только:<br>ch = getchar(); //эхоповтор, Enter |
| Запись символов | putc(ch, f);<br>или fprintf(f, "%c",ch); | putchar(ch);<br>или printf("%c",ch); |
| Чтение строк char str[80] из нескольких слов | fgets(str, 79, f); | fgets(str, 79, stdin);<br>gets(str); // с клавиатуры |
| Запись строк (до символа ‘\0’) | fputs(str, f);<br>или fprintf(f,"%s\n",str); | puts(str); fputs(str, stdin);<br>или printf("%s\n",str); |
| Закрытие файла | fclose(f) | fclose(f) |
| Дополнительные функции | feof(f) – достигнут конец файла, fflush, rewind, rename, remove, … | |

* *Букву t* (текстовый файл) можно не писать, она подразумевается по умолчанию.

** *Относительный путь* к файлу указывается от расположения **exe**-файла. Например, в папке ***Debug, Release*** или папке проекта в зависимости от версии среды программирования и режима компиляции.

*Слеш (\) в имени файла удваивается*, так как является служебным символом в строке, например, \n \t \0 \" \\

**Пример 1.** Посимвольное копирование файла, включая символы перевода каретки (‘\r’) и конца строки (‘\n’).
Имена файлов переданы через *параметры программы*.
Файлы в этой программе могут быть открыты как текстовые (*r* или *rt*, *w* или *wt*) или как бинарные/двоичные (*rb* или *r+b* и *wb* или *w+b*). *t = **t**ext, b = **b**inary, r = **r**ead, w =create+**w**rite, только для двоичных: r+b = read+write, w+b= create+write+read*

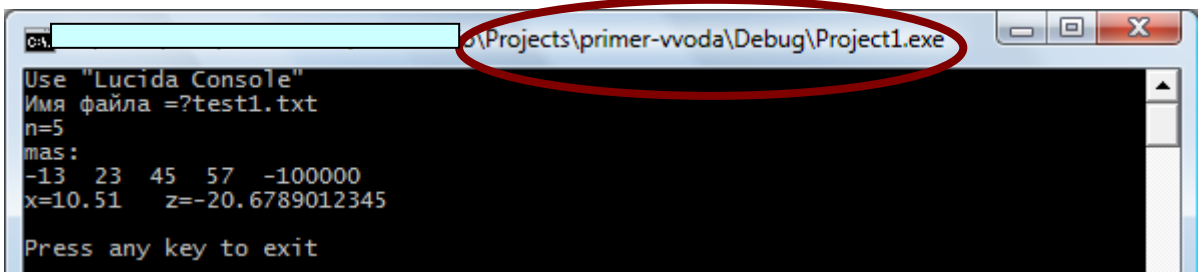
```
//---------------------------------------------------------------------------
#include <stdio.h>  // FILE fopen fclose fprintf, fscanf, printf, scanf
#include <conio.h>  // getch
#include <windows.h> // SetConsoleOutputCP, SetConsoleCP
//---------------------------------------------------------------------------
void main(int argc, char *argv[]) {  // количество параметров и их значения в массиве строк
	SetConsoleOutputCP(1251); // вывод в кодировке 1251
	printf("Use \"Lucida Console\"\n"); // не забудьте в свойствах консольного окна указать шрифт
	if (argc!=3){  // три параметра: путь+имя запущенного файла и два имени текстовых фалов
		printf("Error: Вы забыли передать имена файлов\nPress any key to continue...");
		getch();
		return;
	}
	FILE *fin, *fout;
	if ((fin = fopen( argv[1], "rt")) ==NULL) {
		printf("Errror: Ошибка при открытии файла %s для чтения\nPress any key", argv[1]);
		getch(); // ожидание нажатия клавиши
		return;  // выход из функции main
	}
	if ((fout = fopen( argv[2], "wt")) ==NULL) {
		printf("Errror: Ошибка при открытии файла %s для записи\nPress any key", argv[2]);
		fclose(fin);
		getch(); // ожидание нажатия клавиши
		return;  // выход из функции main
	}

	// посимвольное чтение и копирование файла:
	char ch;
	while (!feof(fin)){
		ch = getc(fin);
		if (!feof(fin)) putc(ch, fout);  // конец файла в Си «по факту» после неудачного чтения
	}

	fclose(fin);  fclose(fout);
	printf("THE END\nPress any key to exit");  getch();
	return;
}
//---------------------------------------------------------------------------
```

**Пример 2. Ввод одномерного массива *mas* из *n* элементов из текстового файла.**
**Кроме того, в третьей строке** есть три вещественных числа: первые два типа *float*, третий – *double*; надо считать два из них: первый в *x* типа *float* и третий – в *z* типа *double*.
Пусть есть файл из 3 строк:

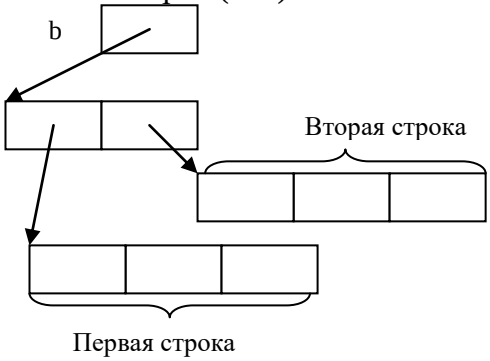
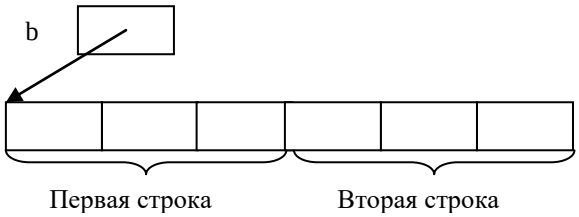
| | |
|---|---|
| 5 | <n> |
| -13 23 45 57 -100000 | <mas[0]> <mas[1]> ... <mas[n-1]> |
| 10.51  3.14 -20.6789012345 | <x> <> <z> |

```
//------------------------------подключение модулей-------------------------------------------
#include <stdio.h>  // FILE fopen fclose fprintf, fscanf, printf, scanf
#include <conio.h>  // getch
#include <windows.h> // SetConsoleOutputCP, SetConsoleCP
//------------------------------главная функция программы--------------------------------
void main()
{
	SetConsoleOutputCP(1251);  // смена кодировки при выводе
	printf("Use \"Lucida Console\"\n"); // не забудьте в свойствах консольного окна указать шрифт

	char fn[80];  // статический массив из символов – статическая строка
	printf("Имя файла =?"); // приглашение
	gets(fn);  //Ввод строки – имени текстового файла (по умолчанию путь – путь к EXE-файлу)

	FILE *f;
	f = fopen( fn, "rt");  // текстовый файл открыть для чтения только
	if (!f) {		// или if (f ==NULL) {
		printf("Error: Ошибка при открытии файла %s для чтения\nPress any key", fn);
		getch(); // ожидание нажатия клавиши
		return;  // выход из функции main
	}

	int n, *mas; // целое, динамический массив из целых – указатель на целое
 // ввод и вывод длины массива из открытого текстового файла, связанного с переменной f
	fscanf(f, "%d", &n); // считать десятичное целое и сохранить по адресу переменной n
	printf("n=%d\n", n); // вывод на экран сообщения “n=”, n и символа конца строки  ‘\n’

// ввод и вывод n значений одномерного динамического массива
	int i;  // значения индекса для ввода и вывода массива
	mas = new int [n]; // выделение памяти для массива из n элементов типа int
	for(i=0; i<n; i++)
		fscanf(f, "%d", &mas[i]); // ввод по-элементный массива из файла, связанного с f
	printf("mas:\n"); // вывод на экран
	for(i=0; i<n; i++) {
		printf("%d  ", mas[i]); // вывод в 10с/с элементов массива через два пробела
	}
	printf("\n");     // вывести строку из одного символа – символа конца строки

	// освобождение памяти одномерного динамического массива:
	delete [] mas;

	// описание, ввод и вывод из третьей строки файла
	float x; // вещ.число с плавающей точкой одинарной точности
	double z; // вещ.число с плавающей точкой двойной точности
	fscanf(f, "%f %*f %lf", &x, &z);  // ввод float(%f) x (передаем адрес памяти (&x) для
			// сохранения считанного значения), пропуск второго числа float (%*f),
			// чтение третьего значения типа double (%lf) (long float)
			// в память по адресу переменной z
	printf("x=%4.2f   z=%14.10lf\n", x, z); // форматированный вывод x и z

	fclose(f); // закрытие файла
	printf("\nPress any key to exit");
	getch(); // задержка экрана до нажатия любой клавиши
	return; //выход из main()
} //---------------------------------------------------------------------------
```

| | |
|---|---|
| В файле test1.txt, лежащим около EXE-файла (в папке Debug или Release | |
| 5<br>-13 23 45 57 -100000<br>10.51 3.14 -20.6789012345 | <n><br><mas[0]> <mas[1]> ... <mas[n-1]><br><x> <> <z> |




## Двухмерный массив

| | Динамический (#include <stdlib.h> или <cstdlib> или <alloc.h>) | Статический |
|---|---|---|
| Описание переменной | int **a, **b; | int a[5][3];<br>int b[2][3]={{0,1,2},{3,4,5}};<br>int c[][3]={{0,1,2},{3,4,5}}; |
| Выделение памяти | a = new int*[5]; // 5x3<br>for(int i=0; i<5; i++) a[i] = new int[3];<br><br>b = (int**)calloc(2, sizeof(int*)); // 2x3<br>for(int i=0; i<2; i++)<br>b[i]= (int*)calloc(3, sizeof(int)); | При *описании* переменной указывается на сколько элементов |
| Доступ к элементу | a[0][0] **a<br>b[i][j] *(*((int*)b+i)+j) | a[0][0] **a<br>b[i][j] *((int*)b+i*3+j) |
| Освобождение памяти | for(i=0;i<5;i++) delete [] a[i];<br>delete [] a;<br><br>for(int i=0;i<2;i++) free(b[i]);<br>free(b); | При выходе из области видимости переменной, если не static |
| Описание параметра | Для изменения значений<br>void proc1(int *a[], int **b);<br>void proc2(int **a, int *b[]); | Для изменения значений<br>void proc1(int a[5][3], int b[2][3]);<br>void proc2(int *a[3], int b[][3]); |
| | Для выделения/освобождения памяти<br>void pr3(int*(*a)[], int***b); // pr3(&a, &b);<br>void pr4(int***a, int*(*b)[]); // pr4(&a, &b);<br>void pr5(int** &a, int** &b); // pr5(a, b);<br>void* func(int **a); //a=(int**)func(a); | *Невозможно* освободить/ перевыделить память во время выполнения программы |
| Структура | Указатель на массив указателей на первые элементы строк (2x3)<br>  | Указатель на первый элемент<br>Строки одна за другой (2x3)<br>  |

## Строки в языке Си

В языке С строки представляются в виде массива элементов типа char. Это означает, что символы строки можно представить расположенными в соседних ячейках памяти – по одному символу в ячейке. Последним элементом массива является символ '\0' - это нуль-символ: в языке С он используется для того, чтобы отмечать конец строки. Нуль-символ – не есть цифра 0; он не выводится на печать и в таблице кодов ASCII имеет код 0. Наличие нуль-символа означает, что количество ячеек массива должно быть по крайней мере на одну больше, чем количество символов, которые необходимо размещать в памяти.

Для представления внутри строки символа "кавычка", перед ней нужно располагать символ '\'. Точно так же можно получить представление и других управляющих символов внутри строки. Например, '\n' – символ перехода на новую строку, '\t' – символ табуляции.

**Создание строк:**

char str1[10]; // Строка - массив из 10 символов. Начальное значение символов не определено.
char str2[10]="Hello";// с инициализацией
/* Используется инициализация (не присваивание!). В первые 5 символов записывается “Hello”, в 6 – нуль-символ, значение трех последних не определено.*/
char str3[10]={'H', 'e', 'l', 'l', 'o', '**0**'}; //Эквивалентно предыдущему.
char str4[10]="Very long line"; //Ошибка. Массив из 10 элементов нельзя инициировать более длинной последовательностью.
char str5[]="Very long line"; /*Компилятор автоматически определяет длину массива (в нашем случае 15 с \0) и инициализирует его последовательностью символов. */
char* str6; /*Строка - указатель на символ. В большинстве случаев для ее использования потребуется выделить память:*/
str6=(char*) malloc(10);
free(str6);

**Присваивание строк**

Первый и самый очевидный способ присваивания строк – присваивание отдельных символов. Например,
str1[0]=’H’;
str1[1]=’e’;
str1[2]=’l’;
str1[3]=’l’;
str1[4]=’o’;
str1[5]=’\0’;
---
Несколько примеров НЕправильного присваивания:
char str1[10], str2[10];
str1="Hello"; // можно только при описании, здесь надо str1=**strcpy**(str1, "Hello");
str2=str1; // надо str2=**strcpy**(str2, str1);
---
char str1[10]= "Hello"; // присвоить так можно только при описании,
char* str2;

```
str2=str1; // не копия, указатель на ту же строку,
                  // для копии надо str2=strcpy(str2, str1);
str2[1]='u'; //изменит строку str1
```

**Функции для работы со строками**

**char *strcat(char *dest, const char *scr);** Объединяет исходную строку scr и результирующую строку dest, присоединяя первую к последней. Возвращает dest.

**int strlen(const char *s);** Возвращает длину строки s - количество символов, предшествующих нулевому символу.

**char *strtok(char *s1, const char *s2);** Делит исходную строку s1 на лексемы (подстроки), разделенные одним или несколькими символами из строки s2. Функция strtok() возвращает адрес первого компонента, отделенного от следующего заданным разделителем. Если заданные разделители не обнаружены, функция strtok() возвращает нуль. Кроме того, эта функция настраивает свой внутренний указатель на символ, следующий за концом последней сформированной подстроки, чтобы последующий вызов strtok() мог продолжить разложение строки. Для этого передайте в качестве первого аргумента значение NULL - это послужит сигналом для функции strtok() использовать ее внутренний указатель как стартовый адрес для поиска другого разделителя. Таким образом, чтобы выделить другие компоненты строки, нужно просто вызвать функцию strtok() несколько раз (в данном примере в цикле), и каждый раз первым ее аргументом должен быть NULL.

**char *strchr(const char *s, int c);** Ищет в строке **s** первое вхождение символа (**char**)**c**, начиная с начала строки s. В случае успеха возвращает указатель на найденный символ, иначе возвращает NULL. Есть аналогичный поиск последнего вхождения: **strrchr.**

```
#include <stdio.h>
#include <conio.h>
#include <string.h>
int main(void)
{
 char *str1 = "CodeGear", ch = 'e', *ptr;
 ptr = strchr(str1, ch);
 if (ptr!=NULL) {
  printf("The first occurence of the char '%c' in the string \"%s\": \"%s\"\n",
              ch, str1, ptr);
  printf("Its positoin is: %d (0,1,2,...)\n", ptr-str1);

  char* ptr2 = strrchr(str1, ch);
  if (ptr2) {
   printf("The last occurence of the char '%c'in the string \"%s\": \"%s\"\n",
              ch, str1, ptr2);
   printf("Its positoin is: %d (0,1,2,...)\n", ptr2-str1);
  }
 }
 else
  printf("The char was not found\n", ptr);
```

```
  getch();
  return 0;
}
```

**char *strstr(const char *s, const char *subs);** Ищет в строке **s** первое вхождение подстроки subs, начиная с начала строки s. В случае успеха возвращает указатель на первый символ подстроки subs в строке s, иначе возвращает NULL.

```
#include <stdio.h>
#include <conio.h>
#include <string.h>
int main(void)
{
   char *str1 = "CodeGear", *str2 = "de", *ptr;
   ptr = strstr(str1, str2);
   printf("The substring is: %s\n", ptr);
   printf("Its positoin is: %d (0,1,2,...)\n", ptr-str1);
   getch();
   return 0;
}
```

**int strcmp(const char *s1, const char *s2);** Сравнивает две строки. Возвращает отрицательное значение, если s1<s2; нуль, если s1==s2; положительное значение, если s1>s2. Параметры - указатели на сравниваемые строки.

**char *strcpy(char *dest, const char *src);** Копирует исходную строку src и завершающий ее нулевой символ в строку результата dest. Возвращает dest.

**double atof(const char *s);** Преобразует строку s в число с плавающей точкой типа double. Заголовочный файл - math.h

**int atoi(const char *s);** Преобразует строку s в число типа int. Возвращает значение или нуль, если строку преобразовать нельзя. Заголовочный файл - stdlib.h

**long atol(const char *s);** Преобразует строку s в число типа long. Возвращает значение или нуль, если строку преобразовать нельзя. Заголовочный файл - stdlib.h

**char *itoa(int value, char *s, int radix);** Преобразует значение целого типа value в строку s. Возвращает указатель на результирующую строку. Значение radix - основание системы счисления, используемое при преобразовании (от 2 до 36). Заголовочный файл - stdlib.h

Подробнее про функции работы со строками: http://it.kgsu.ru/C++/c0063.html

**void *malloc(size_t size);** Функция malloc() возвращает указатель на первый байт области памяти размером size, которая была выделена из динамически распределяемой области памяти. Если для удовлетворения запроса в динамически распределяемой области памяти нет достаточного объема памяти, возвращается нулевой указатель.

**Функции для чтения строк и символов**

**int getc ( FILE * fp );** Возвращает очередной символ из потока, на который указывает fp; при ошибке или в случае конца файла возвращается EOF.

**int putc ( int c, FILE * fp );** Записывает символ c в файл fp и возвращает записанный символ или EOF в случае ошибки.

**char * fgets ( char * line, int maxline, FILE * fp );** Считывает очередную строку (вместе с символом ее конца) из файла fp  массив символов line; в общей сложности

читается не более maxline-1 символов. Получившаяся строка завершается нулевым символом ‘\0’. В конце файла или в случае ошибки она возвращает NULL.

**int fputs ( const char * line, FILE * fp );** Записывает строку line в файл fp. Возвращает EOF в случае ошибки и 0 в противном случае.

**int getchar();** Возвращает значение символа (если он есть), который пользователь набрал на клавиатуре. После ввода символа нужно нажать клавишу Enter. Заголовочный файл - stdio.h

**int getch();** Аналогично предыдущему, только символ на экране не отображается. Используется чаще для организации задержки выполнения программы. Заголовочный файл - conio.h

**int putchar(int c);** Выводит символ ***c*** на экран. В случае успеха возвращает сам символ ***c***, в противном случае - EOF. Заголовочный файл - stdio.h

**и много другних…**