Лекция 10

Файлы. Функции для работы с файлами.

Понятие файла

Файл – это именованная область данных на каком-либо носителе информации.

Типы файлов (относительно языка «С»):

- текстовые;
- бинарные.

Действия над файлами

Основные операции:

1. Открытие файлов.
2. Чтение и запись данных.
3. Закрытие файлов.

Дополнительные операции:

1. Навигация по файлу.
2. Обработка ошибок.
3. Удаление и переименование файлов.

Описание переменной

Библиотека **stdio.h**

FILE *имя = NULL;

Открытие файла

FILE *fopen(const char *filename, const char *mode);

- filename – название файла.
- mode – режим открытия.

Функция возвращает указатель на файл, если тот был успешно открыт. В противном случае – NULL.

Правила указания имени файла

Только имя, если файл находится в текущем каталоге. Иначе необходимо указать абсолютный или относительный путь к файлу.

Примеры:

“data.txt”

“..\\files\\data.txt”

“d:\\temp\\data.txt”

Режимы открытия

r	Только чтение.
w	Только запись. Если файл существовал, то он переписывается.
a	Добавление: открытие файла для записи в конец, или создание файла.
r+	Открывает файл для обновления (чтение и запись).
w+	Открывает файл для обновления (чтение и запись), переписывая файл, если он существует.
a+	Открывает файл для записи в конец файла или для чтения.

Перенаправление потоков

FILE * **freopen(**const char **filename*, const char **mode*, FILE **stream***)**;

Функция возвращает:

- Указатель на файл – все нормально,
- NULL – ошибка переопределения.

Закрытие файла

int **fclose**(FILE *stream);

stream - указатель на открытый файл.

Функция возвращает:
0 – файл успешно закрыт.
1 – произошла ошибка закрытия файла.

Проверка на достижение конца файла

int **feof**(FILE *stream**);**

stream - указатель на открытый файл.

Функция возвращает:

0 – если конец файла еще не достигнут.

!0 – достигнут конец файла.

Открытие текстовых файлов

Во втором параметре дополнительно указывается символ t (необязательно):

rt, wt, at, rt+, wt+, at+

Чтение из текстового файла

Форматированное чтение

int **fscanf(**FILE **stream*, const char * *format*, [*arg*] ...**)**;

Функция возвращает:

- >0 – число успешно прочитанных переменных,
- 0 – ни одна из переменных не была успешно прочитана,
- EOF – ошибка или достигнут конец файла.

Чтение из текстового файла

Чтение строки

char * **fgets(**char * *buffer*, int *maxlen*, FILE **stream***)*;

Функция возвращает:

- buffer – все нормально,
- NULL – ошибка или достигнут конец файла.

Чтение из текстового файла

Чтение символа

int **fgetc(**FILE *_stream_**)**;

Функция возвращает:

- код символа – если все нормально,
- EOF – если ошибка или достигнут конец файла.

Чтение из текстового файла

Помещение символа обратно в поток

int **ungetc(**int *c*, FILE **stream***)**;

Функция возвращает:

код символа – если все успешно,

EOF – произошла ошибка.

- FILE *f=NULL;
- f = fopen (“f.txt”,”r”);
- char c = fgetc(f);
- ungetc(‘z’,f);
- char s[20];
- fgets(s,19,f);
- printf(“%s”,s);

Запись в текстовый файл

Форматированный вывод

int **fprintf(**FILE *_stream_, const char *_format_, [_arg_] ...**)**;

Функция возвращает:

- число записанных символов – если все нормально,
- отрицательное значение – если ошибка.

Запись в текстовый файл

Запись строки

int **fputs(**const char **string*, FILE **stream***)**;

Функция возвращает:

- число записанных символов – все нормально,
- EOF – произошла ошибка.

Запись в текстовый файл

Запись символа

int **fputc(**int *c*, FILE **stream***)**;

Функция возвращает:

- код записанного символа – все нормально,
- EOF – произошла ошибка.

Открытие бинарных файлов

Во втором параметре дополнительно указывается символ b (**обязательно**):

rb, wb, ab, rb+, wb+, ab+

Чтение из бинарных файлов

size_t **fread(**void **buffer*, size_t *size*, size_t *num*,
FILE **stream***);

Функция возвращает количество прочитанных блоков. Если оно меньше num, то произошла ошибка или достигнут конец файла.

Запись в бинарный файл

size_t **fwrite(**const void **buffer*, size_t *size*, size_t *num*, FILE **stream***)**;

Функция возвращает количество записанных блоков. Если оно меньше num, то произошла ошибка.

Навигация по файлу

Чтение текущего смещения в файле:
long int **ftell(**FILE **stream***)**;

Изменение текущего смещения в файле:
int **fseek(**FILE **stream*, long int *offset*, int *origin***)**;

Возможные значения origin:
SEEK_SET (или 0) – от начала файла.
SEEK_CUR (или 1) – от текущей позиции.
SEEK_END (или 2) – от конца файла.

Функция возвращает:
0 – все нормально,
!0 – произошла ошибка.

Навигация по файлу

Перемещение к началу файла:

void **rewind(**FILE ***stream***);

Навигация по файлу

Чтение текущей позиции в файле:
int **fgetpos(**FILE ***stream*, fpos_t ***pos***)**;

Установка текущей позиции в файле:
int **fsetpos(**FILE ***stream*, const fpos_t ***pos***)**;

Функции возвращают:
0 – все успешно,
!0 – произошла ошибка.

Навигация по файлу

Структура fpos_t:

```
typedef struct fpos_t {
    long off;
    mbstate_t wstate;
} fpos_t;
```

Обработка ошибок

Получение признака ошибки:
int **ferror(**FILE ***stream***)**;
Функция возвращает ненулевое значение, если возникла ошибка.

Функция сброса ошибки:
void **clearerr(**FILE ***stream***)**;

Функция вывода сообщения об ошибке:
void **perror(**const char ***string***)**;

Буферизация

Функция очистки буфера:
int **fflush(**FILE *_stream_**)**;

Функция возвращает:
0 – все нормально.
EOF – произошла ошибка.

Функция управления буфером:
void **setbuf(**FILE *_stream_, char * _buffer_**)**;

Создает буфер размером BUFSIZ. Используется до ввода или вывода в поток.

Временные файлы

Функция создания временного файла:

FILE * **tmpfile(**void**)**;

Создает временный файл в режиме wb+. После закрытия файла, последний автоматически удаляется.

Функция генерации имени временного файла:

char * **tmpnam(**char **buffer***)**;

Удаление и переименование

Функция удаления файла:
int **remove(**const char ***filename*****)**;

Функция переименования файла:
int **rename(**const char ***fname***, const char ****nname*****)**;

Функции возвращают:

- 0 – в случае успеха,
- !0 – в противном случае.

Пример

Дан текстовый файл, содержащий целые числа (<= 10000 элементов). Переписать файл, упорядочив целые числа по возрастанию или убыванию (выбирает пользователь). Имя файла передается в параметрах командной строки.

Программа

```
int main(int argc, char *argv[])
{
  int ARR[10000], n=0, v;
  FILE *f = NULL;
  if(argc<2) {printf("Не указано имя файла!\n"); return 1;}
  f = fopen(argv[1],"r");
  if(!f) {printf("Файл не найден!\n"); return 1;}
  while(fscanf(f,"%d",&v) == 1) ARR[n++] = v;
  fclose(f);
```

Программа

```
  printf("Sort (INC-0, DEC-!0):"); scanf("%d",&v);
  if(v) qsort(ARR,n,sizeof(int),Cmp1);
    else qsort(ARR,n,sizeof(int),Cmp2);
  f = fopen(argv[1],"w");
  if(!f) {printf("Невозможно создать файл!\n"); return 1;}
  for(int i=0;i<n;i++) fprintf(f,"%d\n",ARR[i]);
  fclose(f);
  return 0;
}
```

Программа

```
int Cmp1(const void *p1,const void *p2)
{
  return *((int *)p2) - *((int *)p1);
}

int Cmp2(const void *p1,const void *p2)
{
  return *((int *)p1) - *((int *)p2);
}
```

Пример

Дан текстовый файл, содержащий в строках целые числа (максимальная длина строки – 100 символов). Вычислить сумму чисел в каждой строке и записать в новый файл. Имена файлов передаются в параметрах командной строки.

Программа

```
int main(int argc, char *argv[])
{
  FILE *f = NULL, *r = NULL;
  if(argc<3) {printf("Не указано имя файла!\n"); return 1;}
  f = fopen(argv[1],"r");
  if(!f) {printf("Файл не найден!\n"); return 1;}
  r = fopen(argv[2],"w");
  if(!r) {
    printf("Невозможно создать файл!\n");
    fclose(f); return 1;
  }
```

Программа

```
while(!feof(f)){
   char str[100] = “”;
   fgets(str,100,f);
   if(str[0]==0) continue;
   char *ptr = strtok(str," \t");
   if(!ptr) continue;
   int sum = atoi(ptr);
   while((ptr = strtok(NULL," \t")) != NULL)
     sum += atoi(ptr);
   fprintf(r,"%d\n",sum);
 }
 fclose(f); fclose(r);
 return 0;
}
```

Пример

Дан бинарный файл, содержащий вещественные числа. Найти максимум и минимум, поменять их в файле местами. Имя файла указывается в параметрах командной строки.

Программа

```
int main(int argc, char *argv[])
{
  FILE *f = NULL;
  if(argc < 2) {printf("Не указано имя файла!\n"); return 1;}
  f = fopen(argv[1],"rb+");
  if(!f) {printf("Файл не найден!\n"); return 1;}
  double max, min, val;
  int imax = 0, imin = 0, i = 1;
  fread(&val,sizeof(double),1,f);
  max = min = val;
```

Программа

```
  while(!feof(f)){
    fread(&val,sizeof(double),1,f);
    if(val > max) {max = val; imax = i;}
    if(val < min) {min = val; imin = i;}
    i++;
  }
  fseek(f,imax*sizeof(double),0);
  fwrite(&min,sizeof(double),1,f);
  fseek(f,imin*sizeof(double),0);
  fwrite(&max,sizeof(double),1,f);
  fclose(f); return 0;
}
```

Пример

Дан бинарный файл, содержащий записи со следующими полями:

- ФИО студента (строка 30 символов),
- Курс (целое число)
- Средний балл (вещественное число).

Переписать файл, упорядочив записи по курсу, а внутри курса – по фамилии. Имя файла передается в параметрах командной строки.

Программа

```
typedef struct {
  char fio[30]; int kurs;  double ball;
} STUDENT;

int Cmp(const void *p1, const void *p2)
{
  STUDENT *s1 = (STUDENT *)p1, *s2 = (STUDENT *)p2;
  if(s1->kurs != s2->kurs) return s1->kurs - s2->kurs;
  return strcmp(s1->fio,s2->fio);
}
```

Программа

```
int main(int argc, char *argv[])
{
 if(argc < 2) {printf("Не указано имя файла!\n"); return 1;}
 FILE *f = fopen(argv[1],"rb+");
 if(!f) {printf("Файл не найден!\n"); return 1;}
 long int num = 0;
 fseek(f,0,2); num = ftell(f); rewind(f);
 if(num%sizeof(STUDENT)) {
   printf("Invalid file!\n"); fclose(f); return 1;
 }
 num/=sizeof(STUDENT);
 STUDENT Arr[num];
```

Программа

```
  fread(Arr,sizeof(STUDENT),num,f);
  rewind(f);
  qsort(Arr,num,sizeof(STUDENT),Cmp);
  fwrite(Arr,num,sizeof(STUDENT),f);
  fclose(f);
  return 0;
}
```