Лекция 18

Работа с индикатором позиции файла

Для каждого файла после его открытия определяется индикатор позиции, который указывает на смещение от начала файла в байтах.

Тип индикатора позиции определяется как

```
typedef long fpos_t
```

Для работы с индикатором позиции применяются следующие функции:

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

Устанавливает индикатор позиции на начало файла. Индикаторы ошибки и конца файла сбрасываются.

```
int fseek(FILE* stream, long offset, int mode);
```

Сдвигает индикатор позиции файла на **`offset`** байт. Параметр **`mode`** определяет режим сдвига и может принимать следующие значения:

`SEEK_SET`– от начала файла;
`SEEK_CUR` – от текущей позиции;
`SEEK_END` – от конца файла

При работе с текстовым потоком должны использоваться только следующие комбинации значений параметра:

mode	offset
`SEEK_SET`	0 или `ftell`
`SEEK_CUR`	0
`SEEK_END`	0

```
long ftell(FILE* stream);
```

В случае успеха возвращает текущую позицию файла, иначе возвращает -1.

```
int fgetpos(FILE* stream, fpos_t* pos);
```

Работает аналогично **`ftell`**, но возвращает текущую позицию в **`pos`**.

Функция **`fseek`** возвращает 0 в случае успешного завершения, так что выражение

```
if (!fseek(f1, 0, SEEK_END)) filesize=ftell(f1)
```

позволяет получить информацию о длине файла в байтах.

int fsetpos(FILE* stream, const fpos_t* pos);

Устанавливает индикатор позиции файла в позицию, на которую указывает **pos**. Индикатор конца файла сбрасывается. Код возврата: удача – 0, неудача – не 0.

```
printf("Input an index: ");
scanf("%u", &i);
//устанавливает указатель на нужную запись
fseek(in, i*sizeof(struct emp),SEEK_SET);
if(!fread(&s,sizeof(struct emp),1,in))
{printf("The wrong index.\n");
 return;}
printf("\tcode = %d name = %s\n",s.code, s.name);

// сортировка справочника
int SortSpr(char *filename)
{struct record_spr
    {
        int shifr;
        char name[20];
    };
 record_spr z1,z2;
 long k, i, j;
 long filesize;
 FILE* f1;
 cout<<"сортировка справочника"<<endl;
 if(!(f1=fopen(filename,"r+b")))
 {
    cout << "Open spr file failed\n";
    return 1;
 }
 if (!fseek(f1, 0, SEEK_END))
  filesize = ftell(f1);
 k=filesize / sizeof z1;
 rewind(f1);
 for (i=k-1; i>0; i--)
  for (j=0; j<i; j++)
  {
    fseek(f1,(sizeof z1)*j,SEEK_SET);
    fread(&z1, sizeof z1, 1, f1);
```

```
      fread(&z2, sizeof z1, 1, f1);
      if (z1.shifr>z2.shifr)
      {
        fseek(f1,(sizeof z1)*j, SEEK_SET);
        fwrite(&z2, sizeof z1, 1, f1);
        fwrite(&z1, sizeof z1, 1, f1);
      }
   }
 fclose(f1);
 return 0;
}
```

Дополнительные функции

`int remove(const char* filename);`
Удаляет файл с именем **`filename`**. В случае удачи возвращает 0, иначе возвращает не 0.

`int rename(const char* old_filename, const char* new_filename);`
Переименовывает файл. В случае удачи возвращает 0, иначе возвращает не 0.
`FILE* tmpfile(void);`
Создает временный файл в режиме w+b. В случае неудачи возвращает NULL.

`char* tmpname(char* str);`
Возвращает имя временного файла. Максимальное число имен равно (для Windows) 32767, максимальная длина имени – 255. В случае удачи возвращает str, иначе возвращает NULL.

Примеры

```
int renFile(char *filename)
{
 if(rename(filename, "C:\\emp.txt"))
 {
    printf("There is no such a file.\n");
    return 0;
 }
 printf("The file was renamed.\n");
 return 1;
}
```

```
int remFile(char *filename)
{
     if(remove(filename))
     {
          printf("There is no such a file.\n");
          return 0;
     }
     printf("The file was deleted.\n");
     return 1;
}
```

ПРИМЕРЫ

```
#include <stdio.h>
#include <iostream>
#include <string.h>
#include <limits.h> //константы INT_MAX
using namespace std;
// Слияние двух упорядоченных по возрастанию
// текстовых файлов целых чисел
int main()
{
     FILE* f1; FILE* f2; FILE* fout;
     int n1,n2,nnnn;
     if(!(f1 = fopen("f1.txt", "r")))
     {    printf("Open f1 file failed.\n");
          return 1;
     }
     if(!(f2 = fopen("f2.txt", "r")))
     {    printf("Open f2 file failed.\n");
          return 1;
     }
     if(!(fout = fopen("fout.txt", "w")))
     {    printf("Open fout file failed.\n");
          return 1;
     }
     fscanf(f1,"%d",&n1);
     if (feof(f1))
          n1=INT_MAX;
     fscanf(f2,"%d",&n2);
     if (feof(f2))
          n2=INT_MAX;
     while((n1!=INT_MAX) || (n2!=INT_MAX))
     {    if (n1<n2)
```

```
            {    fprintf(fout,"%d ",n1);
                 fscanf(f1,"%d",&n1);
                 if (feof(f1)) n1=INT_MAX;
            }
            else
            {    fprintf(fout,"%d ",n2);
                 fscanf(f2,"%d",&n2);
                 if (feof(f2)) n2=INT_MAX;
            }
      }
      fclose(f1);
      fclose(f2);
      fclose(fout);
      return 0;
}
```

Пример

```
// В первой строке текстового файла записаны кол-во строк и
// столбцов матрицы
// В каждой следующей - по одной строке матрицы.
// В выходной файл записать матрицу и суммы элементов каждой
// строки в виде a11+a12+...+a1n=сумма1

#include <stdio.h>
#include <iostream>
#include <string.h>
using namespace std;
int main()
{
      setlocale(LC_ALL,".1251");
      FILE* in=NULL;//Для избежания проверок в Debug
                    // В режиме Release NULL не нужен
      FILE* out=NULL;
      int m, n, i, j, max;
      if(!(in = fopen("in1.txt", "r")))
      {
           printf("Open in file ailed.\n");
           return 1;
      }
      fscanf(in, "%d%d%", &m, &n);
      printf("m = %d n = %d \n", m,n);
      // матрица динамическая
      int **a;
       a=new int *[m]; //массив указателей
```

```
    for (i=0;i<m;i++)
        a[i]= new int [n];
    for (i=0;i<m;i++)
        for (j=0;j<n;j++)
            if (!feof(in))
                fscanf(in, "%d",&a[i][j]);
            else
            {
                printf("error\n");
                return 1;
            }
    fclose(in);

    if(!(out = fopen("out2.txt", "w")))
    {
        printf("Open out file failed.\n");
        return 1;
    }
    int sum;
    for (i=0;i<m;i++)
    {
        sum=0;
        for (j=0;j<n;j++)
        {
            sum+=a[i][j];
            fprintf(out,"%d %s",a[i][j],(j==n-1?"= ":"+ "));
        }
        fprintf(out, "%d \n", sum);
    }
    fclose(out);
    return 0;
}
```

Пример обработки файла блоками

Пусть имеется файл с записями следующей структуры:

```
struct TPrice {
   int goods;  // код товара, не может быть равен нулю
   int seller; // код продавца, не равен нулю
   int price;  // продажная цена товара
} p1,p2;
```

Требуется для каждого товара определить его минимальную продажную

цену, а также продавца (или одного из продавцов), соответствующего этой цене. Результаты работы перенести в новый файл. Эта типичная задача, называемая задачей группировки, может быть решена за один проход, если в исходном файле записи, соответствующие одному коду товара, объединены в компактную группу, т.е. идут подряд и не перемешиваются с кодами других товаров.

Это, в свою очередь, достигается, если файл отсортирован по коду товара, или если для его создания использовалась следующая программа (в ней опущена проверка на уникальность кода товара, которую предлагается выполнить самостоятельно):

```
void CreateFile(char *filename)
{if((f1=fopen(filename,"wt"))!=NULL)
{
 while (true)
 {
  cout<<"Enter goods code (0-end)" << endl;
  cin >> p1.goods;
  if (p1.goods == 0) break;
  while (true)
  {
     cout<<"Enter seller code for goods"
     <<p1.goods<<" (0 - end)"<< endl;
      cin >> p1.seller;
      if (p1.seller == 0) break;
      cout << "Enter price for seller "
      << p1.seller << " and goods " <<
      p1.goods << endl;
      cin >> p1.price;
      fwrite(&p1, sizeof p1, 1, f1);
  }
 }
 fclose(f1);
}
}

void EndGroup(int c_goods, int c_seller, int c_price)
{
  p2.seller = c_seller;
  p2.goods  = c_goods;
  p2.price  = c_price;
  fwrite(&p2, sizeof p2, 1, f2);
```

```
  cout << c_goods << " " <<
                  c_seller<<" "<<c_price<<endl;
}

void GroupFile(const char  *inputname,
                      const char *outputname)
{ int c_goods = 0, c_seller,  c_price;
  if ((f1 = fopen(inputname, "rt"))!= NULL)
  {
   if ((f2 = fopen(outputname,"wt"))!= NULL)
   {
    fread(&p1, sizeof p1, 1, f1);
    while (!feof(f1))
    {
     if (p1.goods != c_goods)
     {
       if (c_goods != 0)
          EndGroup(c_goods, c_seller, c_price);
       c_goods = p1.goods;
       c_price = 0;
     }
     if ((c_price==0)||(c_price > p1.price))
     {
       c_seller=p1.seller;
       c_price=p1.price;
     }
     fread(&p1, sizeof p1, 1, f1);
    }
    EndGroup(c_goods, c_seller, c_price);
    fclose(f2);
  }
  fclose(f1);
 }
}
```