ФАЙЛОВЕ

Изготвил:
гл.ас. д-р. Нора Ангелова

Файлове

- Редица от байтове с номерация, която започва от 0.

ВХОДНО-ИЗХОДНИ ОПЕРАЦИИ

- **Входни операции** – потокът предава данни от файл към оперативната памет.

файл  оперативна памет

- **Изходни операции** – потокът предава данни от оперативната памет към файл.

файл  оперативна памет

* *Устройство може да бъде клавиатура, диск и др.*

Поток

- **Указатели**

ifstream или istream - **get** указател, който реферира елемента, който ще се прочете при следващата входна операция.

ofstream или ostream - **put** указател, който реферира мястото, където ще се запише следващият елемент.

Работа с файлове

- Включване на библиотека за работа с файлове

```
#include <fstream>
```

- Декларация
- Отваряне
- Четене/Писане
- Затваряне

ФАЙЛОВЕ

- **Декларация на файлове:**

- за извличане (четене)

```
ifstream iFileName;
```

- за вмъкване (писане)

```
ofstream oFileName;
```

- за извличане и вмъкване

```
fstream ioFileName;
```

ФАЙЛОВЕ

- **Отваряне на файл**

```
open(<име_файл>, <режим_на_работа>{|<друг_режим>});

fstream <обект>;
<обект>.open(<име_файл>, <режим_на_работа>{|<друг_режим>});

ИЛИ

fstream <обект>(<име_файл>, <режим_на_работа>{|<друг_режим>});

Пример:
fstream file(<име_файл>, <режим_на_работа>{|<друг_режим>});

// Проверка дали отварянето е успешно
if (!file) {
  cerr << "File couldn’t be opened!\n";
  return 1;
}
```

ФАЙЛОВЕ

- **Отваряне на файл**

```
fileName.open("file", режим за достъп);
```

- за извличане (четене)

```
iFileName.open("file", ios::in);
```

- за вмъкване (писане)

```
oFileName.open("file", ios::out);
```

- за извличане и вмъкване

```
ioFileName.open("file", ios::in|ios::out);
```

Режим за достъп

ios::in (default for ifstream)	Отваря файл за извличане.
ios::out (default for ofstream)	Отваряне на файл за вмъкване. Допуска се вмъкване на произволни места във файла. Ако файлът съществува, съдържанието се изтрива.
ios::app	Отваря за вмъкване и установява указателя put в края на файла.
ios::ate	Отваря за вмъкване и установява указателя put в края на файла. Допуска вмъкване на произволни места.
ios::trunc	Ако файлът съществува, съдържанието се изтрива.
ios::binary	Превключва режима от текстов в двоичен.
ios::nocreate	Отваря за вмъкване само ако файлът с указаното име съществува.
ios::noreplace	Отваря за вмъкване само ако файлът с указаното име не съществува.

ФАЙЛОВЕ

- ios::in|ios::ate !!!
  Комбинацията ios::in | ios::ate установява put и get указателите в края на файла.
  Съществуват реализации, където опитите за извличане след подходящо позициониране на get указателя са успешни, но опитите за вмъкване са неуспешни.

- ios::in|ios::app !!!
  Комбинацията ios::in|ios::app зависи от реализацията. При някои реализации не отваря файла. При други отваря файла и позиционира put и get в началото му. Опитите за извличане след подходящо позициониране на get указателя са успешни. Операциите за вмъкване са само в края на файла.

ФАЙЛОВЕ

- **Затваряне на файл**

`close()` - затваря файла прикрепен към потока.

Член-функция на класовете fstream, ifstream, ofstream.

- **`istream& seekg`**`(streamoff p, ios::seekdir r)` - премества get указателя, с p байта относно режима на позициониране r (`ios::beg,ios::end,ios::cur`).
- **`ostream& seekp`**`(streamoff p, ios::seekdir r)` - премества put указателя, с p байта относно относно режима на позициониране r (`ios::beg,ios::end,ios::cur`).

** streamoff – отместване в байтове*

ФАЙЛОВЕ

- Позициониране на **get** и **put** указателите.

istream& seekg(streamoff p, ios::seekdir r) - премества get указателя, с p байта относно режима на позициониране r (ios::beg,ios::end,ios::cur).

ostream& seekp(streamoff p, ios::seekdir r) - премества put указателя, с p байта относно относно режима на позициониране r (ios::beg,ios::end,ios::cur).

** streamoff – отместване в байтове*

ФАЙЛОВЕ

- `streampos tellg()` - връща текущата позиция на get указателя на файла.

- `streampos tellp()` - връща текущата позиция на put указателя на файла.

Пример:

```
file.seekg(0, ios::end);
long loc = file.tellg();
```

** streampos – позиция в байтове*

ФАЙЛОВЕ

- **Видове файлове (според режима им на отваряне):**

- Текстови файлове;
- Двоични файлове;

ФАЙЛОВЕ

- **Текстови файлове:**

- Файлове с форматиран вход и изход;
- Работи се със форматиране - \n и др.;

ФАЙЛОВЕ

- **Двоични файлове:**

- Файлове с неформатиран вход и изход;
- Позволяват пряк достъп;

ФАЙЛОВЕ

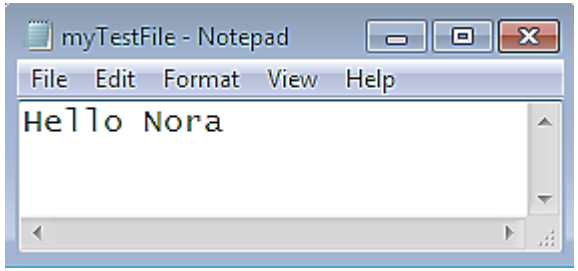
```
#include <iostream>
#include <fstream>
using namespace std;

int main() {
  ofstream myTestFile("C:/.../myTestFile.txt");

  if (!myTestFile) {
    cerr << "File couldn’t be opened!\n";
    return 1;
  }

  myTestFile << "Hello Nora";

  return 0;
}
```



ФАЙЛОВЕ

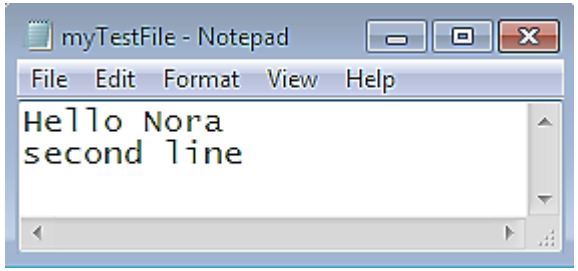
```
#include <iostream>
#include <fstream>
using namespace std;

int main() {
  ofstream myTestFile("C:/.../myTestFile.txt");

  if (!myTestFile) {
    cerr << "File couldn’t be opened!\n";
    return 1;
  }

  myTestFile << "Hello Nora \n" << "second line";

  return 0;
}
```



ФАЙЛОВЕ

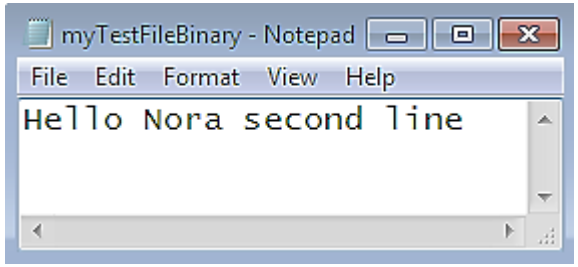
```
#include <iostream>
#include <fstream>
using namespace std;

int main() {
  ofstream myTestFileBinary("C:/.../myTestFileBinary.bin", ios::binary);

  if (!myTestFileBinary) {
    cerr << "File couldn’t be opened!\n";
    return 1;
  }

  myTestFileBinary << "Hello Nora \n" << "second line";

  return 0;
}
```



ФАЙЛОВЕ

- **Видове файлове (според достъпа до елемент):**

- Файлове с последователен достъп;
- Файлове с пряк достъп;

ФАЙЛОВЕ

- **Файлове с последователен достъп**

Компонентите на тези файлове са редица от символи завършващи с ‘\n’.

За да бъде достигнат и прочен елементът с пореден номер n, трябва последователно да бъдат прочетени всички предшестващи го елементи.

first line ‘\n’

. . . . . ‘\n’

. . . . . ‘\n’

. . . . . ‘\n’

final . . . ‘\n’

ФАЙЛОВЕ

- **Файлове с пряк достъп**

Търсеният елемент се достига директно (с адреса му), без да е необходимо да се прочетат предшестващите го елементи.

ФАЙЛОВЕ

- **Файлове с пряк достъп**

За да се достъпи директно елемент на определена позиция е необходимо всички елементи да имат една и съща дължина (големина).

Реализация

Може да се използват класове и обекти.
Данните, които записваме не се третират като символи.

Пример:
обект 1 памет
обект 2 памет
обект 3 памет
...
обект n памет

ФАЙЛОВЕ

- **Двоични файлове**

Как могат да се запишат и прочетат обекти от тип student.

```
struct student {
  char name[20];
  int fn;
};

student st1 = {"Angelova", 44394}, st2;
```

Използват се функциите read/write.

```
oFileName.write((char*)&st1, sizeof(student));
iFileName.read((char*)&st2, sizeof(student));
```

ФАЙЛОВЕ

```
struct student {
  double result;
  int fn;
};
student st1 = { 6.0, 44394 };

ofstream oFileName;
oFileName.open("C:/...", ios::out|ios::binary); //check

oFileName.seekp(0,ios::beg);
oFileName.write((char*)&st1, sizeof(student));
oFileName.close();

ifstream iFileName;
iFileName.open("C:/...", ios::in); // check

student st2;
iFileName.read((char*)&st2, sizeof(student));
cout << st2.fn << endl;
cout << st2.result << endl;
iFileName.close();
```

Примери

Да се напише програмент фрагмент, който записва данните за шестия студент.

ПРИМЕРИ

Пример:

```
struct student {
  double result;
  int fn;
};
student st1 = { 6.0, 44394 };

ofstream oFileName;
oFileName.open("C:/...", ios::out|ios::binary); // check

oFileName.seekp(5 * sizeof(student));
oFileName.write((char*)&st1, sizeof(student));
oFileName.close();
```

ПРИМЕРИ

Пример:

```
ifstream fin("file1", ios::in);
ofstream fout("file2", ios::out);
```

- Какъв е резултатът?

```
while (fin.get(ch)) fout << ch;
while (fin.get(ch)) fout.put(ch);
```

Файлът се копира дословно.

- Какъв е резултатът?

```
while (fin >> ch)) fout << ch;
```

Не се копират интервали, нов ред, табулация.

ПРИМЕРИ

Файл:

abcd

12345

```
fstream fileName;
fileName.open("C:/…");

char c1;
fileName.get(c1);
cout << c1;
```

Резултат:

a

Примери

Файл:

abcd

12345

```
fstream fileName;
fileName.open("C:/…");

char c1;
while (fileName.get(c1)) { // Използва се състоянието на потока
  cout << c1;
}
```

Резултат:

abcd

12345

Примери

Файл:

abcd

12345

```
fstream fileName;
fileName.open("C:/…");

char c1;
while (fileName >> c1) {
  cout << c1;
}
```

Резултат:

abcd12345

ПРИМЕРИ

Файл:
abcd
12345

```
fstream fileName;
fileName.open("C:/…");

char c1;
while (fileName.get(c1)) {
  cout << c1;
}

char c2;
while (fileName.get(c2)) {
  cout << c2;
}
```

Резултат:
abcd
12345

ПРИМЕРИ

Файл:
abcd
12345

```
fstream fileName;
fileName.open("C:/…");

char c1;
while (fileName.get(c1)) {
  cout << c1;
}

fileName.clear();
fileName.seekg(0, ios::beg);

char c2;
while (fileName.get(c2)) {
  cout << c2;
}
```

Резултат:
abcd
12345abcd
12345

ПРИМЕРИ

Файл:

abcd

12345

```
fstream fileName;
fileName.open("C:/…");

char str[10];

fileName.get(str, 10, '\n');
cout << str;
```

Резултат:

abcd

ПРИМЕРИ

Файл:

abcd

12345

```
fstream fileName;
fileName.open("C:/…");

char str[10];
fileName.get(str, 10, '\n');
cout << str;


char str2[10];
fileName.get(str2, 10, '\n');
cout << str2;
```

Резултат:

abcd

ПРИМЕРИ

Файл:
abcd
12345

```
fstream fileName;
fileName.open("C:/…");

char str[10];

fileName.get(str, 10, '\n');
cout << str;

fileName.get();

char str2[10];

fileName.get(str2, 10, '\n');
cout << str2;
```

Резултат:
abcd12345

ПРИМЕРИ

Файл:

abcd

12345

```
fstream fileName;
fileName.open("C:/…");

char str[10];
fileName.getline(str, 10, '\n');
cout << str;

char str2[10];
fileName.get(str2, 10, '\n');
cout << str2;
```

Резултат:

abcd12345

ПРИМЕРИ

Файл:

abcd

12345

```
fstream fileName;
fileName.open("C:/…");

fileName.write("789", 3);
```

Резултат:

789d

12345

ПРИМЕРИ

Файл:

abcd

12345

```
fstream fileName;
fileName.open("C:/…", ios::app);

fileName.write("789", 3);
```

Резултат:

abcd

12345789

ПРИМЕРИ

```
ofstream file("clients.dat", ios::out);
if (!file) {
  cerr << "File couldn’t be opened!\n";
  return 1;
}

int account;
char name[16];
float balance;
while (cin >> account >> name >> balance) {
  file << account << " " << name << " " << balance << '\n';
  cout << '?';
}
```

```
// Ctrl+Z
```

ФАЙЛОВЕ

- Игнориране на символи от потока

```
istream& ignore (streamsize n = 1, int delim = EOF);
```

Извлича n символа от потока и ги игнорира.
Спира при достигане на n или delim.

ДИНАМИЧНО ЗАДЕЛЕНЯТЕ НА ПАМЕТ

```
struct student {
  int fn;
  char * name;
};
. . .
student st1;
st1.fn = 44394;
st1.name = new char[20];
. . .

ofstream oFileName;
oFileName.open("C:/…", ios::out|ios::binary);  // check

oFileName.seekp(0,ios::beg);
oFileName.write((char*)&st1, sizeof(student)); // int + char* !!!
oFileName.close();
```

Динамично заделеняте на памет

- При стандартни операции се записват член-данни от тип int, char*.
- Динамично заделената памет съществува до нейното изтриване или до приключване на изпълнението на програмата.
- Директен прочит на член-данните ще изведе цялото съдържание.
- След спиране и стартиране на програмата, член-данната за име няма да бъде достъпна – динамичната памет е изтрита.

- Как да запишем динамично заделен масив (низ)?

ДИНАМИЧНО ЗАДЕЛЕНЯТЕ НА ПАМЕТ

- Как да запишем динамично заделен низ (аналогично за масив)?

```
char * name = new char[20]; // стойността може да бъде въведена от клавиатурата
strcpy(name, "nora angelova");

ofstream oFileName;
oFileName.open("C:/... ", ios::out|ios::binary);  // check

oFileName.seekp(0,ios::beg);
oFileName.write(name, strlen(name));
oFileName.close();

// Прочита се записаният низ
ifstream iFileName;
iFileName.open("C:/... ", ios::in);              // check

char nameResult[100];
int strLength = strlen(name);

iFileName.read(nameResult, strlen(name));

// Добавяме детерминираща 0
nameResult[strLength] = '\0';
```

** Как да разберем колко е големината на пате при повторно стартиране на програмата?*

ДИНАМИЧНО ЗАДЕЛЕНЯТЕ НА ПАМЕТ

- Как да запишем динамично заделен низ ?

Вариант 2

```
char * name = new char[20]; // стойността може да бъде въведена от клавиатурата
strcpy(name, "nora angelova");

ofstream oFileName;
oFileName.open("C:/... ", ios::out|ios::binary);  // check

oFileName.seekp(0,ios::beg);
oFileName.write(name, strlen(name) + 1);
oFileName.close();

// Прочита се записаният низ
ifstream iFileName;
iFileName.open("C:/... ", ios::in);              // check

char nameResult[100];
int strLength = strlen(name);

iFileName.read(nameResult, strlen(name) + 1);
```

** Как да разберем колко е големината на пате при повторно стартиране на програмата?*

ДИНАМИЧНО ЗАДЕЛЕНЯТЕ НА ПАМЕТ

- Как да разберем колко е големината на name при повторно стартиране на програмата?

```
char * name = new char[20];
strcpy(name, "nora angelova");

ofstream oFileName;
oFileName.open("C:/... ", ios::out|ios::binary); //check

oFileName.seekp(0, ios::beg);
int strLength = strlen(name);

// Записваме дължината в началото
oFileName.write((char*)& strLength, sizeof(strLength));
oFileName.write(name, strLength);
oFileName.close();
```

ДИНАМИЧНО ЗАДЕЛЕНЯТЕ НА ПАМЕТ

- Как да разберем колко е големината на name при повторно стартиране на програмата?

```
ifstream iFileName;
iFileName.open("C:/... ", ios::in);                    // check

int lengthResult = 0;
char nameResult[100];

// Прочитаме размера
iFileName.read((char*)& lengthResult, sizeof(lengthResult));

// Прочитаме низа, може паметта да е динамично заделена
iFileName.read(nameResult, lengthResult);

// Добавяме детерминираща 0
nameResult[lengthResult] = '\0';

cout << nameResult << endl;

iFileName.close();
```

КРАЙ

?