Механизми за записване и извличане на информация (файлове)

Изготвил:
доц. д-р Нора Ангелова

ИНФОРМАЦИЯ?!

Файлове	Памет
Големи обеми	Ограничено количество
Бавни	Бързи
Персистентни	Само до приключване на програмата

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

вход за програмата

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

файл  оперативна памет

изход за програмата

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

РАБОТА С ФАЙЛОВЕ

постоянната памет  оперативна памет

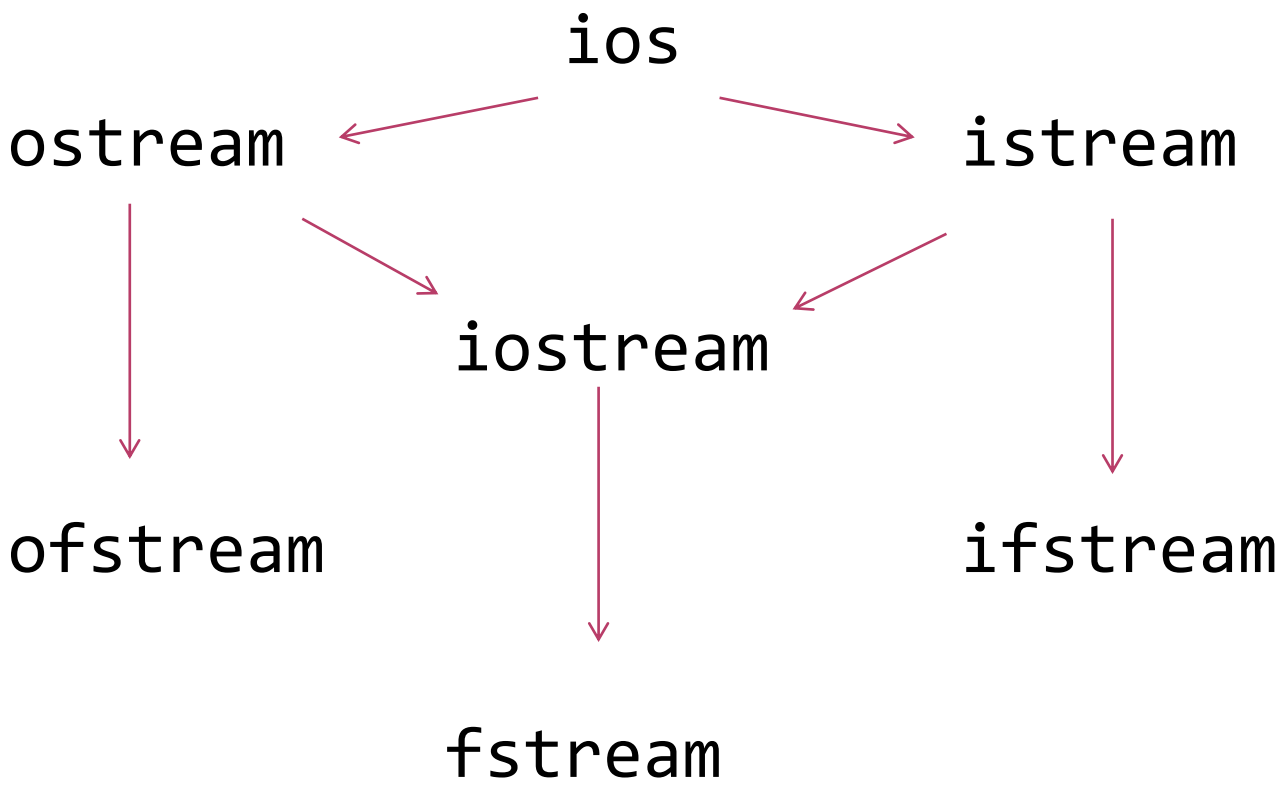
Файлови потоци

- За работа с файлове се използват файлови потоци – поток, който може да пренася информация от файл към оперативната памет и обратно

Хедър за работа с файлове - `#include<fstream>`

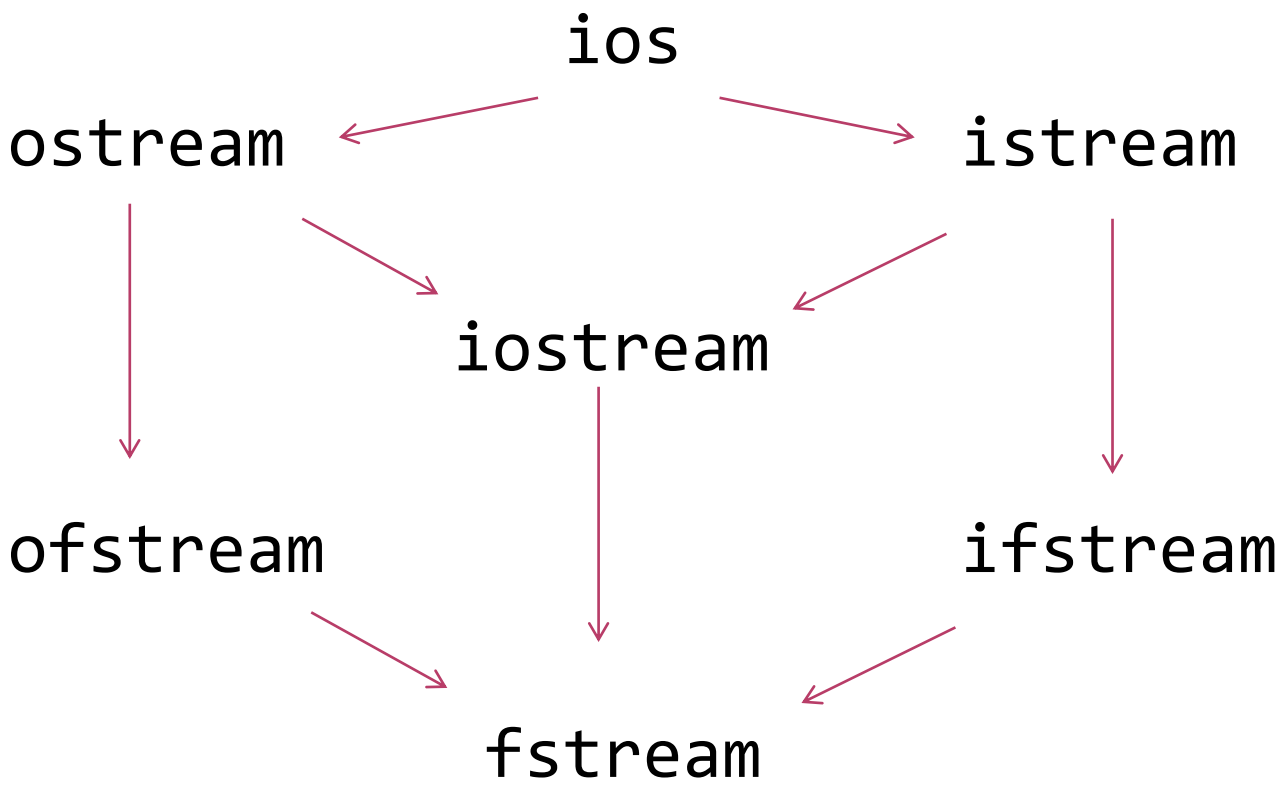
(ФАЙЛОВИ) ПОТОЦИ

- Потоците представляват обекти от даден клас. Съществуват множество от класове, които са в следната йерархия.



(Файлови) потоци

- Всички методи и оператори за работа с входно-изходните потоци(cin, cout, etc.) важат и за работа с файлове



ФАЙЛОВЕ

- **Декларация на файл**

- за извличане (четене)

```
std::ifstream iFileName;
```

- за вмъкване (писане)

```
std::ofstream oFileName;
```

- за извличане и вмъкване

```
std::fstream ioFileName;
```

ФАЙЛОВЕ

- **Отваряне на файл**

- с използване на open

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

Синтаксис
<std_fstream> <име_файл>;
<име_файл>.open(<име_файл_във_файловата_система>,
[<*режим_на_работа*>{|<*друг_режим*>}$_{опц}$]$_{опц}$);

<std_fstream> ::= std::ifstream | std::ofstream | std::fstream

Пример
```
std::ofstream oFileName;
oFileName.open("text.txt");
```

ФАЙЛОВЕ

- **Отваряне на файл**

- в самата дефиниция

Синтаксис

```
<std_stream> <име_файл>(<име_файл_във_файловата_система>,
[<режим_на_работа>{|<друг_режим>}опц]опц);

<std_fstream> ::= std::ifstream | std::ofstream | std::fstream
```

Пример

```
std::ifstream iFileName("text.txt");
```

ФАЙЛОВЕ

- **Проверка дали файл е отворен успешно**

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

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

- **Отваряне на файл**

```
fileName.open("file.txt", [<режим_на_работа>{|<друг_режим>}опц]опц);
```

- за извличане (четене)

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

- за вмъкване (писане)

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

- за извличане и вмъкване

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

** Съдържанието няма да бъде изтрито*

Позиция в поток

- **Указатели(припомняне от потоци)**

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.

Позиция във поток

- 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*

Позиция в поток

- `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); // Позиционира указателя в края на файла
long loc = file.tellg(); // Големината на съдържанието
```

** streampos – позиция в байтове*

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.

За да бъде достигнат и прочетен елемент 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.

Въпрос: Как може да се постигне този резултат?!?

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(структури) и обекти. Данните, които записваме не се третират като символи.

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

ФАЙЛОВЕ

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

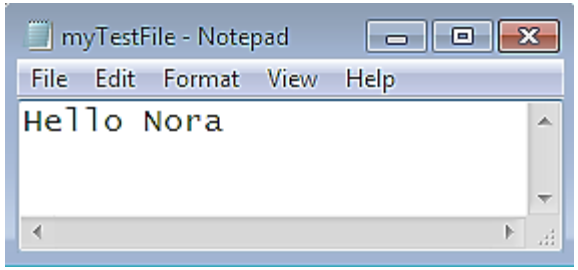
- Текстови файлове;
- Двоични файлове;

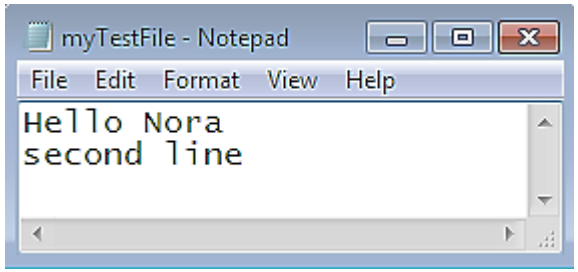
Файлове

- **Текстови файлове:**
  - Файлове с форматиран вход и изход;
  - Работи се със форматиране – `\n, endl` и др.;
  - Подходящи за последователно записване и извличане на информация

ФАЙЛОВЕ

```
…
ofstream outFile("C:/.../myTestFile.txt");
outFile << "Hello Nora";
outFile << endl << "second line";
```





Файлове

- **Текстови файлове**
  - Последователно четене и писане
  - Как се добавя елемент на дадена позиция?
  - Как се достъпва елемент по индекс?
  - Как се изтрива елемент?

ФАЙЛОВЕ

- **Двоични файлове:**

- Файлове с неформатиран вход и изход (байтове);

- Позволяват пряк достъп – може да записваме и четем байтове;

Файлове

- **Двоични файлове**
  - Пряк достъп до елемент (на отстояние определен брой байтове)

Файлове

- **Четене и писане във файлове**
  - <<, >>
  - put, get
  - read, write
  - други

Файлове

- **Състояние на поток – важи и за файлови потоци**

Файлове

- **Затваряне на файл**

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

ФАЙЛОВЕ

- **Записване на обект във файл с пряк достъп**

```
struct Person {
  char name[50];
  int age;
  char phone[24];
};

Person me = {"Nora", 32, "364-2534"};

// Отваряне на файл в двоичен формат за писане
std::ofstream outBinaryFile("...", std::ios::out|std::ios::binary);

// Записване на обект във файл (работи се с байтове)
outBinaryFile.write((char *)&me, sizeof(me));
```

ФАЙЛОВЕ

- **Записване и извличане на обект при файлове с пряк достъп**

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

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

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

std::ifstream iFileName;
iFileName.open("file.txt", std::ios::in); // check

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

ПРИМЕРИ

Даден е двоичен файл с пряк достъп.
Да се напише програмент фрагмент, който записва данните за шестия студент.

ПРИМЕРИ

Пример:

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

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

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

Примери

Пример:

```
std::ifstream fin("…", std::ios::in);
std::ofstream fout("…", std::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

```
std::fstream fileName;
fileName.open("test.txt");

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

Резултат:

a

Примери

Файл:

abcd

12345

```
std::fstream fileName;
fileName.open("test.txt");

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

Резултат:

abcd

12345

Примери

Файл:

abcd

12345

```
std::fstream fileName;
fileName.open("test.txt");

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

Резултат:

abcd12345

ПРИМЕРИ

Файл:
abcd
12345

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

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

// Състояние EOF
char c2;
while (fileName.get(c2)) {
  std::cout << c2;
}
```

Резултат:
abcd
12345

ПРИМЕРИ

Файл:
abcd
12345

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

char c1;
while (fileName.get(c1)) {
  std::cout << c1;
}
// Изчиства потока, той е в състояние EOF
fileName.clear();
fileName.seekg(0, std::ios::beg);

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

Резултат:
abcd
12345abcd
12345

ПРИМЕРИ

Файл:

abcd

12345

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

char str[10];

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

Резултат:

abcd

ПРИМЕРИ

Файл:

abcd

12345

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

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

// Започва от знак за нов ред и спира там – str2 е празен низ
char str2[10];
fileName.get(str2, 10, '\n');
std::cout << str2;
```

Резултат:

abcd

ПРИМЕРИ

Файл:
abcd
12345

```
std::fstream fileName;
fileName.open("…");

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

// Прочита знака за нов ред
fileName.get();

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

Резултат:
abcd12345

ПРИМЕРИ

Файл:

abcd

12345

```
std::fstream fileName;
fileName.open("…");

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

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

Резултат:

abcd12345

ПРИМЕРИ

Файл:

abcd

12345

```
std::fstream fileName;
fileName.open("…");

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

Резултат:

789d

12345

ПРИМЕРИ

Файл:
abcd
12345

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

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

Резултат:
abcd
12345789

ПРИМЕРИ

- Работа с файлове с последователен достъп

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

int account;
char symbol;
float balance;

// Четем от конзолата (прескачат се разделите)
while (std::cin >> account >> symbol >> balance) {
  // Записва се съдържанието във файл с определени разделите
  file << account << " " << symbol << " " << balance << '\n';
  std::cout << '?';
}


 // Изход: Ctrl+Z
```

** За низове трябва да се използва get OR getline, за да се гарантира коректността на входа*

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];
. . .

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

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

. . .
// изтриване на динамично заделената памет
```

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 заделяте на памет

- Как да запишем динамично заделен масив (низ)?
  - Ако работим с файлове с пряк достъп и записваме обект, в обекта се записва само адреса на динамично заделената памет
  - Ако записваме динамично заделени низове, те са с различна големина и не можем да имаме блокове от памет с еднакъв размер
  - Можем да записваме динамично заделени низове, но във файлове с последователен достъп

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");

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

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

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

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

oFileName.seekp(0, std::ios::beg);
oFileName.write(name, strlen(name) + 1);
oFileName.close();
// Прочита се записаният низ
ifstream iFileName;
iFileName.open("...", std::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?*
Може да четем символите до знака за край, но той може да означава друго.
Може да искаме да изчетем целия низ

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");

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

oFileName.seekp(0, std::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:/... ", std::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';

std::cout << nameResult << std::endl;

iFileName.close();
```

ВРЕМЕ ЗА ВАШИТЕ ВЪПРОСИ

?