Тема 2: Потоци и текстови файлове

1. **Потоци**

Поток (stream) - последователност от байтове данни, свързани в или изходящи от програмата

потоци за вход — Байтовете влизат от څونтроля за вход (клавиатура, файл, друга програма)

потоци за изход — Байтовете излизат от програмата и се „изливат" в څونтроля за изход (екран, файл, друга програма)

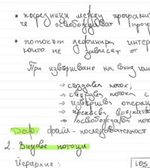
- хедърните файлове предоставят на IO операциите по такъв начин, че IO операциите работят еднакво с всяко څونтроля
- потоците предлагат интерфейс с операции върху него, които не зависят от څونтроля на IO операциите

При извеждане на вход или изход, е-то преминава:
→ създаване на поток;
→ свързване на поток с IO څونтроля;
→ извършване операции за вход/изход от څونтроля;
→ прекъсване/затваряне на поток;
→ освобождаване на поток.

Файл - последователност /масив от байтове (числа от 0 до 255)

2. **Видове потоци**

Йерархия:



3. **Интерфейс**

3.1 Потоци за изход

- Съхранява масив (буфер), в който се записват новите байтове данни, докато не се изпрати څونтрола (flush(), когато 'изчистим' буфера или когато буферът е пълен

څونтрола е тая, която извършва операциите с файла с съхраняваните в буфера байтове. Буферът е нужен, за по-бързо изпълнение на четенето/записа на файлова памет, а не такова операция.

Потокът за изход имат put указател, който сочи към поредния свободен елемент, в който може да бъде записан следващият елемент

a) форматиране изход - отчита типа на данните, когато е чете/пише

operator << (ostream &, <обект> &)
(istream)

- свързва по символен достъп обекти

б) неформатиране

- put(char ch) - ф-я чете поредния следващ елемент от څونтрола в буфера
- write(const char *str, size_t size) - записва string څونтрола и size при фиксиран размер

в) синхронизация - flush() - изчиства се автоматично на интервали и при достигане на файлов край, когато да е изпълнена и автоматично

г) позициониране в потока за изход

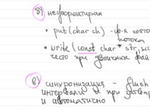
- tellp() - връща индекса на текущо е put указателя
- seekp(pos_type, ios::seekdir) - премества put указ. в څونтрола по pos-та от seekdir, който да е ios::beg - начало на څونтрола
  - ios::cur - текущата позиция
  - ios::end - края на څونтрола

Пример за изход към файл:

```
std::ofstream ofs("file.txt");
if (!ofs.is_open()) {
    // обработване грешката
}
int x = 35;
char y = 35;
ofs << x << y;
ofs.seekp(0, std::ios::cur);
ofs.put('x');
ofs.flush();
```

- инициализиране на څونтрола
- проверка дали файла е отворен (څونтрола съществува е ИС)
- премества put указ. 3 позиции назад от текущата
- записва 'x' на текущата позиция
- изтриване на څونтрола

3.2 Потоци за вход



istream и ostream имат get указател, който реферира елемента, който ще е прочетен при следващата входна операция

a) форматиран вход

operator >>

ifs >> <обект>

int n;
ifs >> n;

int x;
cin >> x; "35" - потокът ще е празен

б) неформатиране вход

- get() - чете символ, на който е get указ.; връща int - ASCII код на символа
  overloads:
  - get(char* buffer, брой символи, символ за прекъсване) - void
  - get(char* buf, брой символи, разделител) - void
- getline(буфер, размер на буфера, разделител) - void
- peek() - връща следващия символ, не чете указ.
- read(char*, size) - за четене на двоични файлове

в) позициониране

- tellg() - връща индекса на get
- seekg(pos_type, ios::seekdir) - get указателя
- sync() - изчиства буфера
- ignore() - чете указ. с един следващ void
  ignore(брой символи)
  ignore(брой символи, разделител)

- istream_iterator - ако е istream, чете от څونтрола, свързан с istream
- istreambuf_iterator - чете брой символи, или до څونтрола, свързан с istream

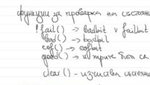
4. **Проверка на грешки**

istream f(ios::in | ios::binary);
↳ څونтрола

- ако са на новия ред, не изтриваме с '|'

① ios::in - отваряне потока за четене
② ios::out - // - запис - създаване, ако не съществува, или изтриване на предишно съдържание
③ ios::app - // - запис - създаване, ако не съществува, съдържанието се запазва
④ ios::ate - отваряне за изход и позиция put указ. в края на څونтрола
⑤ ios::app - // - НЕ изтрива файлове, за разлика от app
⑥ ios::trunc - ако файлът съществува, съдържанието се изтрива
⑦ ios::binary - двоичен режим от текстов в двоичен

5. **Състояния на потока**



eofbit - флага се вдига, с прочитат край на файла (EOF)
failbit - флага се вдига, когато е имало грешка - неизпълнена операция
- неизвестна, фатална грешка
операция не е завършена
badbit - няма фатална/сериозна операция, не е изпълнена

функции за проверка на състоянието:

fail() -> failbit v badbit
bad() -> badbit
eof() -> eofbit
good() -> all флага са 0

clear() - изчиства състоянието на потока

6. **Stringstream** - автоматично превръща данните в string

- отчита се при се чете от څونтрола

stringstream ss("ABC")
↳ initial value

A B C

ss << 2;

A B C