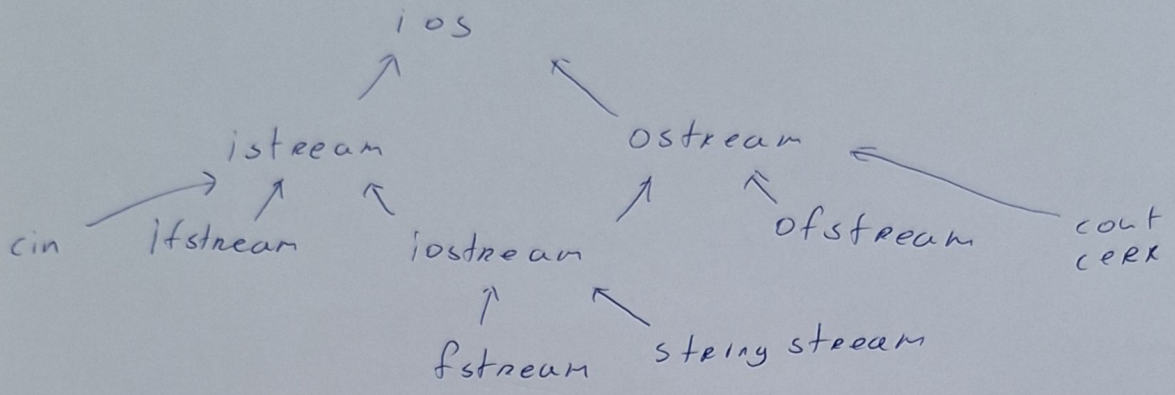
2. Поток. Стандартни потоци. Текстови файлове. Йерархия на потоците. Интерфейс на потоци. Потоци за вход/изход от файл. Режимите работа. Флагове на състоянията на потока. Позициониране във файл.

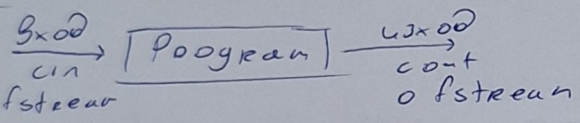
Поток.
- ~~нещо~~ наредена последователност от байтове
- с origin и destination

Йерархия на потоците



Стандартни потоци

Диаг.



- Вход
  - байтовете идват от източник за вход (клавиатура, файл, сензор, тръба, програмата)
- Изход
  - байтовете излизат от програмата и отиват във външен "приемник" (конзола, файл, тръба)
- потоците работят като посредници м/у програмистта и I/O устройствата, така освобождават програмистта от работа с I/O операции на ниско ниво
- стъпки с работа в/у поток в C++

създаване на поток → свързване с I/O устройство → извършване на I/O операции → прекъсване на връзката → освобождаване на потока

Потоци за изход: ostream
- форматирани: оператор <<

Пр: cout << 3 << 4;
- неформатирани

```
put(char ch)
write(const char* str, size_t size);
```

- работа
  а) към конзолата (обект от тип ostream)
     ostream os << <обект>;

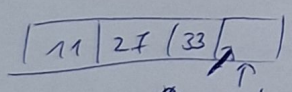
  б) към файл (обект от тип ofstream)
     ofstream ofs (<име-на-файл>, <режим-на работа>);

⚠ проверка дали потока е отворен
  if(!ofs.is_open()){
     throw std::runtime_error("test is");
  }

⚠ затварянето се извиква ф-цията close() или автоматично се извиква при край на scope-а

- синхронизиране
  .flush() - записва буфера във файла
  ⚠ close() извиква flush();

- позициониране
  имаме put указател, който сочи към настоящия адрес, към който ще се записва в паметта



put указател

ф-ции от интерфейса:
  .tellp() - показва къде в паметта се намира put указателя
  .seekp(size_t idx) - премества put указателя на idx
  .seekp(dest, offset) - премества put указателя като:
     dest - ios::begin - началото на потока
            ios::cur - настоящата позиция
            ios::end - в края на потока
     offset - с какво отместване да бъде put указателя от подадената dest.
     0 - значи същата позиция

Потоци за вход (istream)

- форматирано оператор >>

Пр: `int i;`
`cin >> i;`

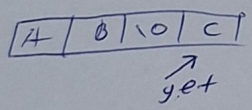
`istream cin >> <обект>;`

- неформатиран
  - `.get();` → чете 1 символ
  - `.get(buff, size)` == `.get(buff, size, '\n')`
  - `.get(buff, size, del);` - чете или size на брой байтове или докато не срещне del, всичко се записва в buff (без del)
  - `.getline(buff, size);` - чете size на брой докато не срещне нов ред, като прескача, новия ред и поставя get указателя след това
  - `.getline(buff, size, delim);`

Пр: имаме стрин

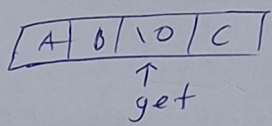
`getline(buff, 32)`

A B



`get(buff, 32);`

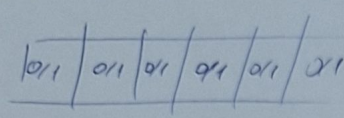
A B



- работа
  - а) към конзолата (обект от тип istream)
    `int i;`
    `cin >> i;`
  - б) към файл
    `ifstream ifs(<име-на-файл>, <режим-на-работа>)`
- синхронизация
  - като ostream
- позициониране
  - аналогично на ostream по имате get указател
    който сочи към следващия елт за четене
  - `.tellg()`
  - `.seekg(sizeof(int) * idx)`
  - `.seekg(dest, offset)`
  - `.ignore(sizeof, del)` - прескача size на брой символи или докато няма разделител
    по дефолт size = 1, del = eof();

Режими на работа

- ofstream / ifstream / fstream stream (име на файл >, <режим>)
- режима на работа е число
- в паметта:



= в зависимост от това дали битовете са вдигнати или не потока работи в режим на работа

- ако имаме няколко режима за работа можем да ги обединяваме, чрез "|" - побитово или

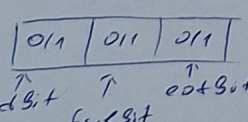
Опции:

срещу всяко число съответства име на режима в ios namespace-а

- ios::in (1) - потока е отворен за четене / извличане
- ios::out (2) - потока е отворен за писане / вмъкване. Ако файлът съществува, то съдържанието му се трие
- ios::ape (4) - отваря потока, поставя put на края, позволява местене/позициониране
- ios::app (8) - отваря потока, поставя put накрая, не позволява достъп до вече съществуващо съдържание
- ios::trunc (16) - по def, ако файла съществува изтрива го
- ios::binary (32) - отваря файла в двоичен режим на работа
- ios::_Nocreate (64) - отваря за вмъкване, само ако съществува
- ios::_Noreplace (128) - отваря за вмъкване, само ако не съществува

Състояние на потока

- всеки поток има 3 бита, които показват състоянието му



- bad() - има загуба на информация (операция I/O не е била успешна) → badbit
- fail() - последната I/O операция е невалидна → badbit | failbit
- eof() - дали сме достигнали до края → eofbit
- good() - дали всичко е наред
- clear() - автоматично сваля всички