Потоци

Задача

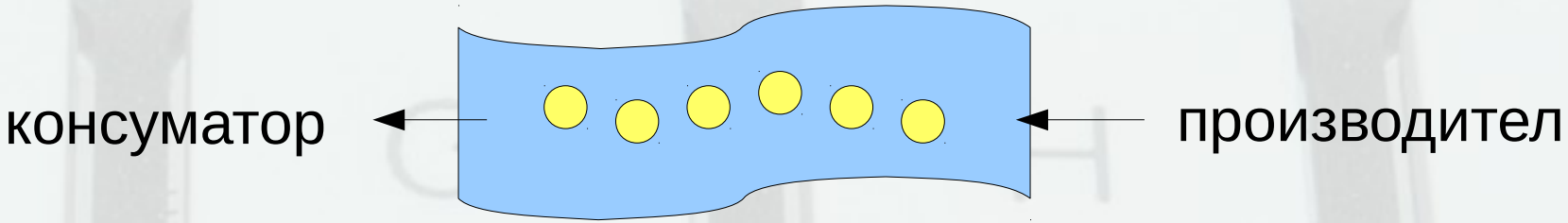
- Програма А пресмята поредица от данни
  - простите числа
  - кадри от видео клип
  - списък от постове във Facebook/Twitter
- Програма Б обработва поредица от данни
  - търси числа - близнаци
  - прави снимки на “интересни” моменти от клипа
  - събира всички постове с линк към YouTube
- Как да организираме работата на двете програми?

Задачи за дробни числа

- Да се пресметне числото $\frac{m}{n}$ с произволна точност
- Пример: $\frac{9}{7}$ = 1,285714285714...
- Да се намери сборът

$$\frac{m}{n} + 0{,}12345678910111213141516...$$

Абстракцията поток



Обектно-ориентиран подход

- cin >> number >> char >> string;
- file << student << list << tree_of_files;
- while (stream1 >> x) stream2 << f(x);

Поточна обработка

- Конвейер
- Unix pipes
  ls | grep new | wc -l
- Позволява паралелна обработка
- Файловете като потоци

Дробите като потоци от цифри

- DecimalFraction df(9, 7); int digit;
- while (df >> digit) cout << digit;
- CountingFraction cf; DigitCounter dc(5);
- do { cf >> digit; dc << digit; }
  while (dc.count() < 50);
- Да се намери n-тата позиция на съвпадение на цифри в две дроби
- Да се съберат две дроби

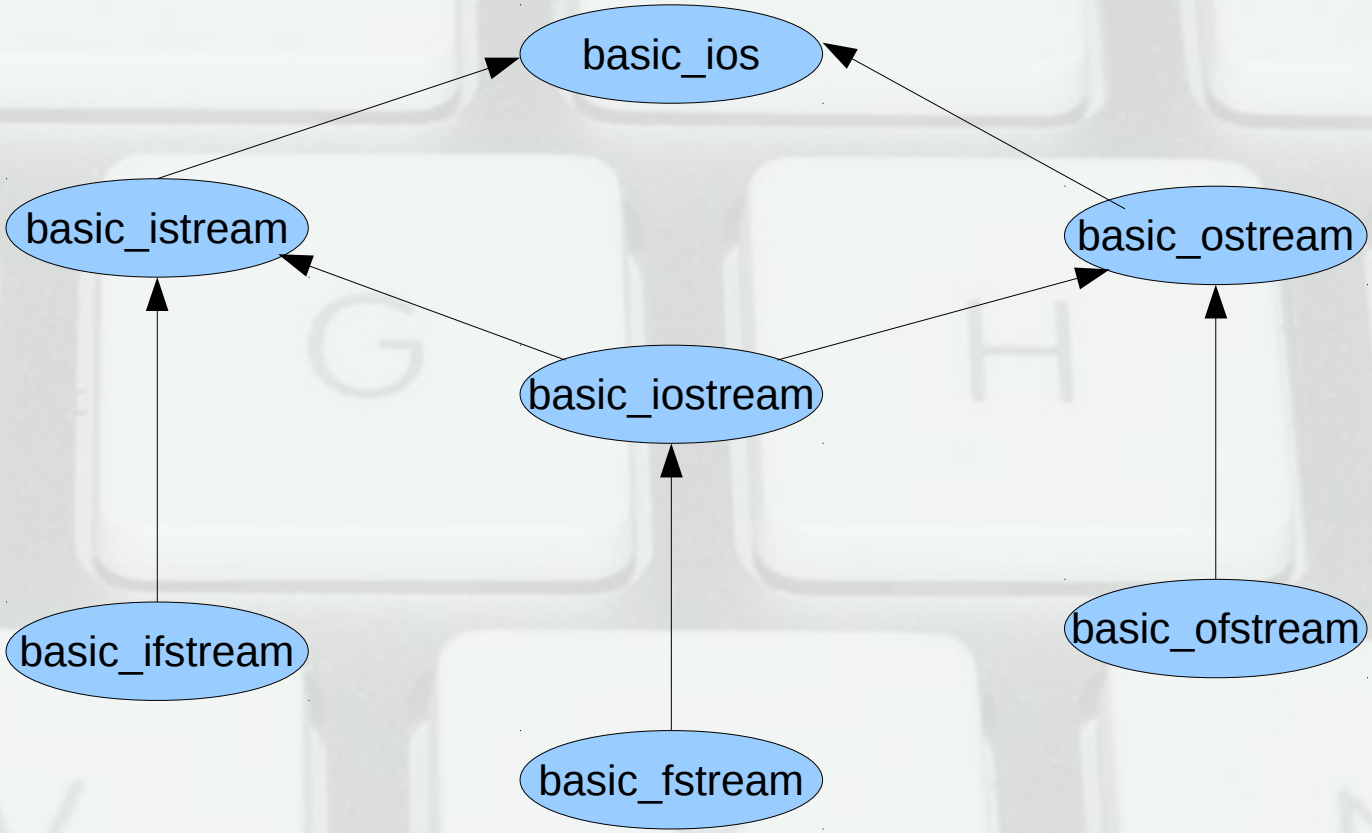
Буфериране

- Какво е?
- Защо има нужда от него?
- Кога няма нужда от него?

Стандартни потоци и пренасочване

- cout (stdout)
  - dir > filelist.txt
- cin (stdin)
  - grep password < email.txt > password.txt
- cerr (stderr)
  - move *.dat d:\ 2> errors.txt
- clog (отново stderr)

Поточна йерархия



Форматиран и неформатиран вход/изход

- Текстова и двоична информация
- ASCII (char)
- Служебни символи
- Кодиращи таблици
- Unicode (wchar_t)
- UTF-8

Функции за изход

- ostream& put(char);
- ostream& write(const char*, streamsize);
- ostream& operator<<(ostream&, T);

Функции за вход

- istream& get(char&);
- istream& get(char*,streamsize,char);
- istream& getline(char*,streamsize,char);
- streamsize gcount() const;
- istream& read(char*, streamsize);
- istream& operator>>(istream&, T&);

Още функции за вход

- int peek();
- istream& putback(char);

Операции << и >>

- cin >> a >> b >> c;
- >> и << не може да са член-функции!
  - защо?
- *friend* ostream& operator<<(ostream&, A const&);
- *friend* istream& operator>>(istream&, A&);
- Предефиниране на операторите за вход и изход

Състояние на поток

- enum iostate { goodbit = 0, eofbit = 1,
  failbit = 2, badbit = 4 }
- bool good() const;
- bool eof() const;
- bool fail() const;
- bool bad() const;

Състояние на поток

- enum iostate { goodbit = 0, eofbit = 1,
  failbit = 2, badbit = 4 }
- iostate rdstate() const;
- void clear(iostate = 0);
- operator void*() const;
- bool operator !() const;

Манипулатори

- stream << data1 << manipulator << data2;
- cout << 20 << hex << 20 << oct << 20;
- cout << scientific << 100 << fixed << 100;

Флагове за формат

- hex, oct, dec
- fixed, scientific
- left, right, internal
- fmtflags flags() const;
- setf(fmtflags flg);
- stream << setiosflags(fmtflags);

Манипулатори или функции?

- cout << setprecision(5)
  cout.precision(5)
- cout << setw(10)
  cout.width(10)
- cout << setfill('x')
  cout.fill('x')

Потребителски манипулатори

- istream& (*)(istream&);
- ostream& (*)(ostream&);

stream >> manipulator >> data
↕
manipulator(stream) >> data

stream << manipulator << data
↕
manipulator(stream) << data