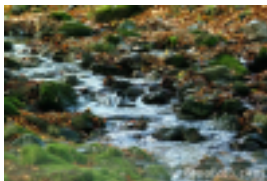


Потоци

Трифон Трифонов

Обектно-ориентирано програмиране,
спец. Компютърни науки, 1 поток,
спец. Софтуерно инженерство,
2016/17 г.

6–20 април 2017 г.



Взаимодействие на две програми

Програма А пресмята поредица от данни

- простите числа
- кадри от видео клип
- списък от постове във Facebook/Twitter

Взаимодействие на две програми

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

Взаимодействие на две програми

Програма А пресмята поредица от данни

- простите числа
- кадри от видео клип
- списък от постове във Facebook/Twitter

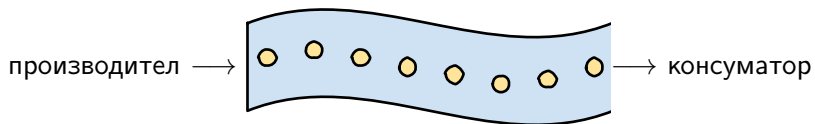
$B(A())$

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?

Абстракцията поток



Обектно-ориентиран подход

```
cin >> number >> character >> string;
```

```
file << student << list << tree;
```

```
while (stream1 >> x) stream2 << f(x);
```



Конвейерна обработка

- събирането на няколко потока в един голям поток

Конвейерна обработка

- събирането на няколко потока в един голям поток
- ефективна паралелна обработка

Конвейерна обработка

- събирането на няколко потока в един голям поток
- ефективна паралелна обработка
- саморегулиращ се механизъм

Конвейерна обработка

- събирането на няколко потока в един голям поток
- ефективна паралелна обработка
- саморегулиращ се механизъм
- **Пример:** Unix pipes

Конвейерна обработка

- събирането на няколко потока в един голям поток
- ефективна паралелна обработка
- саморегулиращ се механизъм
- **Пример:** Unix pipes
- `ls | grep new | wc -l`

Конвейерна обработка

- събирането на няколко потока в един голям поток
- ефективна паралелна обработка
- саморегулиращ се механизъм
- **Пример:** Unix pipes
- `ls | grep new | wc -l`
- Файловете като производители или консуматори на потоци

Поточен буфер

- Какво представлява буферът?

Поточен буфер

- Какво представлява буферът?
- Кога е нужен буфер?

Поточен буфер

- Какво представлява буферът?
- Кога е нужен буфер?
- Кога буферът вреди?

Поточен буфер

- Какво представлява буферът?
- Кога е нужен буфер?
- Кога буферът вреди?

H	e	l	l	o	,		w	o	r	l	d	!	\n
---	---	---	---	---	---	--	---	---	---	---	---	---	----

Стандартни потоци и пренасочване

- Стандартен изходен поток `cout` (`stdout`)
 - Пренасочване на изхода:
 - `ls > filelist.txt`

Стандартни потоци и пренасочване

- Стандартен изходен поток `cout` (`stdout`)
 - Пренасочване на изхода:
 - `ls > filelist.txt`
- Стандартен входен поток `cin` (`stdin`)
 - Пренасочване на вход и на изход:
 - `grep password < email.txt > password.txt`

Стандартни потоци и пренасочване

- Стандартен изходен поток `cout` (`stdout`)
 - Пренасочване на изхода:
 - `ls > filelist.txt`
- Стандартен входен поток `cin` (`stdin`)
 - Пренасочване на вход и на изход:
 - `grep password < email.txt > password.txt`
- Стандартен поток за грешки `cerr` (`stderr`)
 - Пренасочване на изход за грешки:
 - `mv *.dat /data 2> errors.txt`



Стандартни потоци и пренасочване

- Стандартен изходен поток `cout` (`stdout`)
 - Пренасочване на изхода:
 - `ls > filelist.txt`
- Стандартен входен поток `cin` (`stdin`)
 - Пренасочване на вход и на изход:
 - `grep password < email.txt > password.txt`
- Стандартен поток за грешки `cerr` (`stderr`)
 - Пренасочване на изход за грешки:
 - `mv *.dat /data 2> errors.txt`
- Стандартен поток за дневник `clog` (отново `stderr`)

Форматиран и неформатиран вход/изход

- Текстова и двоична информация

Форматиран и неформатиран вход/изход

- Текстова и двоична информация
- ASCII (`char`)

Форматиран и неформатиран вход/изход

- Текстова и двоична информация
- ASCII (`char`)
- Служебни символи

Форматиран и неформатиран вход/изход

- Текстова и двоична информация
- ASCII (`char`)
- Служебни символи
- Кодиращи таблици

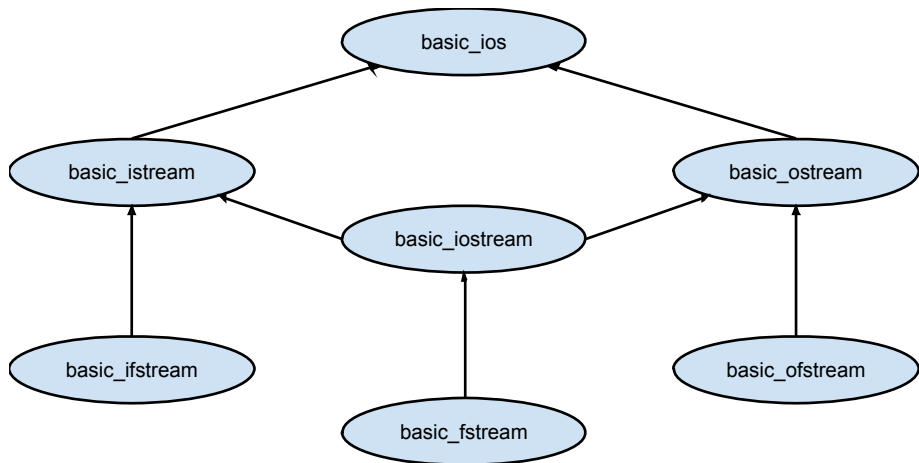
Форматиран и неформатиран вход/изход

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

Форматиран и неформатиран вход/изход

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

Поточна йерархия в C++



Изход на поток

Неформатиран изход:

```
ostream& put(char);  
ostream& write(const char*, streamsize);
```

Изход на поток

Неформатиран изход:

```
ostream& put(char);  
ostream& write(const char*, streamsize);
```

Форматиран изход:

```
ostream& operator<<(ostream&, T);
```

Вход от поток

Неформатиран вход:

```
istream& get(char&);  
istream& get(char*,streamsize,char);  
istream& getline(char*,streamsize,char);  
streamsize gcount() const;  
istream& read(char*, streamsize);
```

Вход от поток

Неформатиран вход:

```
istream& get(char&);
istream& get(char*, streamsize, char);
istream& getline(char*, streamsize, char);
streamsize gcount() const;
istream& read(char*, streamsize);
```

int x;
double y;
char z[10];
123

Форматиран вход:

```
istream& operator>>(istream&, T&);
```

z

1	2	3	0.
---	---	---	----

cin >> x;

x
123

cin >> y;

y
123.0

cin >> z;

Вход от поток

Неформатиран вход:

```
istream& get(char&);  
istream& get(char*,streamsize,char);  
istream& getline(char*,streamsize,char);  
streamsize gcount() const;  
istream& read(char*, streamsize);
```

Форматиран вход:

```
istream& operator>>(istream&, T&);
```

Допълнителни функции:

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


Низови потоци

```
#include <sstream>
```

Входен поток от низ: `istringstream`

Пример:

```
char s[] = "1 2 3";  
istringstream iss(s);  
int a, b, c;  
iss >> a >> b >> c;
```

Низови потоци

```
#include <sstream>
```

Входен поток от низ: `istringstream`

Пример:

```
char s[] = "1 2 3";  
istringstream iss(s);  
int a, b, c;  
iss >> a >> b >> c;
```

Изходен поток към низ: `ostringstream`

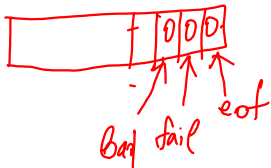
Пример:

```
ostringstream oss;  
oss << 1.2 << ' ' << 3.4;  
cout << oss.str();
```

Състояние на поток

Флагове за състояние:

iostate	goodbit	eofbit	failbit	badbit
	0	1	2	4



Състояние на поток

Флагове за състояние:

iostate	goodbit	eofbit	failbit	badbit
	0	1	2	4



Селектори:

```
bool good() const; bool eof() const;
```

```
bool fail() const; bool bad() const;
```

```
iostate rdstate() const;
```

Мутатор:

```
void clear(iostate = 0);
```

Примери:

```
if (cin.rdstate() & (eofbit | badbit)) ...
```

```
cin.clear(failbit);
```

```
if (cin) ...
```

```
if (!cin) ...
```



Потокови манипулатори

```
#include<iomanip>
```

```
stream << data1 << manipulator << data2;
```

- Манипулатори за изход: endl, ends, flush
- Манипулатори за бройна система: hex, oct, dec
- Манипулатори за поле: setw, setfill, left, right, internal
- Манипулатори за дробни числа: fixed, scientific, setprecision
- Манипулатори за формат: setiosflags, setbase
- ... и много други

1e5
100000