

bayan

Условие

Пользуясь имеющимися в библиотеке Boost структурами и алгоритмами разработать утилиту для обнаружения файлов-дубликатов.

Утилита должна иметь возможность через параметры командной строки указывать

- имена для сравнения
- размер блока, которым производится чтения файлов, в задании этот размер упоминается как S
- один из имеющихся алгоритмов хэширования (crc32, md5 - конкретные варианты определить самостоятельно), в задании эта функция упоминается как H

Результатом работы утилиты должен быть список полных путей файлов с идентичным содержимым, выводимый на стандартный вывод. На одной строке один файл. Идентичные файлы должны выводиться подряд, одной группой. Разные группы разделяются пустой строкой.

Обязательно свойство утилиты - бережное обращение с дисковым вводом выводом.

Каждый файл может быть представлен в виде списка блоков размера S. Если размер файла не кратен, он дополняется бинарными нулями. Файлы разной длины не могут быть дубликатами по определению.

Файл world.txt из одной строки

```
Hello, World\n
```

При размере блока в 5 байт, будет представлен как

```
Hello  
, Wor  
ld\n\0\0
```

Каждый блок должен быть свернут выбранной функцией хэширования. Возможные коллизии игнорируются. Из предположения, что

```
H("Hello") == A  
H(", Wor") == B  
H("ld\n\0\0") == C
```

Наш файл world.txt может быть представлен в виде последовательности ABC

Рассмотрим второй файл cpp.txt

```
Hello, C++\n
```

Который после хэширования блоков

```
H("Hello") == A  
H(", C++") == D  
H("\n\0\0\0\0") == E
```

может быть представлен в виде последовательности ADE

Порядок сравнения этих файлов должен быть максимально бережным. То есть обработка первого файла world.txt вообще не приводит к чтению с диска, ведь нам еще не с чем сравнивать. Как только мы добираемся до файла cpp.txt только в этот момент происходит первое чтение первого блока обоих файлов. В данном случае блоки идентичны, и необходимо прочесть вторые блоки, которые уже различаются. Файлы различны, оставшиеся данные не читаются.

Файлы считаются идентичными при полном совпадении последовательности хешей блоков.

Пример отключения буферизации.

```
#include <fstream>  
  
int main()  
{  
    std::ifstream file;  
  
    file.rdbuf()->pubsetbuf(nullptr, 0);  
    file.open("b.cpp");
```

```
char buffer[1000];
file.read(buffer, 11);
file.close();
}
```

Результат

- описание параметров в файле README.md в корне репозитория
- пакет bayan содержащий исполняемый файл bayan опубликован на bintray
- отправлена на проверку ссылка на страницу решения (github)

Проверка

Задание считается выполнено успешно, если после просмотра кода, подключения репозитория, установки пакета и запуска бинарного файла командой (параметры из описания):

```
$ bayan [...]
```

будут обнаружены файлы-дубликаты, без ложных срабатываний и пропуска существующих дубликатов.

Количество прочитанных данных с диска минимально.

Оценка

максимум 11

баллы	за что
5	Успешно прошла проверка.
2	Возможно указать более двух файлов на определение их идентичности.
2	Возможно указать более двух файлов на определение дубликатов среди них.
2	Возможно указать путь для определения дубликатов среди вложенных файлов.