Лабораторная работа по операционным системам. Алгоритмы планирования ввода-вывода. Алгоритм C-LOOK.

Соколов Николай, ФПКиФ 2-2

1 Общие сведения

Программа представляет собой реализацию некоторых алгоритмов планирования ввода-вывода, таких как:

- First Come First Served (FCFS)
- Shortest Seek Time First (SSTF)
- SCAN
- C-SCAN
- LOOK
- C-LOOK

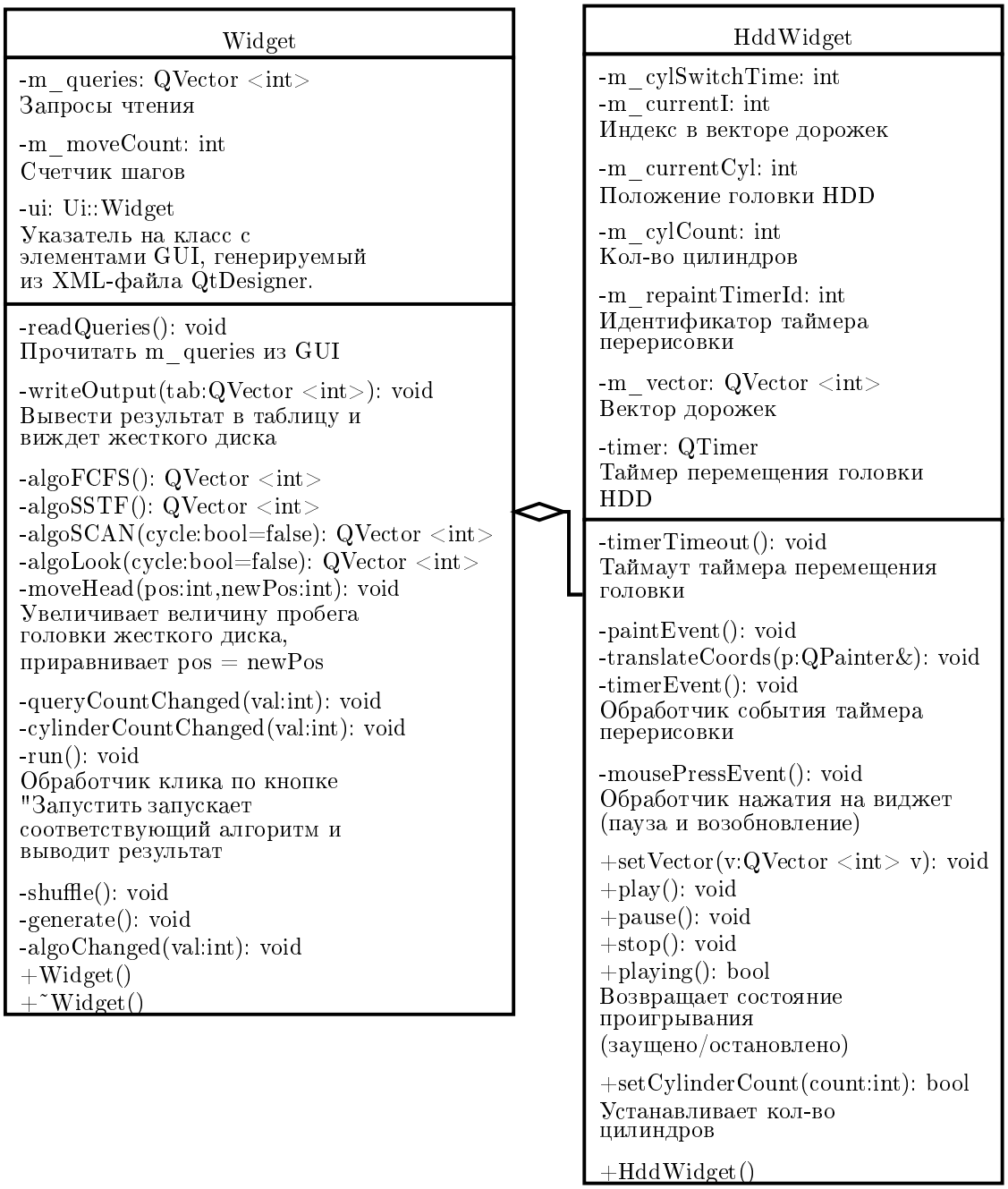
Реализация выполнена на языке *C++* с использованием кроссплатформенного фреймворка *Qt* и может быть скомпилирована под большинство современных платформ, таких как *Windows*, *GNU/Linux*, *MacOS X*, а так же *Maemo* и *Symbian*. Для функционирования программы необходимы библиотеки *QtCore* и *QtGUI*.

2 Функциональное назначение

По введенной последовательности цилиндров программа формирует правильную очередность чтения данных с диска в соответствии с выбранным алгоритмом и направлением движения головки жесткого диска. Кроме этого, в окне программы отображается схематическое анимированное изображение жесткого диска, на котором наглядно иллюстрируется полученная последовательность чтения секторов.

3 Описание логической структуры

Объектная структура состоит из двух классов: `Widget`, реализующего окно, и `HddWidget`, реализующего виджет жесткого диска. Классы объявлены в файлах `widget.h` и `hddwidget.c` соответственно. Объектная структура приведена на следующей UML диаграмме:



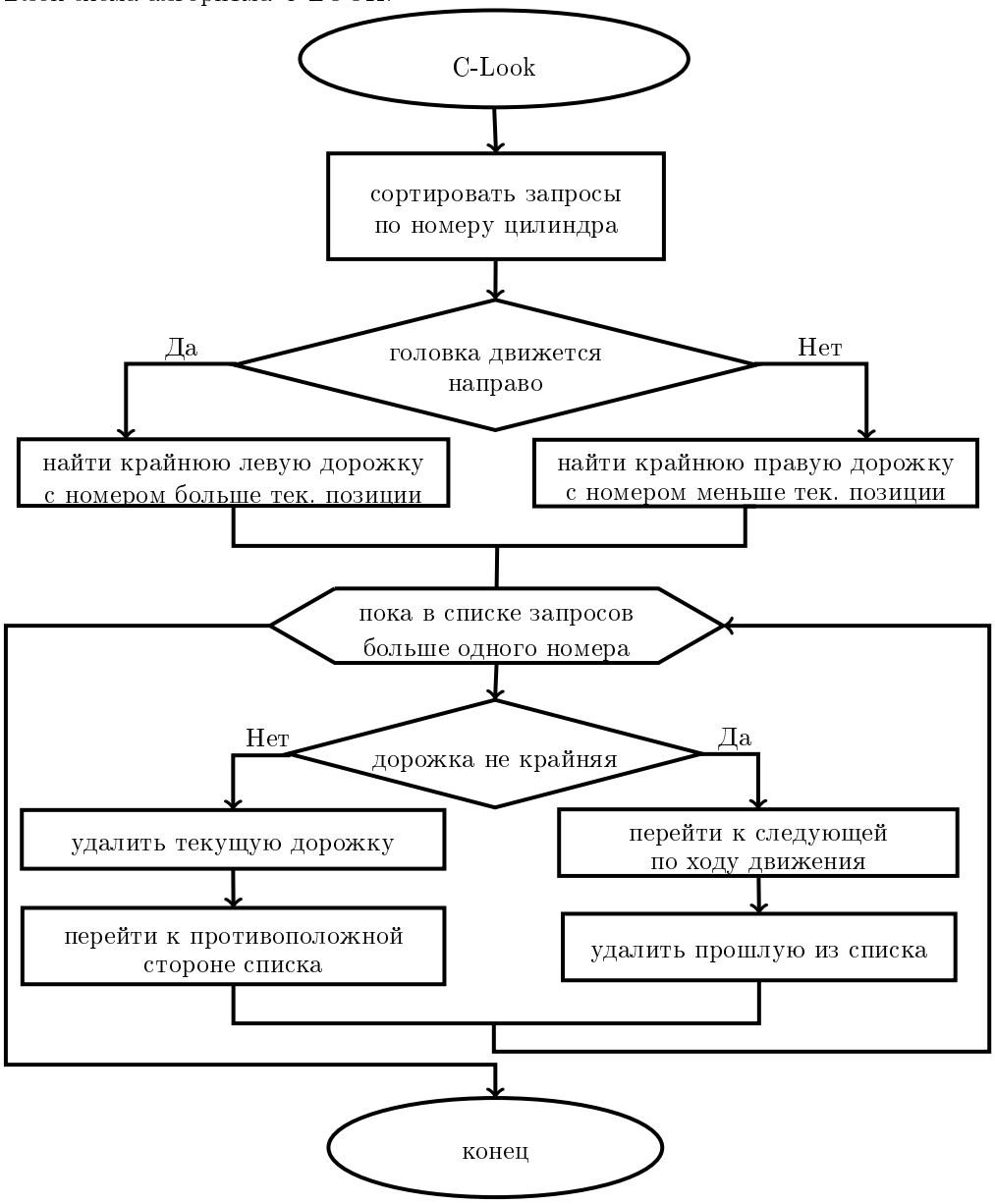
Алгоритм FCFS наиболее прост и заключается в переходе к дорожкам в последовательности их ввода. Как правило, результат работы алгоритма наименее оптимален.

SSTF выбирает наиближайшую к текущей дорожку. Результат работы этого алгоритма намного лучше, чем у FCFS в общем случае, но возможны крайние случаи при которых SSTF ненамного лучше FCFS.

Алгоритм SCAN состоит в последовательном переходе от дорожки к дорожке в той последовательности, в которой они идут на диске. Если в текущем направлении перемещения головки не осталось дорожек, она доходит до края диска и начинает двигаться в противоположном направлении.

LOOK - алгоритм, похожий на SCAN. В случае LOOK не происходит перемещения к краю диска, направление меняется после прочтения “крайней“ дорожки. Циклические модификации SCAN и LOOK (C-SCAN и C-LOOK) отличаются от нециклических только тем, что движение головки осуществляется только в одну сторону. Это позволяет избежать большого ожидания для дорожек, находящихся на противоположном крае диска.

Блок-схема алгоритма C-LOOK:



Исходный код функции, реализующей LOOK и C-LOOK:

```
 QVector <int> Widget::algoLook(bool cycle /* = false */ )
{
        // выделяем память для результата
        QVector <int> res;
        res.reserve(m_queries.size()+1);
        // направление движения
        enum Direction { LeftToRight, RightToLeft } direction;
        direction = ui->directionBar->value() ? LeftToRight : RightToLeft;

        // текущая позиция
        int pos = ui->startPosition->value();
        res.push_back(pos);

        // подготавливаем очередь для обработки
        m_queries.append(pos);
        qSort(m_queries);
        // копируем в связный список для удаления за O(1)
        QLinkedList <int> unread;
        foreach (int val, m_queries)
                unread.append(val);
        // итератор текущей позиции
        QLinkedList <int>::iterator it;
        // находим первоначальную позицию в списке
        if (direction == LeftToRight)
                for (it = unread.begin(); *it!=pos; ++it);
        else
                for (it = unread.end()-1; *it!=pos; --it);

        // пока есть куда переходить
        while (unread.count()-1)
        {
                if (direction == LeftToRight)   // если движемся направо
                {
                        if (it+1 != unread.end())   // если не крайний
                        {
                                moveHead(pos, *(++it)); // перемещаемся
                                res.push_back(pos);
                                unread.erase(it-1);     // и удаляем обработанный
                        }
                        else
                        {                               // если крайний
                                if (cycle)              // если C-LOOK
                                {
                                        unread.erase(it);   // удаляем текущую позицию
                                        it = unread.begin();// переходим в начало
                                        moveHead(pos, *it);
                                        res.push_back(pos);
                                }
```

```
                        else                    // если LOOK
                        {                       // меняем направление
                                direction = RightToLeft;
                        }
                }
        }
        else
        {
                if (it != unread.begin())   // если не крайний
                {
                        moveHead(pos, *(--it)); // перемещаемся
                        res.push_back(pos);
                        unread.erase(it+1);     // и удаляем прошлый
                }
                else
                {
                        if (cycle)              // если C-LOOK
                        {
                                unread.erase(it);   // удаляем обработанный
                                it = unread.end()-1;
                                moveHead(pos, *(it));// перемещаемся
                                res.push_back(pos);
                        }
                        else                    // если LOOK
                        {                       // меняем направление
                                direction = LeftToRight;
                        }
                }
        }
    }

    return res;                         // возвращаем результат
}
```

4 Входные данные

В программу вводятся следующие параметры:

- максимальный номер цилиндра (нумерация начинается с нуля)
- количество запросов
- начальная позиция головки жесткого диска
- направление движения головки жесткого диска

5 Выходные данные

Результатом работы программы является последовательность номеров цилиндров, в которой будет двигаться головка жесткого диска.

6 Пример работы программы

Для 10 запросов с максимальным номером цилиндра 99 и начальной позицией 25:

Ввод: 63, 64, 37, 16, 9, 93, 20, 63, 91, 74.
Вывод: 25, 35, 63, 63, 64, 74, 91, 93, 9, 16, 20.
Длина пробега: 163.