Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ **ИНФОРМАТИКА И СИСТЕМЫ УПРАВЛЕНИЯ**

КАФЕДРА **«ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ ЭВМ И ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ» (ИУ7)**

НАПРАВЛЕНИЕ ПОДГОТОВКИ **09.03.01 Информатика и вычислительная техника**

ОТЧЕТ

по рубежному контролю № 1

Название: Стена Фокс

Дисциплина: Анализ алгоритмов

Студент	ИУ7-52Б		Короткая В. М.
	(Группа)	(Подпись, дата)	(И.О. Фамилия)
Преподаватель			Волкова Л.Л.
		(Подпись, дата)	(И.О. Фамилия)

Москва, 2021

Содержание

Введение **3**

1 Аналитическая часть **4**

1.1 Стена Фокса 4

2 Конструкторска часть **6**

2.1 Схемы 6

3 Технологическая часть **8**

3.1 Средства реализации 8

3.2 Сведенья о модулях программы 8

3.3 Реализация 8

Список литературы **12**

Введение

1. Аналитическая часть

В данном разделе будет описана задача решаемая в данной работе.

1.1. Стена Фокса

Решая задачу построения «стены Фокса», можно разбить стену на равные по длине участки и поручить постройку каждого участка отдельному каменщику. В этом случае все каменщики могут начать работу одновременно, укладывая нижний слой кирпичей (рис. 1.1). Перед укладыванием очередного слоя каждому каменщику следует убедиться, что кирпичи предыдущего слоя уложены не только на его участке, но и на прилегающих участках. Если работа на соседних участках еще не закончена, возникают вынужденные паузы, связанные с синхронизацией работ на соседних участках. Отметим, что паузы могут возникать, даже если каменщики работают с одинаковой скоростью, поскольку объем работ, вообще говоря, зависит от номера участка, например он разный для крайних и внутренних частей стены. Можно сделать вывод о том, что эффективность организации параллельной работы будет тем выше, чем длиннее участки стены, и при достаточно длинных участках следует ожидать эффективности, близкой, но все же меньшей 1. При коротких участках стены (очень много каменщиков) эффективность будет невысокой, поскольку время независимой работы каждого каменщика будет невелико, они будут много времени проводить в ожидании друг друга на стыках участков. Отметим, что этот метод организации работы эффективен, даже если стена не очень высокая. Главным фактором, определяющим эффективность, является ее достаточная протяженность, по отношению к числу задействованных каменщиков.

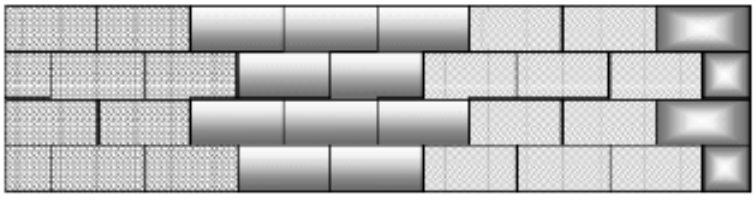

Рис. 1.1: Визуализация работы каменщиков.

Вывод

В данном разделе была рассмотрена задача "Стена Фокса".

2. Конструкторска часть

В данном разделе будут разработаны схемы алгоритмов, реализующих стену Фокса.

2.1. Схемы

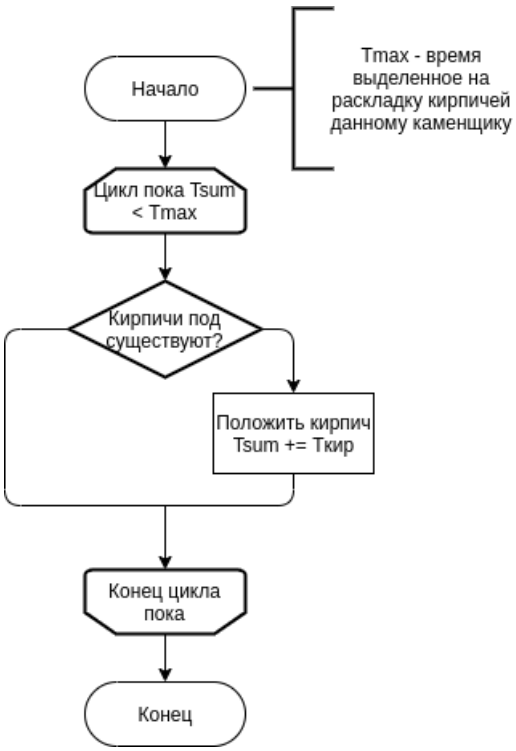

Рис. 2.1: Схема работы одного каменщика.

Для реализации визуализации стены Фокса, введем пользовательские типы данных:

- struct wall -
  - brickWall - булевая матрица, описывающая существование кирпича на определенном участке;
  - border - массив, описывающий границы работы каменщиков;
- struct bricklayer -
  - number - номер каменщика;
  - countBrick - количество положенных кирпичей;
  - speed - скорасть работы каменщика;

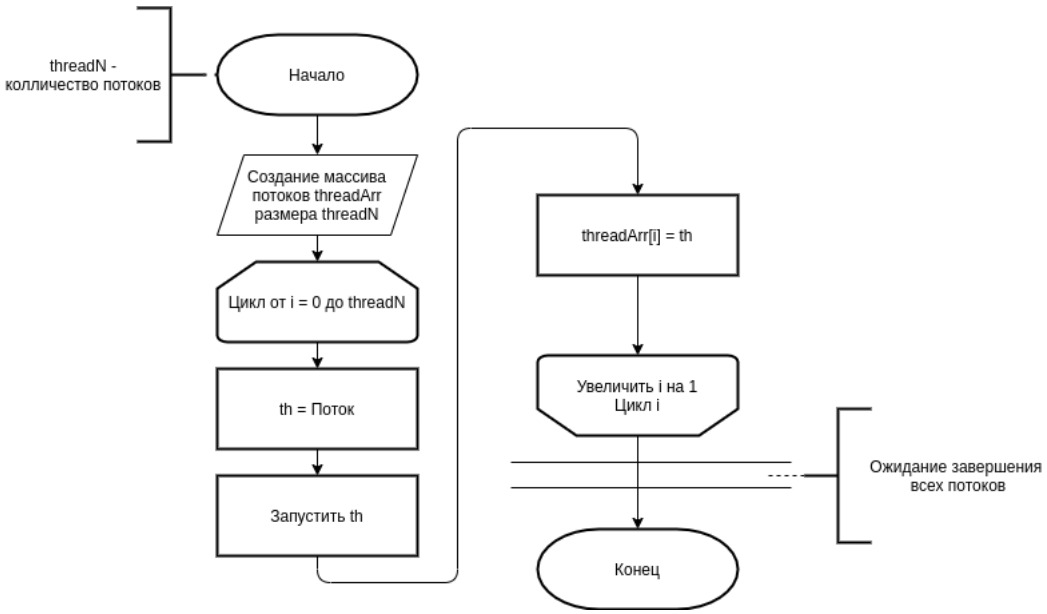

Рис. 2.2: Схема создания каменщиков в разных потоках.

Вывод

На основании теоритических данных были построены схемы алгоритмов.

Так же приведено описание пользовательских структур данных.

3. Технологическая часть

В данном разделе приведены средства реализации и листинги кода.

3.1. Средства реализации

В качестве языка программирования был выбран c++. Данный язык знаком и предостовляет все необходимые ресурсы. В качестве среды разработки я использовала Visual Studio Code, т.к. считаю его достаточно удобным и легким. Visual Studio Code подходит не только для Windows, но и для Linux, это еще одна причина, по которой я выбрала VS code, т.к. у меня установлена ОС fedora 34.

3.2. Сведенья о модулях программы

Данноя программа разбита на модули:

- main.cpp - файл, содержащий точку входа в программу;
- MainWindow.cpp - файл, содержащий реализацию алгоритмов.

3.3. Реализация

Листинг 3.1: Функция запускающая конвейер.

Листинг 3.1: Функция запускающая конвейер.

```
1  void MainWindow::brickparallel(QPainter &painter)
2  {
3      std::vector<std::thread> threadArray;
4      int start;
5      int end;
6
7      QVector<QBrush> color =
8      {QBrush(Qt::red, Qt::SolidPattern),
9       QBrush(Qt::darkCyan, Qt::SolidPattern),
10      QBrush(Qt::green, Qt::SolidPattern),
11      QBrush(Qt::yellow, Qt::SolidPattern) };
12
```

```
13      for (int i = 0; i < 4; i++)
14      {
15          start = wall.border[i][0];
16          end = wall.border[i][1];
17          painter.setBrush(color[i]);
18          threadArray.push_back
19          (std::thread(&MainWindow::bricklayer, this, start,
20           end, std::ref(brickl), i, std::ref(painter),
21            color[i]));
22      }
23
24      for (int i = 0; i < 4; i++)
25      {
26          threadArray[i].join();
27      }
28  }
```

Листинг 3.1: Функция реализующая каменщика.

```
1  void MainWindow::bricklayer(int start, int end,
2  Bricklayer brick[], int num, QPainter &painter,
3  QBrush color)
4  {
5      const int delay_value = 1;
6
7      int time = 0;
8
9      int row = (end - start)/deltaBrick;
10     int column = 1;
11     int delta = 0;
12     int x = 0;
13     int y = 0;
14     bool flag;
15     int wait = 1;
16     int i = 0;
17
```

```
18      int blok = 1500;
19
20      while (time < blok)
21      {
22          i++;
23          row--;
24          x = start + deltaBrick * (row) + delta;
25          y = y_max - deltaBrick/2 * column;
26          flag = (wall.brickWall[column - 1][x/25]*
27          wall.brickWall[column - 1][x/25 + 1]);
28          if ( (  flag ) && !(wall.brickWall[column][x/25]) &&
29          !( wall.brickWall[column][x/25 + 1] )) {
30              mtx1.lock();
31
32              painter.setBrush(color);
33              painter.drawRect(x, y, deltaBrick, deltaBrick/2);
34
35              wall.brickWall[column][x/25] = 1;
36              wall.brickWall[column][x/25 + 1] = 1;
37              mtx1.unlock();
38
39
40              mtx2.lock();
41              displayImage();
42              mtx2.unlock();
43
44              wait = (rand()%20 + 1 ) * brickl[num].speed;
45              time += wait;
46
47              delay(delay_value * wait);
48              brickl[num].countBrick++;
49          }
50          else
51          {
```

```
52                  i--;
53              }
54              if (!row) {
55                  row = (end - start)/deltaBrick;
56                  column++;
57              }
58              if (column % 2)
59              delta = 0;
60              else
61              delta = 25;
62          }
63  }
```

Вывод

В данном разделе был реализован выше описанный алгоритм шифрования. Разработано программное обеспечение, представленны листинги программы.

4. Исследовательская часть

В данном разделе будет приведена работа программы.

Список литературы

1. Дж. Макконнел. Анализ алгоритмов. Активный обучающий подход. – М.: Техносфера, 2017. – 267с.

2. Основы программирования на языках Си и C++ для начинающих[Электронный ресурс]. Режим доступа: http://cppstudio.com/ (дата обращения 10.10.2021)

3. LINUX.ORG.RU - Русскоязычная информация о ОС Linux[Электронный ресурс] Режим доступа://www.linux.org.ru/(дата обращения 25.10.2021)

4. Документация языка C++ 98 [Электронный ресурс], режим доступа: http://www.open-std.org/JTC1/SC22/WG21/ (дата обращения 10.12.2021)

5. Р.А. Волков, А.Н. Гнутов, В.К. Дьячков и др. Конвейеры. Справочник. -Машиностроение, Ленинградское отделение, 1984.