Домашнее задание номер 5
по предмету
«Архитектура вычислительных систем».
Вариант 33.
Выполнено студентом группы БПИ208
Дерели Серканом.

Описание задания.

33. ***И снова пляшущие человечки.*** Узнав о планах преступников озвученных в задаче 32, Шерлок Холмс предложил лондонской полиции специальную машину для дешифровки сообщений злоумышленников. Реализовать многопоточное приложение, дешифрующее кодированный текст. В качестве ключа используется известная кодовая таблица, устанавливающая однозначное соответствие между каждой буквой и каким-нибудь числом. Процессом узнавания кода в решении задачи пренебречь. Каждый поток дешифрует свои кусочки текста. При решении использовать парадигму портфеля задач.

Требуется по данной таблице дешифровки зашифрованного текста дешифровать текст. Дешифровать текст нужно параллельно, по частям, с использованием потоков. Рекомендуется использовать парадигму параллельного программирования «портфель задач».

Основная специфика решения, выбранные подходы.

Для реализации поставленной задачи был выбран язык c++ и его стандартная библиотека. При этом, мною было принято решение немного изменить поставленную задачу: вместо взаимооднозначного соответствия между буквой и числом я выбрал взаимооднозначное соответствие между символом и

символом. Почему я так сделал? Потому что далее в задании говорится, что процессом узнавания кода в программе следует пренебречь. Отсюда я сделал вывод о том, что суть задания состоит не в том, чтобы разобраться, как проверять входные данные на декодируемость с помощью таблицы декодирования, а в том, чтобы изучить подход параллельного программирования. Таким образом, пренебречь узнавание кода в программе легче всего заменив число в таблице дешифровки на другой символ. Для реализации многопоточности был выбран подход портфеля задач. Он описан ниже.

Описание парадигмы параллельного программирования, выбранного для выполнения дз.

Для выполнения дз была выбрана рекомендованная парадигма: портфеля задач. Она заключается в том, что большая задача делится на много маленьких независимых друг от друга задач. Работа считается выполненной, когда все такие маленькие задачи будут выполнены. В моём случае, на части делится строка, которую требуется дешифровать. Эти части отдаются потокам на дешифровку, а результат записывается в массив дешифрованных строк. Результатом дешифровки всей строки будет конкатенация всех дешифрованных подстрок исходной строки. Было решено использовать ceil(sqrt(n)) потоков, где n – длина шифрованной строки. Длина подстроки получается тоже ceil(sqrt(n)). Замечу, что строка может не делиться ровно. Все такие случаи отражаются только на длине

последней строки. Для понимания подхода были изучены статьи из списка источников[1][2][3]. В них было написано, что парадигама портфеля задач заключается в разделении большой задачи на много маленьких, но такого же типа. Затем в делегировании этих задач рабочим(потокам), а в конце сборке результата работы всех рабочих и выдача итогового ответа. Парадигма портфеля задач хороша тем, что она позволяет эффективнее(многопоточность) решить задачу, которую можно разделить на много однотипных независимых задач. Дешифровка строки дешифровкой отдельных её символов — яркий пример этого. В источниках было написано, что преимуществом данной парадигмы является пластичность в выборе количества рабочих(потоков), выполняющих какую-то часть работы. Простота распределения задачи по рабочим. Таким образом, каждый поток выполняет примерно один и тот же объём работы.[4].

Подробное описание хода дешифровки строки

Для дешифровки строки у пользователя сначала просят ввести в консоль таблицу дешифровки (как описано в формате входных и выходных данных ниже). Затем строка делится на корень из длины подстрок, каждая из которых отдаётся отдельному потоку выполнения программы. Каждый поток дешифрует свою подстроку с помощью таблицы дешифровки, а затем записывает расшифрованную строку в массив расшифрованных строк. При этом в записи в массив используется мьютекс, чтобы несколько потоков не

имели одновременный доступ к массиву дешифрованных подстрок. На самом деле, я не уверен, надо это или нет, так как каждый потом дешифрует подстроку с разным индексом в массиве, так что доступа к одним и тем же данным не будет. Однако, на всякий случай, мьютекс я использовал. Результатом работы дешифратора становятся конкатенированные дешифрованные подстроки.

Формат входных и выходных данных.

Мною был выбран исключительно консольный ввод и вывод. В начале программы от пользователя просят ввести таблицу дешифровки. Формат ввода таблицы таков:
<char1>-<char2>. То есть через «-» вводится 2 символа: сначала зашифрованный, а затем дешифрованный.
То есть если дешифратор должен перевести А в Б, то ввести нужно А-Б. Такие пары вводятся пользователем до тех пор, пока пользователь того хочет. Вводом слова «stop» он может закончить ввод таблицы. Таблицу можно переввести после дешифровки очередного введённого слова.

После ввода дешифровочной таблица пользователю предлагается ввести строку для дешифровки. Если во введённой ранее дешифровочной таблице не содержится одного или несколько символов из слова, то программа выдаёт сообщение об ошибке и просит либо ввести другое слово, либо ввести новую таблицу дешифровки, либо дополнить старую.

Программа продолжает работать до тех пор, пока пользователь не введёт «exit». На основных этапах дешифровки строки выводятся сообщения-флаги в консоль. Реализована модификация дешифровочной таблицы.

Замечание по поводу компиляции программы.

Для того, чтобы консольно скомпилировать программу, использующую библиотеку потоков, нужно использовать специальный флаг. Чтобы этого избежать была добавлена строчка

```
SET(CMAKE_CXX_FLAGS "${CMAKE_CXX_FLAGS} -pthread")
```

в файл CmakeLists.

Тестовые данные.

Программа тестировалась на корректность. Ниже приведены несколько тестов.

Тест 1.

Когда просят ввести таблицу вводим это:

n-s
a-e
s-r
l-k
i-a
t-n

Когда просят ввести слово вводим это:

naslim

Выходные данные:
serkan.
В консоли также появятся сообщения-флаги, возникающие при дешифровке строки.

Тест 2(когда в таблице нет символа).

Когда просят ввести таблицу вводим:
n-s
a-e
s-r
l-k
i-a
t-n

Когда просят ввести слово вводим это:
nas lim

Дальше нас попросят выбрать одну из опций: получить новую таблицу, модифицировать старую, ввести новое слово или выйти из программы. Выбираем модификацию следующим образом:
modify

Далее вводим:
-
(Пробел, Тире, Пробел) — чтобы добавить пробел в

таблицу. Далее заново вводим слово:
nas lim
Выведется дешифрованное слово:
ser kan

Список источников:

1. **https://pro-prof.com/forums/topic/parallel-programming-paradigms**. Парадигмы параллельного программирования.
2. **https://studref.com/702392/informatika/paradigmy_parallelnyh_prilozheniy**. Парадигмы параллельного программирования.
3. **http://staff.mmcs.sfedu.ru/~dubrov/files/sl_parallel_05_paradigm.pdf** . Парадигмы параллельного программирования.
4. http://www.soft.architecturenet.ru/70/index-upravljajushhij-rabochie-raspredelennyj-portfel.htm