*Софийски университет „Св. Климент Охридски“  
 Факултет по математика и информатика*

*Курсова работа по Структури от данни и програмиране*

*специалност Компютърни науки*

Тема: Машини с неограничени регистри

Съдържание

1. Увод……………………………………………………………………………………..…....3
   1. Описание и идея на проекта
   2. Цели и задачи на разработката
   3. Връзка към хранилището в Github
2. Общ поглед върху проекта……………………………………………………………….....5
3. Управление на паметта и алгоритми……………………………………………………….6
4. Тестване………………………………………………………………………………………6
5. **Увод**
   1. **Описание и идея на проекта**

Проектът Машини с неограничени регистри (МНР) реализира абстрактна машина, разполагаща с неограничена памет. Паметта на машината се представя с безкрайна редица от естествени числа *m*[0], *m*[1], …, където *m*[*i*] **∈** N. Елементите *m*[*i*] на редицата наричаме „клетки“ на паметта на машината, а числото *i* наричаме „адрес“ на клетката *m*[*i*].

МНР разполага с набор от инструкции за работа с паметта. Всяка инструкция получава различен брой аргументи (операнди) и може да предизвика промяна в стойността на някоя от клетките на паметта. Инструкциите са:

* ZERO n: Записва стойността 0 в клетката с адрес n.
* INC n: Увеличава с единица стойността в клетката с адрес n.
* MOVE x y: Присвоява на клетката с адрес у стойността на клетката с адрес х.
* JUMP z: Изпълнението на програмата прескача и продължава от инструкция с номер z.
* JUMP x y z: Ако съдържанието на клетките х и у съвпада, изпълнението на програмата прескача и продължава от инструкция с номер z.

Програма за МНР се нарича всяка последователност от инструкции на МНР и техните операнди. Всяка инструкция има пореден номер – index. Изпълнението на програмата започва от първата инструкция и преминава през всички инструкции последователно, освен при JUMP инструкцията, която променя последователността на изпълнение на програмата.

Освен инструкции програмата може да съдържа и друг вид оператори – команди. Командите се въвеждат по различен начин – започват със символа „**/**“ и нямат поредни номера за разлика от инструкциите. Също така не влияят върху номерацията на инструкциите. Интерпретаторът поддържа следните команди:

* /zero x y: Нулира клетките с адреси от х до у включително.
* /set x y: Променя на у съдържанието на клетката с адрес х.
* /copy x y z: Копира съдържанието на z последователни клетки, започващи от адреса х в съответните z последователни клетки с адрес от у нататък.
* /mem x y: Извежда на стандартния изход съдържанието на клетките с адреси от х до у включително.
* /load <filename>: Зарежда програма за МНР от текстов файл
* /run: Изпълнява заредената програма, спазвайки последователността на инструкциите и командите.
* /add <filename>: Разширява заредената програма за МНР с оператори, прочетени от текстов файл.
* /quote <string>: Добавя нов оператор на края на заредената програма.
* /code: Извежда на стандартния изход заредената в паметта програма.
* /comment <string>: Служи за добавяне на коментар.

* 1. **Цели и задачи на разработката**

Целта на проекта е да се създаде оптимална програма с максимално разнообразна функционалност, която да отговаря на поставената задача. Проектът е изграден така, че да работи с абстрактни данни, което дава възможност за по-лесно описание и модификация на програмата.

* 1. **Връзка към хранилището в Github:**

<https://github.com/KonstantinaKS21/Unlimited-Register-Machine>

1. **Общ поглед върху проекта -   
   Основни дефиниции, концепции, подходи и алгоритми**

Данните в проекта са разпределени в съответните класове. За целите на задачата е реализирано наследяване на три нива (фиг.1):

Operator  
Base Class

Instruction  
Derivative Class

INC

ZERO

MOVE

JUMP

ZeroCommand  
Derivative Class

SetCommand  
Derivative Class

CopyCommand  
Derivative Class

MemCommand  
Derivative Class

CodeCommand  
Derivative Class

RunCommand  
Derivative Class

Фиг.1

Всеки клас съдържа необходимите за неговата имплементация член-данни и член-функции, които са разпределени в *public*, *private* или *protected* секциите в зависимост от необходимия достъп до тях. Също така са използвани външни функции, които не са членове на класовете като функцията void checkLength(vector<string>& line, int len), която проверява дали въведеният оператор има достатъчен брой аргументи, за да бъде правилно реализиран. Използвана е конвенцията за именуване на функции и променливи *camelCase*.

Класовете съдържат съответните конструктори, за да може програмата да работи правилно. В тях са инициализирани член-данните чрез обобщена синтактична конструкция (в така наречените инициализиращи списъци). Използвано е наследяването, където Operator е абстрактен клас, в който са дефинирани чисто виртуалните функции void execute() и string toString(), които се *override*-ват в съответните *derivative* класове. Използван е *Command Design Pattern*, което улеснява добавянето и премахването на нова функционалност към машината.

Реализиран е и класът InvalidCommandException, който се грижи за *handle*-ването на грешките.

За реализация на паметта на МНР се използва *sparse array*, който е пресъздаден чрез std::map , където *key* е индексът на регистъра, a *value* е съответната стойност в клетката.

Използвани са следните библиотеки:

* <fstream> // библиотека за работа с файлове
* <iostream> //стандартантната библиотека за вход и изход
* <sstream> //stringstream за функцията split\_string();
* <string>
* <vector>
* <map>

1. **Управление на паметта и сложност**

Паметта автоматично се заделя и унищожава от класа *vector* в стандартната библиотека, т.е. работим с динамична памет. Повечето използваните алгоритми са с линейна сложност, а деференцирането в *map* е с логаритмична.

1. **Тестване**

При изготвянето на проекта са направени тестове с помощта на подходящи файлове, съдържащи примерни програми, които изпълняват желаната функционалност на машината с неограничен регистър.