

**ФАКУЛТЕТ ПО МАТЕМАТИКА И ИНФОРМАТИКА –  
СОФИЙСКИ УНИВЕРСИТЕТ „СВ. КЛИМЕНТ  
ОХРИДСКИ“**

*Курс Обектно-ориентирано програмиране  
за специалност Информационни системи  
летен семестър 2019/2020 г.*

# Проект IML

# Глава 1: Увод

## 1.1. Описание и идея на проекта

Идеята на проект *IML* е да се напише *parser* за езика *IML (Ivan MarkUp Language)*. Езикът съдържа 11 тага и всеки от тях съдържа данни, които указват каква операция трябва да се извърши. Потребителят ще въведе имена за входен и изходен файл и програмата ще прочете информацията от първия, ще я интерпретира и ще я запише във втория файл.

## 1.2. Цел и задачи на разработката

Избиране на подходяща структура от данни, с която ще работи програмата.

Създаване на необходимите класове.

Възможните операции са следните :

- ✓ *MAP*
  - *INC „N”* – увеличава всеки елемент от списък с N
  - *MLT “N”* – умножава всеки елемент от списък по N
- ✓ *AGG*
  - *SUM* – връща сбора от елементите на списък
  - *PRO* – връща резултата от умножението на всички елементи на списък
  - *AVG* – връща средното аритметично от елементите на списък
  - *FST* – връща първия елемент от списък
  - *LST* – връща последния елемент от списък
- ✓ *SRT*
  - *REV* – обръща списък
  - *ORD*
    - *ASC* – сортира списък във възходящ ред
    - *DSC* – сортира списък във низходящ ред
  - *SLC „N”* – връща подсписък от посочения индекс нататък
  - *DST* – премахва всички дубликати от списък

## 1.3. Структура на документацията

В документацията на този проект се описват идеята, целите, крайният резултат и трудностите, срещнати по време на изпълнението. Описва се структурата на проекта, взетите решения и са включени извадки от кода на програмата.

## Глава 2: Преглед на предметната област

### 2.1. Основни дефиниции и концепции

За успешното реализиране на програмата е избрана работа с двойно свързан списък – *DLList*. Имплементацията на *DLList* е направена с шаблонен клас, който също така включва и някои необходими функции за изпълняване на необходимите операции. Този клас също е пригоден и за ползване извън заданието.

### 2.2. Дефиниране на проблеми и сложност на поставената задача

Основен проблем при разработването е необходимостта от работа със различни *streams* – пример *stringstream*, използван за записване на данните във файл и четенето от файл. Този проблем беше преобладан, благодарение на съответните източници и подходяща литература. Също така, сложно беше изграждането на самия двойно свързан списък.

### 2.3. Подходи за решаване на поставените проблеми

За правилното изпълнение на задачата са изградени различни класове, правейки разрешаването на проблеми по-лесно. Командите, въведени от потребителя, преминават през съответните проверки, които следят за правилно въведени данни и изпълнение на необходимите условия.

## Глава 3: Проектиране

### 3.1. Обща архитектура

Проектът се съдържа от два вида файлове - *\*.cpp*, *\*.h*. Програмата започва от главния файл (*main.cpp*), където са включени файловете с дефинираните класове и файл, съдържащ останалите операции. В заданието са включени и файлове с разширение *\*.txt*, като един от тях е примерен за проверка на изпълнението, но програмата включва опция за създаването на нов текстов файл. За целта на изпълнението са създадени клас *Expressions*, клас *Tag*, клас *Iml* и шаблонен клас *DLList*.

### 3.2. Структура на проекта

#### 3.2.1 Expressions

В заданието има два файла, в които са реализирани основните функции, които програмата трябва да може да изпълнява. Единият от тях е Expressions. Основните команди са *MAP*, която прилага функция, към всеки елемент от списък; *AGG*, която извършва операция върху лист и връща едно число; *SRT*, която сортира списък по различни начини. С командите се работи от файлът IML.

### 3.2.2. IML

Основен файл, върху който е построена програмата. В него чрез операциите *getKind*, *getExpression*, *getArgument* се определя коя функция от expression да се извика. Работата с файлове е реализирана, чрез използване на *stringstream* и *fstream*.

### 3.2.3. DLLList<T>

Това е структурата от данни, която се използва при реализацията на проекта. Избрана е, заради бързите алгоритми за сортиране и улеснената работа при четене и записване във файл.

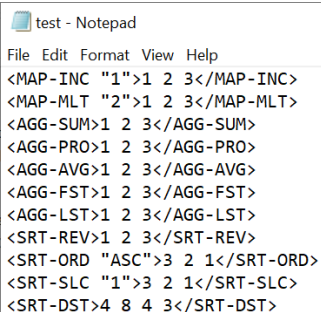
## Глава 4 : Реализация на проекта

Реализацията на проекта става след като потребителя въведе две имена на файлове. Примерен вход и изход от конзолата при въвеждане на име, запазването му във файл и излизане от програмата.

*Input:*

```
int main()
{
    IML parser1;
    std::string input;
    std::string output;

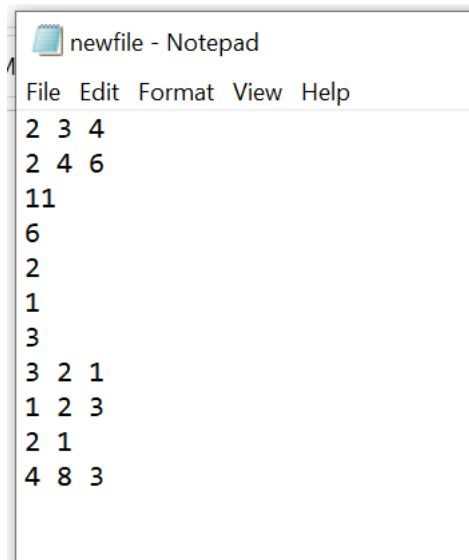
    std::cout << "Enter name of input file.\n";
    std::cin >> input;
    std::cout << "Enter name of output file.\n";
    std::cin >> output;
    parser1.parser(input, output);
}
```



```
File Edit Format View Help
<MAP-INC "1">1 2 3</MAP-INC>
<MAP-MLT "2">1 2 3</MAP-MLT>
<AGG-SUM>1 2 3</AGG-SUM>
<AGG-PRO>1 2 3</AGG-PRO>
<AGG-AVG>1 2 3</AGG-AVG>
<AGG-FST>1 2 3</AGG-FST>
<AGG-LST>1 2 3</AGG-LST>
<SRT-REV>1 2 3</SRT-REV>
<SRT-ORD "ASC">3 2 1</SRT-ORD>
<SRT-SLC "1">3 2 1</SRT-SLC>
<SRT-DST>4 8 4 3</SRT-DST>
```

*Output:*

```
Enter name of input file.  
test.txt  
Enter name of output file.  
newfile.txt  
File is now opened!  
File is now opened!
```



## Глава 5: Заключение

В бъдеще проекта може да бъде развит като се добави представителен потребителски интерфейс, управлението на паметта стане по-ефективно и се добави възможност за работа с файлове от различни формати. Отделно, двойно свързания списък може да се подобри.

[https://github.com/mariya29k/PROJECT\\_IML.git](https://github.com/mariya29k/PROJECT_IML.git)