**Документация по проект №6 Бази данни**

[Линк към моя Github](https://github.com/LyudmilDS?tab=repositories)

Структура на документацията:

* Увод
* Преглед на предметната област
* Проектиране
* Реализация, тестване
* Заключение

**Глава 1: Увод**

***1.1*** ***Описание и идея на проекта***:

Проектът работи с бази данни съставени от таблици. Отваря се таблица, като може и да са повече таблици, и се дава избор на потребителя за това какви функции да извърши над тях. После всяка записана таблица се добавя в каталог(списък), от където по-късно потребителят може да пак да я отвори и да извърши поредица функции.

***1.2*** ***Цел и задачи на разработката***:

Целта на програмата е да осигури възможност на потребителя да работи с файлове с таблици. Тъй като таблиците са от съществено значение, защото представляват систематизирана съвкупност от данни. Те се използват за записване и представяне на най-различна информация: от числа до цели изречения, и това ги прави толкова необходими на всеки, който работи с някакъв тип данни. Една таблица може да бъде полезна сама по себе си, но цял списък от няколко на брой е още по-полезен. Точно затова програмата създава списък от таблици и позволява да се създават допълнителни такива, да се добавят към списъка и да се отварят отново в бъдеще, когато потребителят реши, че му трябва да прочете някоя. По проекта предстоят да се изгладят някои детайли за по-голяма ефективност на работата, като по-точно и по-гладко обработване на всеки файл и намаляване на действията, изпълнявани от конзолата, с цел по-бързо извършване на всяко основно действие. Както и добавяне на основни функции, с цел улеснение и бързо извършване на действията.

**Глава 2: Преглед на предметната област**

***2.1.* *Основни дефиниции, концепции и алгоритми, които ще бъдат използвани***

В проектът ще бъдат използвани работа с двоични (бинарни) файлове. Обработване на данни и записването им във файлове и след това четене от там. Също ще бъдат използвани вектори за съхранение на множество от еднотипна информация. Например имената на всички записани таблици. По естество ще има файлово разделение за по-добра структура и по-добро четене на кода, както и множество от функции.

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

Проблемите, за които програмата е създадена да решава са:

* Запис на всякакъв тип информация – числова или символна
* Търсене на данни в дадена таблица
* Разширяване на таблицата – добавяне на колони и/или редове
* Копиране на данни от една таблица в нова
* Сравняване на две таблици по стойности и тип на всяка колона
* Основни аритметични операции върху числовата част на всяка таблица

Сложността на поставената задача беше подходяща за дадения срок за нейното изпълнение.

***2.3. Подходи, методи за решаване на поставените проблемите***

За решаване на гореспоменатите проблеми, прпограмата използва множество функции.

* Потребителят първо укозва какъв тип данни да бъдат колоните на всяка таблица
* Обхождане на таблицата като двумерен масив с цел намиране дадени стойности, промяна на стойността на клетки, извършване на основни аритметични операции, разширяване на таблицата

**Глава 3: Проектиране**

***3.1. Обща архитектура***

В проекта не са използвани класове и обекти, тъй като задачата може да бъде изпълнена и без тях. Примитивните типове данни са напълно достатъчни за осъществяване на заданието.

Всеки файл се състои от таблица, като след всяка колона има символ за разделение(табулация или интервал). След края на таблицата, на нов ред са записани типа на всяка колона, а след това са записани размерите на таблицата като последни два символа във всеки файл.

***3.2. Диаграми и слоеве с най-важните извадки от кода***

За да бъде точно прочетен файлът и цялата таблица, последните два символа във всеки файл обозначават размерите на таблицата- първият от двата символа за брой редове, а вторият и последен за файла – брой колони. Те са от целочислен вид, разделени със символ за разстояние (интервал или табулация). За прочитане на тези два символа файловият указател за четене “*get pointer*” се премества 9 байта преди указателя за край на файла. След това се записва в една променлива броя на редовете, веднага след това се чете символа за разделение, като той не е важен какъв точно е. Накрая се прочита и запазва във втора променлива броя на колоните.

File.seekg(-9, std::ios::end);

File.read((char\*)&n, sizeof(int));

File.read((char\*)&ws, sizeof(char));

File.read((char\*)&m, sizeof(int));

Където ws е от символен тип и целта й е да бъде буфер за всеки символ за разстояние.

За четене на типа на всяка колона, което е необходимо за обхождането на всяка таблица, файловият указател за четене се слага в началото на последния ред на файла.

File.seekg(-5 \* m - 9, std::ios::end);

Където *m* е променливата, в която по-рано е бил записан броя на колоните. Целта на това е да се създадат две променливи, които да присвояват две различни позиции на файловия указател. Тяхната цел е да се използват за „*checkpoints“* – места от които укателят ще се връща след своята промяна.

check1 = File.tellg(), check2 = 0;

Чрез тези две променливи, указателят ще чете първо типа на колоната, след това ще се връща до началната позиция на клетката в таблицата, до която е стогнало обхождането. Чете се клетката и указателят се връща да чете типа на следващата колона. И така се заформя цикъла.

for (int i = 0; i < n; i++)

{

for (int j = 0; j < m; j++)

{

File.seekg(check1);

File.read((char\*)&tip, sizeof(int));

File.read((char\*)&ws, sizeof(char));

check1 = File.tellg();

File.seekg(check2, std::ios::beg);

.....

}

File.seekg(-5 \* m - 9, std::ios::end);

check1 = File.tellg();

}

**Глава 4: Реализация, тестване**

***Управление на паметта и алгоритми. Оптимизации.***

Както вече беше написано по-горе, в програмата не присъстват потребителски-дефинирани типове данни, като класове и структури. Програмата е сведена до минимално опростен и лесен за написване начин. Поради конструктори и деконструктори, които се намират в основните библиотеки на езика С++, и тяхното извикване по подразsбиране, когато дадена статична променлива излезе от даден блок, програмата няма специфични начини за манипулация на компютърната памет. Програмата се състои от множество блокове от код, които се извикват за да бъде изпълнено дадено действие т.е. програмата се състои от много блокове във вид на функции, които се сменят постоянно.

За оптимизации на проекта могат да се подобрят някои функции, така че те да работят малко по-бързо и да се избегне излишно копиране на таблиците. Вместо това да се работи динамично и с адреси на различните променливи.

**Глава 5. Заключение**

***5.1. Обобщение на изпълнението на началните цели***

Програмата се справя с оновните действия при работа с таблици, като отваряне, четене, запис, манипулация на данните, добавяне на таблица към списъка.

***5.2. Насоки за бъдещо развитие и усъвършенстване***

В бъдещи версии на програмата предстои да се добавят други важни за функции, които да допринесат за пълна и максимална работа с таблици. Така че програмата да стане като прототип, не пълна версия, на известни други програми, създадени от известни компании.