4. ТЕМА: Динамични структури от данни - динамични матрици

Работа с текстови файлове. Манипулации с низове. Динамични масиви от указатели към низове. Сортировка.

4.1. Задание

Да се разработи меню-програма за обработка на текстови файлове, която има следната функционалност:

- избор на файл за обработка (подменю):
  - от клавиатурата (без да се записва във временен файл) – въвежда се символ по символ до Ctrl/Z;
  - от текстови файл с произволно избрано име от потребителя;
- Създаване на речник на използваните думи, като се преброи колко пъти се среща всяка дума.
- Извеждане речника на екрана;
- Търсене на дума в речника;
- Сортиране на речника на използваните думи (по метод на мехурчето). Избор на начина на сортировка (подменю):
  - сортиране по азбучен ред;
  - сортиране по честотата (броя) на срещане на думата във файла;
- Записване на речника на използваните думи в текстов файл (името се задава в момента на записване). На първият ред във файла да се запише името на файла от който е създаден този речник. Избор на начина на записване (подменю):
  - всяка дума да се запише на нов ред, а след нея да се запише колко пъти се е срещала във файла;
  - броя на срещанията на думите да се запишат в отделен текстови файл, със същото име и разширение ”.cnt”;
- Зареждане на речник от текстови файл.
- Кодиране / декодиране на текста:
  - кодиране - Създаване на нов текстов файл, като думите са заменени с номерата им по речника, а всички разделители се запазват. Името на файла се задава от потребителя. На първия ред от файла да се запише името на файла с речника.;
  - декодиране – Прочита се името на речника от кодирания файл, след което речника се зарежда в оперативната памет, възстановява се текста във нов файл с произволно избрано от потребителя име;

4.2. Задължителни изисквания

- Файлът се обработва потоково – символ по символ.
- Програмата да може да работи с файлове с кирилица.
- Само веднъж да се отваря входния текстов файл: ***rewind()*.**
- Увеличаването на масива с думите да не е с ***realloc()*** за всяка нова дума, а на стъпки (за избягване на сегментацията на паметта).
- Проверката за разделите за думи се осъществява с автоматната функция от тема 2.

4.3. Допълнителни изисквания

Всеки от студентите получава допълнителна обработка, оформена като елемент от основното меню на разработеното приложение:

- определяне на най-дългата (най-късата) дума в речника.
- определяне на най-дългото (най-късото) изречение във файла;
- начин на поддържане на думите в речника – малки (главни) букви;

Всеки от студентите получава специфична обработка:

- начин на поддържане на думите в речника – малки (главни) букви;
- начин на сортиране – възходящ / низходящ азбучен ред;

продължителност: 2 лаб. упражнения (2х2 уч. часа)

1 упражнение:
- разделна компилация на началния проект
- създаване на речника
- сортиране на речника (един вариант)

2 упражнение:
- търсене на дума в речника
- записването на речник във файл
- кодиране и декодиране на речника.

4.4. Библиотечни функции за реализиране на проекта

Функции за работа с динамична памет (ALLOC.H)

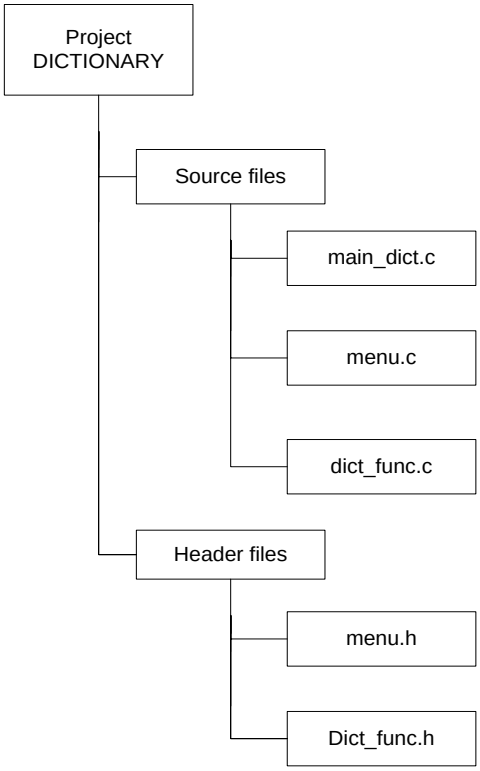
```
void *malloc(size_t size);
void *calloc(size_t nitems, size_t size);
void *realloc(void *block, size_t size);
void free(void *block);
```

Функции за работа с символни последователности (STRING.H)

```
int strcmp( const char *string1, const char *string2 );
char *strchr( const char *string, int c );
char *strcpy( char *strDestination, const char *strSource );
```

4.5. Обща структура на проекта

програмни модули



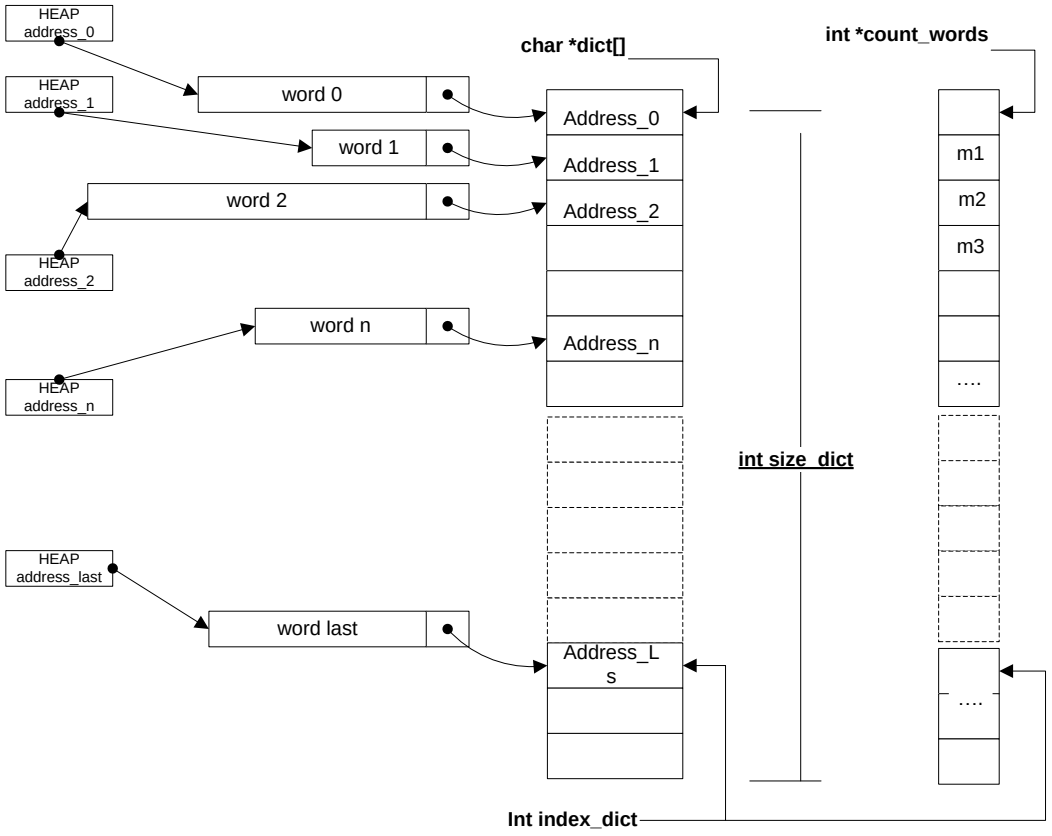
Съдържание на модулите (разработените модули и могат да се вземат от ТУК):

main_dict .c - съдържа главната функция **main** и текстовете на менютата
menu.c - функцията за избор на елемент от меню (с една и съща функция да се извеждат и подменютата – възможно е да се използува и функцията от предишните упражнения);
dict_func.c - съдържа дефинициите на всички функции за обработка на текста и създаване на речника (част от функциите за с пълен код, а останалите с описан алгоритъм за необходимата обработка – те се реализират в часовете определени за провеждане на упражненията);

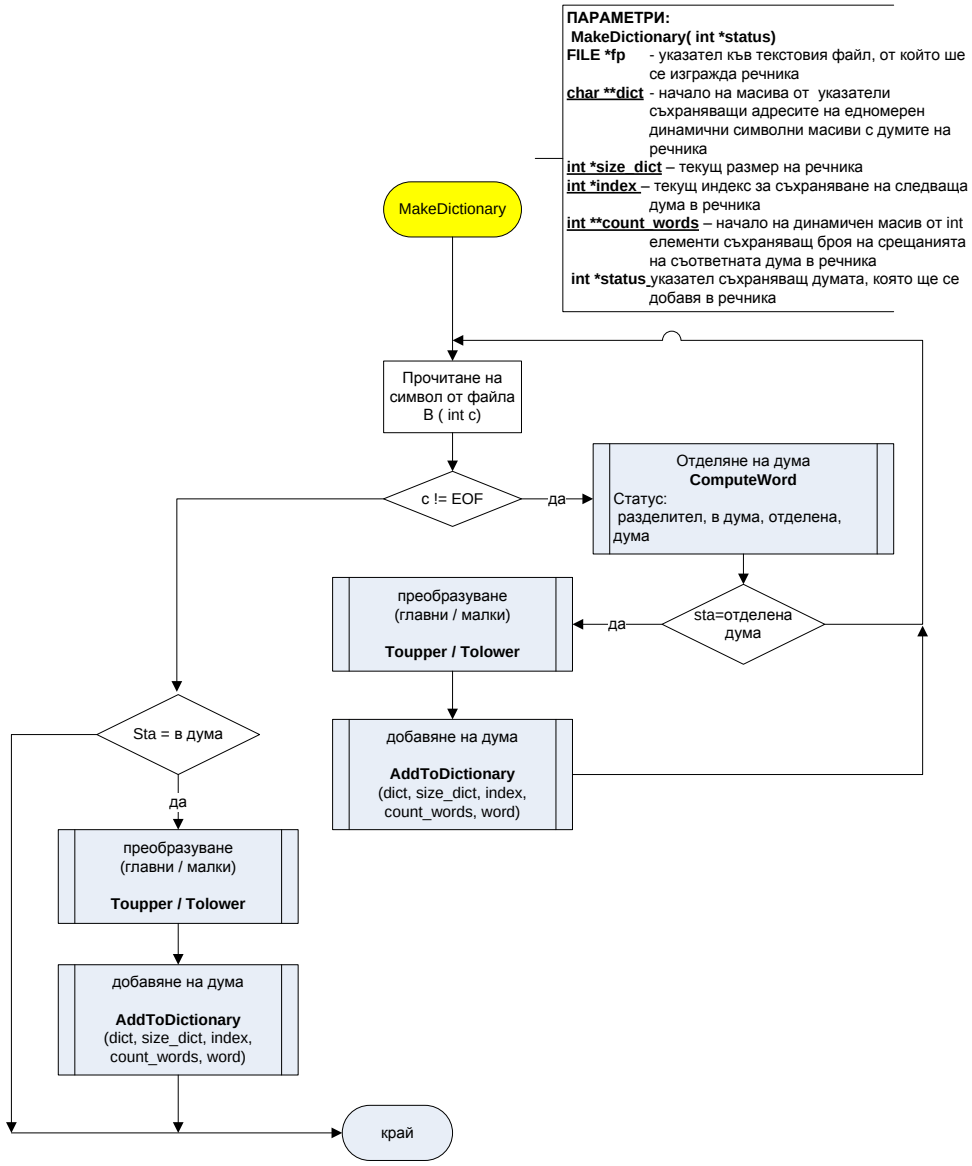
menu.h - съдържа прототипа на функцията **menu()**
dict_func.h - съдържа прототипите на функции за обработка на текста и създаване на речника;

Структура на данните

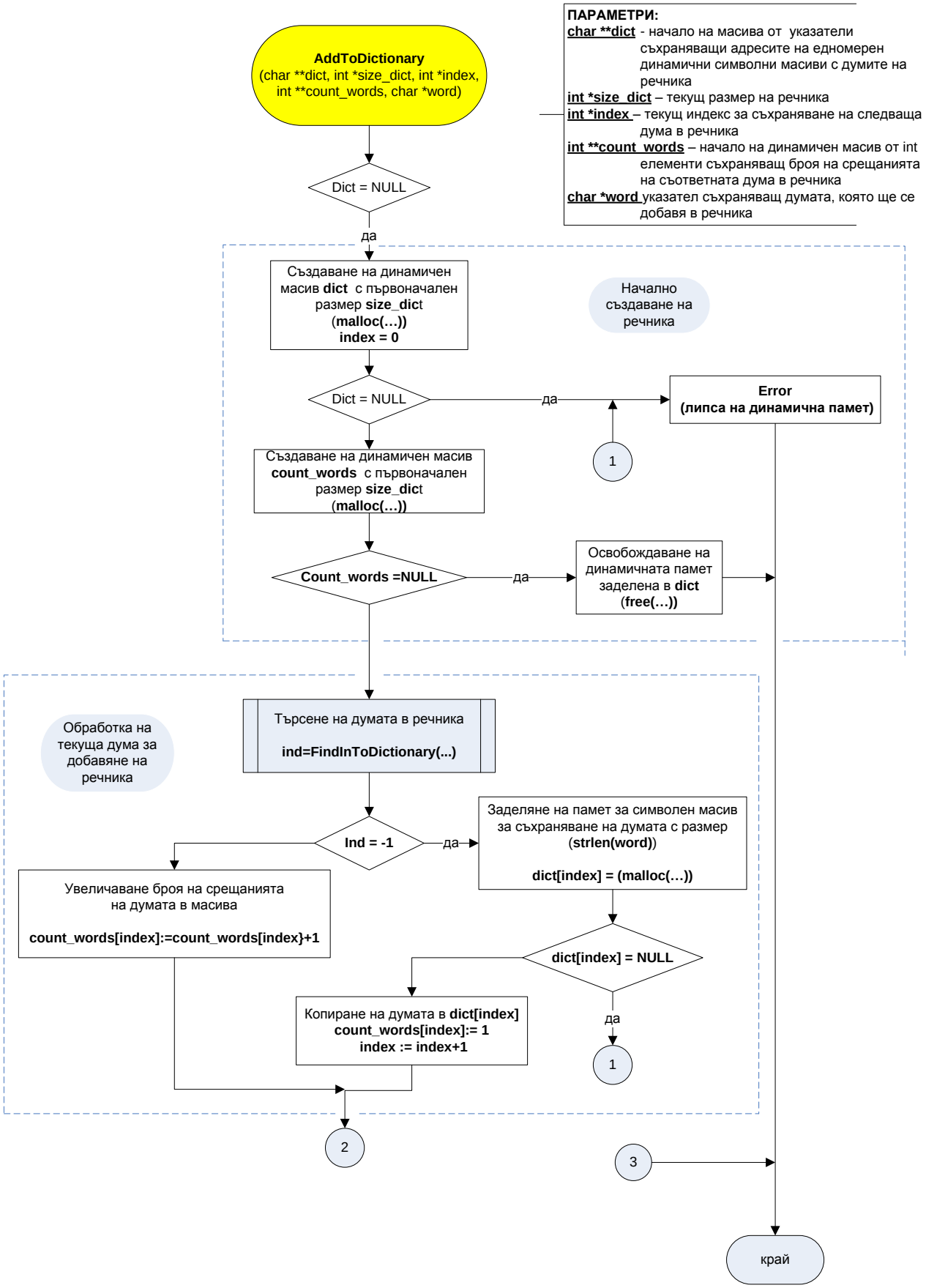
char *dict[] - динамичен масив от указатели,съхраняващ адресите на символни масив, съхраняващи думите на речника;
Int *count_word - динамичен масив от int стойности, съхраняващ броя на срещанията на думата във файла;
Int index_dict - актуален индекс за добавяне на нова дума в речника;
Int size_dict - текущ размер на масивите;

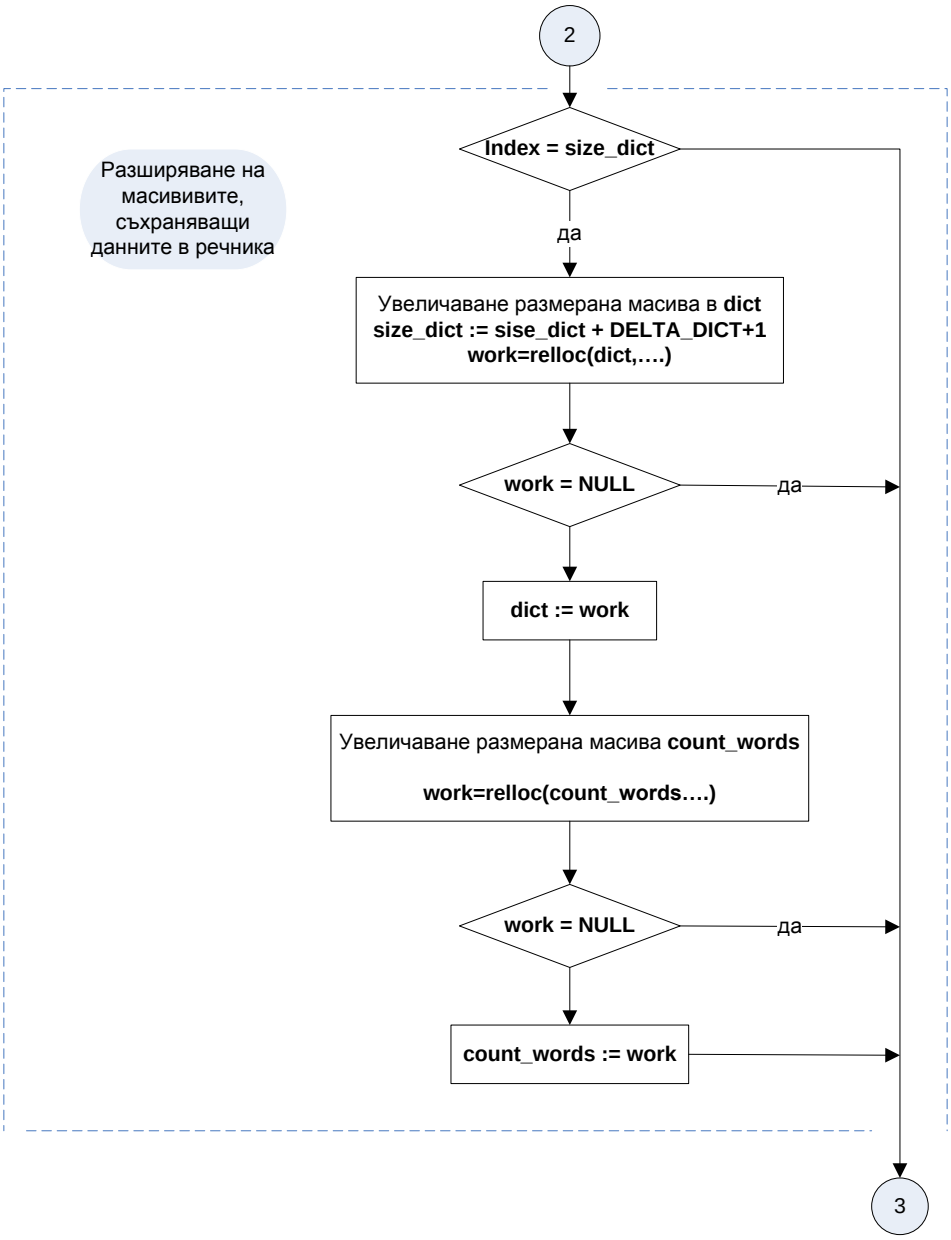


4.6. Алгоритми на функции за „създаване на речник” и добавяне на дума в речника”



да


AddToDictionary
(char **dict, int *size_dict, int *index, int **count_words, char *word)
ПАРАМЕТРИ:
char **dict - начало на масива от указатели съхраняващи адресите на едномерен динамични символни масиви с думите на речника
int *size_dict – текущ размер на речника
int *index – текущ индекс за съхраняване на следваща дума в речника
int **count_words – начало на динамичен масив от int елементи съхраняващ броя на срещанията на съответната дума в речника
char *word указател съхраняващ думата, която ще се добавя в речника
Dict = NULL
да
Начално създаване на речника
Създаване на динамичен масив dict с първоначален размер size_dict (malloc(…)) index = 0
Dict = NULL
да
Error (липса на динамична памет)
1
Създаване на динамичен масив count_words с първоначален размер size_dict (malloc(…))
Count_words =NULL
да
Освобождаване на динамичната памет заделена в dict (free(…))
Обработка на текуща дума за добавяне на речника
Търсене на думата в речника
ind=FindInToDictionary(...)
Ind = -1
да
Заделяне на памет за символен масив за съхраняване на думата с размер (strlen(word))
dict[index] = (malloc(…))
Увеличаване броя на срещанията на думата в масива
count_words[index]:=count_words[index}+1
dict[index] = NULL
да
Копиране на думата в dict[index]
count_words[index]:= 1
index := index+1
1
2
3
край


2
Разширяване на масививите, съхраняващи данните в речника
Index = size_dict
да
Увеличаване размерана масива в dict
size_dict := sise_dict + DELTA_DICT+1
work=relloc(dict,….)
work = NULL
да
dict := work
Увеличаване размерана масива count_words
work=relloc(count_words….)
work = NULL
да
count_words := work
3