ТЕХНИЧЕСКИ УНИВЕРСИТЕТ - ВАРНА

Факултет по изчислителна техника и автоматизация Катедра „Компютърни системи и технологии”

СЕМЕСТРИАЛНА ДОМАШНА РАБОТА

по дисциплина „Базово програмиране”

на тема: Информационна система „Обменни бюра”

Вариант 24

Изготвил: Михаел Митев Проверил: проф. М. Карова

Специалност: Компютърни системи и технологии

Група: 1б

Факултетен номер: 23621516

2024

Съдържание

I. Задание на проекта

Информационна система Обменни бюра

Да се напише компютърна програма, реализираща информационна система, която поддържа обменно бюро. Програмата съхранява и обработва данни за търгуваните валути (наименование на валутата, код по ISO, курс продава (лв.), курс купува (лв.), наличност, максимална сума за обмен при една транзакция (купува или продава), търгува ли се (да, не)). Максималният брой поддържани валути е 50.

Базова задача

A. Меню за избор на функциите от програмата

Функции от програмата са:

B. Добавяне на валути:

a. Добавянето трябва да позволява да се въвеждат различен брой нови валути като се допълват в масива. Не трябва да превишава максималния брой поддържани валути (50).

Пример: Добавяне на списък с валути. Въвежда се цяло число n, което позволява да се въведат n на брой валути. n не може да надвишава свободните елементи в масива с валути.

C. Извеждане на екрана

a. Извеждане на всички валути в оформен вид

D. Търсене и извеждане на екрана:

a. Намиране и извеждане на валутата с най-ниска наличност;

b. Търсене и извеждане на валута по въведен от клавиатурата код по ISO.

E. Подреждане на основния масив с валути, без да се извежда на екрана:

a. Подреждане на валутите в намаляващ ред на предвидения от обменното бюро марж (разлика между курс продава и курс купува)

F. Управление на файл:

a. Извеждане на масива с валути във файл (двоичен)

b. Въвеждане на масива с валути от файл (двоичен)

Допълнение първо (+ базова задача)

G. Създайте подменю, в което се влиза от основното, с нови функции за:

a. Извеждане на валутите, които се търгуват, в подреден ред по наименованието на валутата, без да се променя основния масив.

b. Търсене и извеждане на валутите, чиято наличност е над въведена от клавиатурата стойност и чийто марж (разлика между курс продава и курс купува) е под въведен от клавиатурата процент.

Допълнение второ (+ базова задача)

H. Покупко-продажба на валута (да се предвидят съответни позиции в менюто на програмата за извикването на тези функционалности).

a. Продажбата/Покупката на валутата се осъществява с въвеждане на код по ISO

Допълнение трето (+ базова задача)

I. Данните в програмата да се попълват автоматично от файл при стартиране и да се записват автоматично във файл при затваряне на програмата

II. Анализ на решението

1. Константи, Изброими типове и Структури за данните

const int max_valuti = 50; - максимален брой валути, които поддържата програмата на обменното бюро

Структура	Обяснение	Примерни стойности

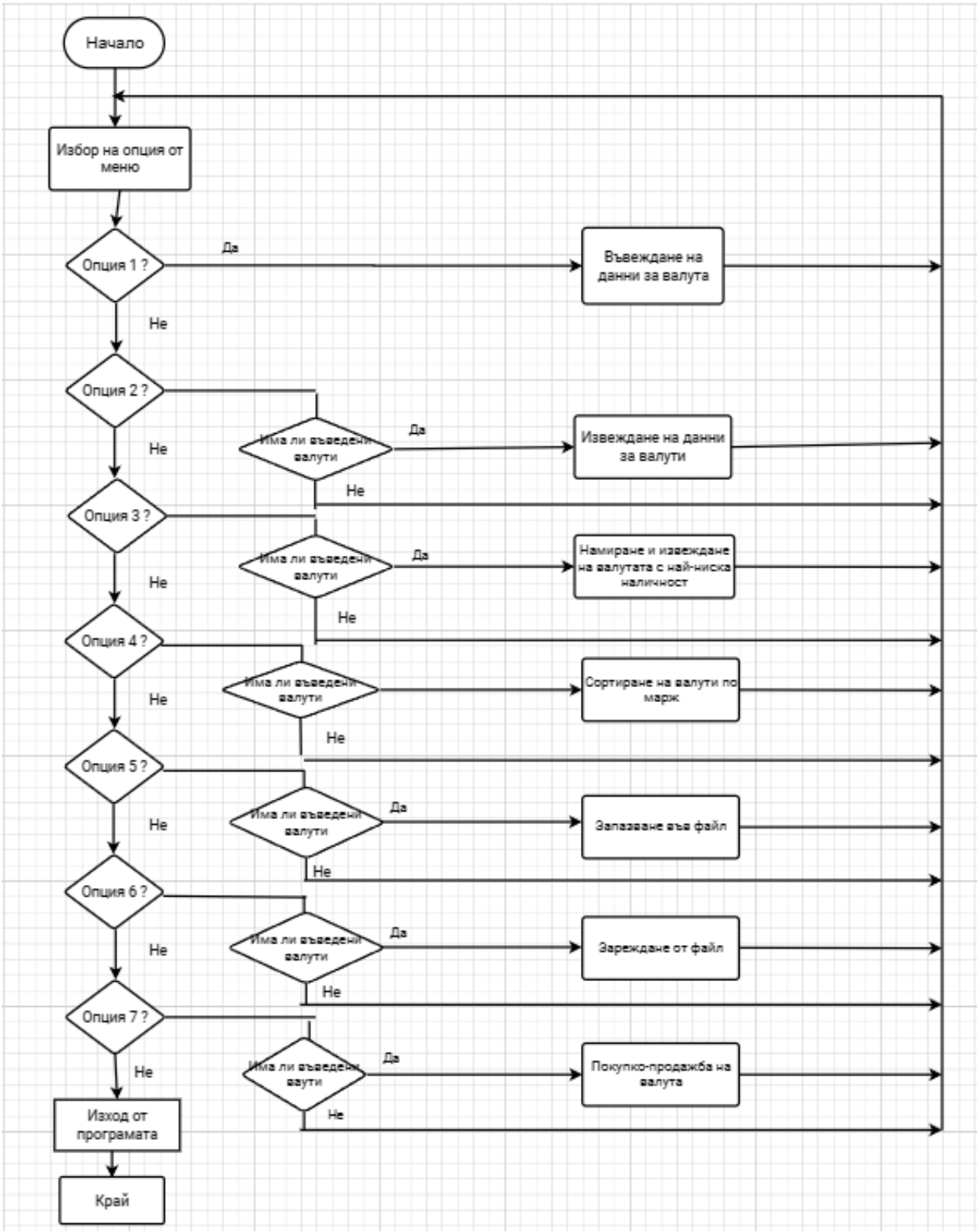
struct Valuta {
string ime_valuta;
string isoCode;
double kurs_prodava;
double kurs_kupuva;
double nalichnost;
double maksSumazaObmen;
bool turguva_se;
};
	ime_valuta - име на валутата
	isoCode - стандартен трибуквен код за парична единица
	kurs_prodava - продажна цена на валутата
	kurs_kupuva - цена на която обменното бюро закупува валутата
	nalichnost - каква сума е налична от дадена валута
	MaksSumazaObmen - максимална сума за обмен при една транзакция
	turguva_se - булиева променлива дали вълутата се търгува
		- Evro
		- EUR
		- 2.05
		- 1.95
		- 7000
		- 1500
		- 1

2. Реализация на условие А

2.1. Алгоритъм Меню с избор на функциите от програмата

Алгоритъм за меню с избор на функциите, който извежда на потребителя различните функционалности, като въведа своя избор от основно меню. Главният алгоритъм се извиква чрез функцията int main(), а подалгоритъм за менюто създаваме с do-while цикъл, в който има switch за различните опции.

2.2 Блок схема на алгоритъма


Начало
Избор на опция от меню
Опция 1 ?
Да
Въвеждане на данни за валута
Не
Опция 2 ?
Има ли въведени валути
Да
Извеждане на данни за валути
Не
Не
Опция 3 ?
Има ли въведени валути
Да
Намиране и извеждане на валутата с най-ниска наличност
Не
Не
Опция 4 ?
Има ли въведени валути
Сортиране на валути по марж
Не
Не
Опция 5 ?
Има ли въведени валути
Да
Запазване във файл
Не
Не
Опция 6 ?
Има ли въведени валути
Да
Зареждане от файл
Не
Не
Опция 7 ?
Има ли въведени валути
Да
Покупко-продажба на валута
Не
Не
Изход от програмата
Край

2.3. Функция, с която е реализиран алгоритъма

Int main()

2.3.1. Входни данни на функцията

Няма входни данни.

2.3.2. Изходни данни на функцията или данни, които се извеждат

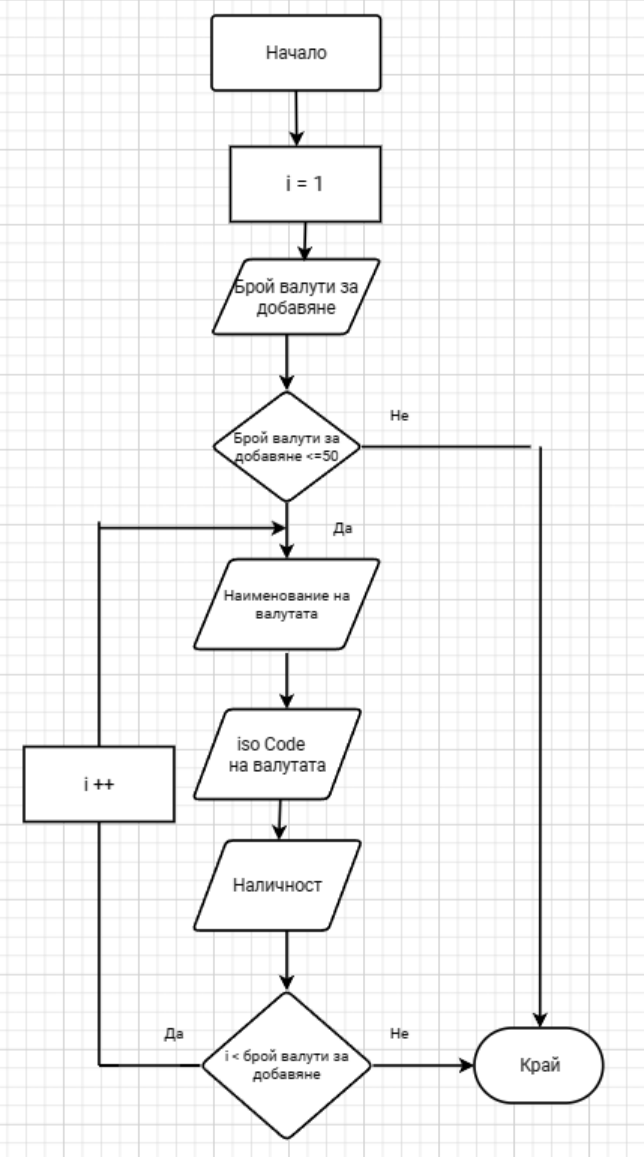
Спрямо избраната опция от менюто, функцията извиква съответните оператори и функции за изпълнение на съответните действия.

3. Реализация на условие B

3.1 Алгоритъм добавяне на валута

- Изисква от потребителя да въведе данните за валута, описани в структурата Valuta.

3.2. Блок схема на алгоритъма



3.3. Функция, с която е реализиран алгоритъма

void dobavi_valuta(Valuta valuti[], int& broiValuti);

3.3.1. Входни данни на функцията

Valuta valuti[] - променлива от тип структура Valuta, в която се съхраняват данните за една валута

3.3.2. Изходни данни на функцията или данни, които се извеждат

Тази функция не връща резултат.

4. Реализация на условие C

4.1. Алгоритъм Извеждане на данни на валути

При добавяне на данни в масива за валути, те се извеждат в табличен вид, като преди това се проверява, дали има въведени вече валути.

4.2. Блок схема на алгоритъма

4.3. Функция, с която е реализиран алгоритъма

void pokaji_valuti(const Valuta valuti[], int broiValuti)

4.3.1. Входни данни на функцията

const Valuta valuti[] - списък с валутите

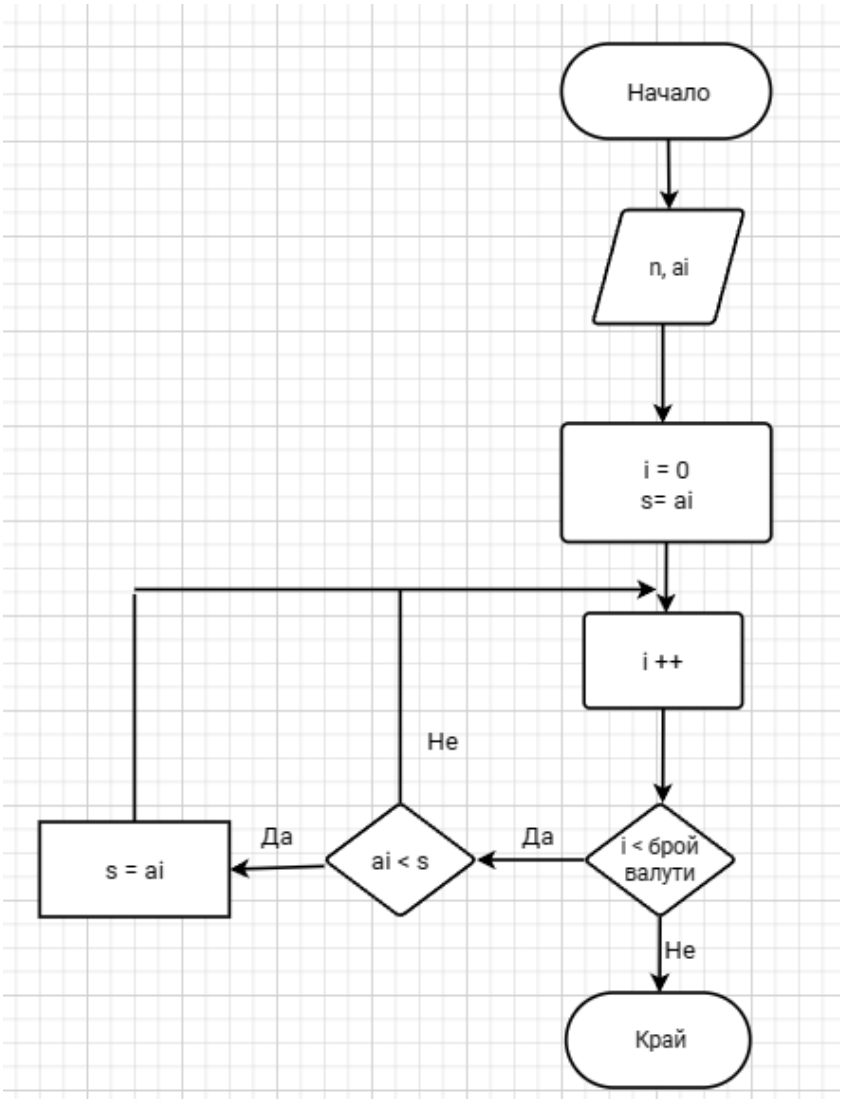
int broiValuti - размер на масива (брой на валутите в масива)

5. Реализация на условие D

5.1. Алгоритъм намиране и извеждане на валутата с най-ниска наличност

Намиране и извеждане на валутата с най-ниска наличност от всички въведени. Като първата я задаваме като валута с най-ниска наличност и проверяваме всяка следващата, съответно ако е с по-ниска наличност тя става валутата с най-ниска наличност.

5.2. Блок схема на алгоритъма


Начало
n, ai
i = 0
s= ai
i ++
Не
Да
Да
s = ai
ai < s
i < брой валути
Не
Край

5.3. Функция с която е реализиран алгоритъма

void nai_niska_nalichnost(const Valuta valuti[], int broiValuti);

5.3.1. Входни данни на функцията

const Valuta valuti[] - списък с валутите

int broiValuti - размер на масива (брой на валутите в масива)

6. Реализация на условие Е

6.1. Алгоритъм за сортиране на валути по марж

Подреждане на основния масив с валути, без да се извежда на екрана: Подреждане на валутите в намаляващ ред на предвидения от обменното бюро марж (разлика между курс продава и курс купува).

6.2. Блок схема на алгоритъма

6.3. Функция с която е реализиран алгоритъма

void podredba_marj(Valuta valuti[], int broiValuti);

6.3.1. Входни данни на функцията

const Valuta valuti[] - списък с валутите

int broiValuti - размер на масива (брой на валутите в масива)

7. Реализация на условие F

7.1. Алгоритъм Извеждане на масива с валути във файл

Данните с валутите се записват в текстов файл чрез избор от главното меню. Данните за валутите се съхраняват всяка на отделен ред. Записването се реализира чрез функцията void zapazi_fail(const Valuta valuti[], int broiValuti);

7.2. Функция с която е реализиран алгоритъма

void zapazi_fail(const Valuta valuti[], int broiValuti);

7.2.1. Входни данни на функцията

const Valuta valuti[] - списък с валутите

int broiValuti - размер на масива (брой на валутите в масива)

8. Реализация на условие … - допълнение първо

8.1. Анализ на алгоритъма, който трябва да се реализира

Да се създаде подменю, в което се влиза от основното, с нови функции за:

a. Извеждане на валутите, които се търгуват, в подреден ред по наименованието на валутата, без да се променя основния масив. Създаваме 2 for цикъла, чрез които да проверим наименованието на всеки елемент от масива. Създаваме променлива от тип Valuta в която запазваме първото наименование.

b. Търсене и извеждане на валутите, чиято наличност е над въведена от клавиатурата стойност и чийто марж (разлика между курс продава и курс купува) е под въведен от клавиатурата процент. Чрез изведена формула проверяваме дали е изпълнено условието и съответно ако е изпълнено условието се извеждат отговарящите на изискванията валути.

8.2. Функция с която е реализиран алгоритъма

a. void podredba_naimenovanie(Valuta valuti[], int broiValuti)

b. void namirane_nai_skupi_valuti_isoCode(const Valuta valuti[], int broiValuti)

8.2.1. Входни данни на функцията

const Valuta valuti[] - списък с валутите

int broiValuti - размер на масива (брой на валутите в масива)

9. Реализация на условие … - допълнение второ

9.1. Анализ на алгоритъма, който трябва да се реализира

Покупко-продажба на валута (да се предвидят съответни позиции в менюто на програмата за извикването на тези функционалности). В началото се въвежда iso Code на валутата, която потребителя иска да закупи, при покупка стойността на закупената валута намалява и първата валута в списъка за левове съответно се увеличава.

9.2. Функция с която е реализиран алгоритъма

void izvurshvane_tranzakciq(Valuta valuti[], int broiValuti);

9.2.1. Входни данни на функцията

Valuta valuti[] - списък с валутите

int broiValuti - размер на масива (брой на валутите в масива)

10. Реализация на допълнение трето

10.1. Анализ на алгоритъма, който трябва да се реализира

Данните в програмата да се попълват автоматично от файл при стартиране и да се записват автоматично във файл при затваряне на програмата. В началото на main функцията трябва да се използва функцията zaredi_fail(valuti, broiValuti);

10.2. Функция с която е реализиран алгоритъма

zaredi_fail(valuti, broiValuti);

10.3.1. Входни данни на функцията

valuti - списък с валутите

int broiValuti - размер на масива (брой на валутите в масива)

III. Примерно действие на програмата

1. Условие А

1.1. Снимка на изгледа с примерни входни данни

1.2. Входни данни

Въвеждане на опция от главното меню. При въвеждане на опция са възможни следните ситуации:

А) Въвеждане на опция < 1 – в този случай т. к. променливата съхраняваща опцията е беззнакова, ще се преизчисли стойността от дясната положителната граница на диапазона и ще се получи положително число

```
1. Dobavi valuta
2. Pokaji valutite
3. Tursene i izvejdane na ekrana
4. Sortirai po marj v namalqvasht red.
5. Upravlenie na faila
6. Nameri valuti s po-golqma stoinost ot vuvedenata i chiito marj e pod wuweden ot klawiaturata procent.
7. Pokupko-prodajba na valuta.
0. Izhod
Vuvedete vashiq izbor.
-3
Nevaliden izbor. Molq opitaite otnovo.
```

Б) Въвеждане на опция > последният номер на опция – в този случай, програмата ще поиска от потребителят да въведе отново опция, т. к. има предвидена валидация на въвеждането.

```
1. Dobavi valuta
2. Pokaji valutite
3. Tursene i izvejdane na ekrana
4. Sortirai po marj v namalqvasht red.
5. Upravlenie na faila
6. Nameri valuti s po-golqma stoinost ot vuvedenata i chiito marj e pod wuweden ot klawiaturata procent.
7. Pokupko-prodajba na valuta.
0. Izhod
Vuvedete vashiq izbor.
8
Nevaliden izbor. Molq opitaite otnovo.
```

- В) Въвеждане на опция в интервала [1 – предпоследната опция] – в този случай програмата изпълнява действията, заложени за изпълнение при дадената опция

```
1. Dobavi valuta
2. Pokaji valutite
3. Tursene i izvejdane na ekrana
4. Sortirai po marj v namalqvasht red.
5. Upravlenie na faila
6. Nameri valuti s po-golqma stoinost ot vuvedenata i chiito marj e pod wuweden ot klawiaturata procent.
7. Pokupko-prodajba na valuta.
0. Izhod
Vuvedete vashiq izbor.
4
Nqma nalichni valuti.
```

Г) Въвеждане на последната опция – в този случай това е опция „Изход“, което води до прекратяване на изпълнението на цялата програма.

```
1. Dobavi valuta
2. Pokaji valutite
3. Tursene i izvejdane na ekrana
4. Sortirai po marj v namalqvasht red.
5. Upravlenie na faila
6. Nameri valuti s po-golqma stoinost ot vuvedenata i chiito marj e pod wuweden ot klawiaturata procent.
7. Pokupko-prodajba na valuta.
0. Izhod
Vuvedete vashiq izbor.
0
Izhod ot programata. Dannite se zapametqvat.
Nqma namereni predishni danni.

C:\Users\dsbex\source\repos\Semestrialna rabota\x64\Debug\Semestrialna rabota.exe (process 12612) exited with code 0.
To automatically close the console when debugging stops, enable Tools->Options->Debugging->Automatically close the conso
le when debugging stops.
Press any key to close this window . . .
```

2. Условие В

2.1. Примерни входни данни

Въвежда се броя на валутите, които потребителя иска да добави и след това се въвеждат данни за самата валута последователно. При въвеждането се спазват изискванията за въвеждане на данни спрямо типовете им! Предвидена е валидация на въвеждането за брой валути, като максималният брой е 50.

2.2. Снимка на изгледа с примерни данни

```
1. Dobavi valuta
2. Pokaji valutite
3. Tursene i izvejdane na ekrana
4. Sortirai po marj v namalqvasht red.
5. Upravlenie na faila
6. Nameri valuti s po-golqma stoinost ot vuvedenata i chiito marj e pod wuweden ot klawiaturata procent.
7. Pokupko-prodajba na valuta.
0. Izhod
Vuvedete vashiq izbor.
1
Vuvedete kolko valuti iskate da dobavite: 2
Dobavete danni za valuta: 1
Naimenovianie na valutata: Evro
ISO code: EUR
Kurs prodava: 1.9
Kurs kupuva: 1.78
Nalichnost: 10000
Maksimalna tranzakciq: 1500
Turguva li se(vuvedete 1 za da i 0 za ne): 1
Dobavete danni za valuta: 2
Naimenovianie na valutata: Levove
ISO code: BGN
Kurs prodava: 1
Kurs kupuva: 0.92
Nalichnost: 15000
Maksimalna tranzakciq: 2000
Turguva li se(vuvedete 1 za da i 0 za ne): 1
2valuti sa dobaveni uspeshno.
```

3. Условие C

3.1. Примерни входни данни

Извеждане на валутите в табличен вид на екрана. Не се изисква въвеждане на входни данни. Предвидена е проверка за празен масив с валути и изваждане на подходящо съобщение, ако няма въведени валути.

3.2. Снимка на изгледа с примерни данни

```
1. Dobavi valuta
2. Pokaji valutite
3. Tursene i izvejdane na ekrana
4. Sortirai po marj v namalqvasht red.
5. Upravlenie na faila
6. Nameri valuti s po-golqma stoinost ot vuvedenata i chiito marj e pod wuweden ot klawiaturata procent.
7. Pokupko-prodajba na valuta.
0. Izhod
Vuvedete vashiq izbor.
2
Vsichki valuti
ISO Code  Ime    Kurs prodava   Kurs kupuva    Smetka         Maks tranzakciq   Turguva se
EUR       Evro   1.9            1.78           10000          1500              1
BGN       Levove 1              0.92           15000          2000              1
```

4. Условие D

4.1. Примерни входни данни

Подреждане на валутите в намаляващ ред на предвидения от обменното бюро марж (разлика между курс продава и курс купува). Тази функция не отпечатва валутите, тъй като е така зададено по условие.

4.2. Снимка на изгледа с примерни данни

```
1. Dobavi valuta
2. Pokaji valutite
3. Tursene i izvejdane na ekrana
4. Sortirai po marj v namalqvasht red.
5. Upravlenie na faila
6. Nameri valuti s po-golqma stoinost ot vuvedenata i chiito marj e pod wuweden ot klawiaturata procent.
7. Pokupko-prodajba na valuta.
0. Izhod
Vuvedete vashiq izbor.
4
Valutite sa sortirani po marj v namalqvasht red.
```

5. Условие E

5.1. Примерни входни данни.

При избор на опция 5 излиза подменю за работа с файл, от където можем да запаметим записаните данни във файла или да изведем данните, които файлът съдържа.

5.2. Снимка на изгледа с примерни входни данни

```
1. Dobavi valuta
2. Pokaji valutite
3. Tursene i izvejdane na ekrana
4. Sortirai po marj v namalqvasht red.
5. Upravlenie na faila
6. Nameri valuti s po-golqma stoinost ot vuvedenata i chiito marj e pod wuweden ot klawiaturata procent.
7. Pokupko-prodajba na valuta.
0. Izhod
Vuvedete vashiq izbor.
5
--------------------------------------------------------------------------------
1. Izvejdane na masiva s valuti ot fail.
2. Vuvejdane na masiva s valuti ot fail.
```

6. Условие F

6.1. Снимка на изгледа с примерни входни данни

При избор на опция 6 от менюто се влиза в подменю, от което можем да изберем да изведем валутите, които се търгуват, в подреден ред по наименованието на валутата, без да се променя основния масив. Другата опция е търсене и извеждане на валутите, чиято наличност е над въведена от клавиатурата стойност и чийто марж (разлика между курс продава и курс купува) е под въведен от клавиатурата процент.

```
1. Dobavi valuta
2. Pokaji valutite
3. Tursene i izvejdane na ekrana
4. Sortirai po marj v namalqvasht red.
5. Upravlenie na faila
6. Nameri valuti s po-golqma stoinost ot vuvedenata i chiito marj e pod wuweden ot klawiaturata procent.
7. Pokupko-prodajba na valuta.
0. Izhod
Vuvedete vashiq izbor.
6
------------------------------------------------------
1. Izvejdane na valutite koito se turguvat v podreden red po naimenovanieto na valutata.
2. Valuti koito sa nad balans vuveden ot vas i marj s procent po-nisuk vuveden ot vas.
3. Vrushtane kum glavnoto menu.
1
Valutite sa podredeni po naimenovanie:
Vsichki valuti
ISO Code  Ime    Kurs prodava   Kurs kupuva    Smetka         Maks tranzakciq   Turguva se
EUR       evro   1.9            1.75           15000          1500              1
BGN       levove 1              0.92           10000          1500              1
```

```
Vuvedete stoinost za prag sprqmo koito da izlizat valutite s po-golqma nalichnost: 1500
Vuvedete prag za marja v procenti ot koito tezi s po maluk procent da se izpishat: 100
Valuta otgovarqshta na tezi kriterii e:
Naimenovanie: levove
ISO Code: BGN
Nalichnost: 10000
Marj: 8%
Valuta otgovarqshta na tezi kriterii e:
Naimenovanie: evro
ISO Code: EUR
Nalichnost: 15000
Marj: 7.89474%
```

7. Допълнение първо

7.1. Снимка на изгледа с примерни входни данни

Покупко-продажба на валута (да се предвидят съответни позиции в менюто на програмата за извикването на тези функционалности). а. Продажбата/Покупката на валутата се осъществява с въвеждане на код по ISO.

IV. ПРИЛОЖЕНИЕ 1

1. Изходен код на програмата

```
#include <iostream>

#include <fstream>

#include <string>

#include <iomanip>

#include <locale>


using namespace std;


const int max_valuti = 50;

int broiValuti = 0;


struct Valuta {
```

```
string ime_valuta;
string isoCode;
double kurs_prodava;
double kurs_kupuva;
double nalichnost;
double maksSumazaObmen;
bool turguva_se;
};

// Функция за добавяне на валута
void dobavi_valuta(Valuta valuti[], int& broiValuti);

// Функция за извеждане на всички валути
void pokaji_valuti(const Valuta valuti[], int broiValuti);

// Функция за извеждане на всички валути
void podmenu_tursene_izvejdane(Valuta valuti[], int broiValuti);

// Функция за намиране и извеждане на валутата с най-ниска наличност
void nai_niska_nalichnost(const Valuta valuti[], int broiValuti);

// Функция за търсене и извеждане на валута по въведен от клавиатурата код по ISO
void tusrsene_isoCode(const Valuta valuti[], int broiValuti);
```

```
// Функция за подреждане на основния масив с валути по марж
void podredba_marj(Valuta valuti[], int broiValuti);

// Функция за подреждане на основния масив с валути по наименование
void podredba_naimenovanie(Valuta valuti[], int broiValuti);

// Функция за управление на файл - записване на масива с валути във файл (двоичен)
void zapazi_fail(const Valuta valuti[], int broiValuti);

// Функция за управление на файл - зареждане на масива с валути от файл (двоичен)
void zaredi_fail(Valuta valuti[], int& broiValuti);

// Функция за намиране на валути с по-голяма наличност и по-нисък марж
void namirane_nai_skupi_valuti_isoCode(const Valuta valuti[], int broiValuti);

// Подменю за операции свързани с наименованието на валутите
void podmenu_naimenovanie_valuta(Valuta valuti[], int broiValuti);

// Подменю за операции свързани с управление на файловете
void podmenu_upravlenie_fail(Valuta valuti[], int broiValuti);

// Функция за извършване на транзакции - покупка/продажба на валута
```

```
void izvurshvane_tranzakciq(Valuta valuti[], int broiValuti);


int main()

{

setlocale(LC_ALL, "bulgarian");

// Масив от валути с максимален размер max_valuti

Valuta valuti[max_valuti];


zaredi_fail(valuti, broiValuti);


// Променлива за избор на опции в главното меню

int izbor;

do

{

// Извеждане на опциите в главното меню

cout << "1. Добави валута" << endl;

cout << "2. Покажи валутите" << endl;

cout << "3. Търсене и извеждане на екрана" << endl;

cout << "4. Соритрай по марж в намаляващ ред." << endl;

cout << "5. Управление на файла" << endl;

cout << "6. Намери валута с по-голяма стойност от въведената и чииго марж е под въведен от клавиатурата процент." << endl;

cout << "7. Покупко-продажба на валута." << endl;
```

```
cout << "0. Изход" << endl;

cout << "Въведете своя избор." << endl;

// Въвеждане на избора от потребителя

cin >> izbor;


// Обработка на избора с помощта на switch

switch (izbor)

{

case 1:

dobavi_valuta(valuti, broiValuti);

break;

case 2:

pokaji_valuti(valuti, broiValuti);

break;

case 3:

podmenu_tursene_izvejdane(valuti, broiValuti);

int r;

cin >> r;

switch (r)

{

case 1:

nai_niska_nalichnost(valuti, broiValuti);

break;
```

```
case 2:
tusrsene_isoCode(valuti, broiValuti);
break;
case 3:
cout << "Obratno kum menuto." << endl;
break;
default:
break;
}
break;
case 4:
podredba_marj(valuti, broiValuti);
break;
case 5:
podmenu_upravlenie_fail(valuti, broiValuti);
int m;
cin >> m;
switch (m)
{
case 1:
zapazi_fail(valuti, broiValuti);
break;
case 2:
```

```
zaredi_fail(valuti, broiValuti);
break;
default:
break;
}
break;
case 6:
podmenu_naimenovanie_valuta(valuti, broiValuti);
int z;
cin >> z;
switch (z)
{
case 1:
podredba_naimenovanie(valuti, broiValuti);
break;
case 2:
namirane_nai_skupi_valuti_isoCode(valuti, broiValuti);
break;
default:
break;
}
break;
case 7:
```

```
izvurshvane_tranzakciq(valuti, broiValuti);

break;

case 0:

cout << "Изход от програмата. Данните се запаметяват." << endl;

zapazi_fail(valuti, broiValuti);

break;

default:

cout << "Невалиден избор. Моля опитайте отново." << endl;

}

} while (izbor != 0); // Повтаряне на цикъла докато потребителят не въведе невалидна опция

return 0;

}


void dobavi_valuta(Valuta valuti[], int& broiValuti)

{

setlocale(LC_ALL, "bulgarian");

// Променлива за бройката на валутите, които потребителят иска да добави

int n;

cout << "Въведете колко валути искате да добавите: ";

// Въвеждане на броя на валутите, които потребителят иска да добави

cin >> n;


// Проверка дали броя на валутите е валиден
```

```
if (n <= 0)
{
cout << "Невалиден брой валути за добавяне. Моля въведете положително число." << endl;
return;
}

// Проверка дали общият брой валути след добавянето не надвишава максималния лимит
if (broiValuti + n > max_valuti)
{
cout << "Надвишава се максималния брой от позволените валути.";
return;
}

// Цикъл за въвеждане на данни за всяка добавяна валута
for (int i = broiValuti; i < broiValuti + n; ++i)
{
cout << "Добавете данни за валута: " << i + 1 << endl;

// Въвеждане на данни за валутата
cout << "Наименование на валутата: ";
cin >> valuti[i].ime_valuta;
cout << "ISO код: ";
cin >> valuti[i].isoCode;
```

```
cout << "Курс продава: ";
cin >> valuti[i].kurs_prodava;
cout << "Курс купува: ";
cin >> valuti[i].kurs_kupuva;
cout << "Наличност: ";
cin >> valuti[i].nalichnost;
cout << "Максимална транзакция: ";
cin >> valuti[i].maksSumazaObmen;
cout << "Търгува ли се(въведете 1 за да и 0 за не): ";
cin >> valuti[i].turguva_se;
}

// Увеличаване на брояча на валутите
broiValuti += n;

// Извеждане на съобщение за успешно добавени валути
cout << n << "валути са добавени успешно." << endl;
}

void pokaji_valuti(const Valuta valuti[], int broiValuti)
{
setlocale(LC_ALL, "bulgarian");
```

```
// Проверка дали има налични валути

if (broiValuti != 0)

{

// Извеждане на заглавен ред с имената на полетата в таблицата

cout << "Всички валути" << endl;

cout << left << setw(10) << "ISO код" << left << setw(7) << "Име" << left << setw(15) << "Курс продава"
<< left << setw(15)

<< "Курс купува" << left << setw(15) << "Сметка" << left << setw(18) << "Макс транзакция" << left <<
setw(15) << "Търгува се" << endl;

}

else

{

// Извеждане на съобщение, че няма налични валути и приключване на функцията

cout << "Няма налични валути." << endl;

cout << endl;

return;

}


// Извеждане на информацията за всяка валута в таблица

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

{

cout << left << setw(10) << valuti[i].isoCode << left << setw(7) << valuti[i].ime_valuta << left << setw(15)
<< valuti[i].kurs_prodava
```

```
<< left << setw(15) << valuti[i].kurs_kupuva << left << setw(15) << valuti[i].nalichnost << left << setw(18)
<< valuti[i].maksSumazaObmen << left << setw(15) << valuti[i].turguva_se << endl;


}


}


void podmenu_tursene_izvejdane(Valuta valuti[], int broiValuti)
{
setlocale(LC_ALL, "bulgarian");
cout << "-------------------------------------------------------------------------------" << endl;
cout << "1. Намиране и извеждане на валутата с най-ниска наличност." << endl;
cout << "2. Търсене и извеждане на валута по въведен от клавиатурата код по ISO." << endl;
cout << "3. Връщане към главното меню." << endl;
}


void podmenu_upravlenie_fail(Valuta valuti[], int broiValuti)
{
setlocale(LC_ALL, "bulgarian");
cout << "-------------------------------------------------------------------------------" << endl;
cout << "1. Извеждане на масива с валути от файла." << endl;
cout << "2. Въвеждане на масива с валути от файла." << endl;
}
```

```
void nai_niska_nalichnost(const Valuta valuti[], int broiValuti)
{
setlocale(LC_ALL, "bulgarian");
// Проверка дали има налични валути
if (broiValuti == 0)
{
cout << "Няма налични валути." << endl;
return;
}

// Инициализация на минималната стойност и индекса на валутата с най-ниска наличност
double minstoinost = valuti[0].nalichnost;
int minindex = 0;

// Намиране на валутата с най-ниска наличност в масива
for (int i = 1; i < broiValuti; ++i)
{
if (valuti[i].nalichnost < minstoinost)
{
minstoinost = valuti[i].nalichnost;
minindex = i;
}
```

```
}

// Извеждане на информация за валутата с най-ниска наличност
cout << "Валути с най-ниска наличност: " << endl;
cout << "Наименование: " << valuti[minindex].ime_valuta << endl;
cout << "ISO код: " << valuti[minindex].isoCode << endl;
cout << "Сметка: " << valuti[minindex].nalichnost << endl;
}

void tusrsene_isoCode(const Valuta valuti[], int broiValuti)
{
setlocale(LC_ALL, "bulgarian");
// Проверка дали има налични валути
if (broiValuti == 0)
{
cout << "Няма налични валути." << endl;
return;
}

// Потребителски вход за търсене на ISO Code
string turseneIsoCode;
cout << "Въведете ISO код, който да се търси: ";
cin >> turseneIsoCode;
```

```
// Търсене на валута по въведения ISO Code
for (int i = 0; i < broiValuti; ++i)
{
if (valuti[i].isoCode == turseneIsoCode)
{
// Извеждане на информация за намерената валута
cout << "Вашата валута: " << endl;
cout << "Наименование на валута: " << valuti[i].ime_valuta << endl;
cout << "ISO код: " << valuti[i].isoCode << endl;
cout << "Наличност: " << valuti[i].nalichnost << endl;
return;
}
}
// Извеждане на съобщение, ако валутата не е намерена
cout << "Валута с ISO код '" << turseneIsoCode << "' не е намерена." << endl;
}

void podredba_marj(Valuta valuti[], int broiValuti)
{
setlocale(LC_ALL, "bulgarian");
// Проверка дали има налични валути
if (broiValuti == 0)
```

```
{
cout << "Няма налични валути." << endl;
return;
}

// Сортиране на валутите по разлика между курса на продажба и купува
for (int i = 0; i < broiValuti - 1; ++i)
{
for (int j = 0; j < broiValuti - i - 1; ++j)
{
double marjA = valuti[j].kurs_prodava - valuti[j].kurs_kupuva;
double marjB = valuti[j + 1].kurs_prodava - valuti[j + 1].kurs_kupuva;

// Проверка за намаляващ ред на разликата между курса на продажба и купува
if (marjA < marjB)
{
// Размяна на местата на валутите
Valuta temp = valuti[j];
valuti[j] = valuti[j + 1];
valuti[j + 1] = temp;
}
}
}
```

```
cout << "Валутите са сортирани по марж в намаляващ ред." << endl;
}

void zapazi_fail(const Valuta valuti[], int broiValuti)
{
setlocale(LC_ALL, "bulgarian");
ofstream outFile("valuti.dat, ios::binary | ios::trunc");

if(!outFile)
{
cout << "Няма намерени предишни данни." << endl;
return;
}
outFile.write((char*)&broiValuti, sizeof(broiValuti));
outFile.write((char*)&valuti, sizeof(valuti) * broiValuti);

outFile.close();
cout << "Информацията е запазена успешно." << endl;
}

void zaredi_fail(Valuta valuti[], int& broiValuti)
{
setlocale(LC_ALL, "bulgarian");
```

```
// Отваряне на файл за четене в бинарен режим
ifstream inFile("valuti.dat", ios::binary);

// Проверка дали файла е успешно отворен
if (!inFile)
{
cout << "Няма предишни данни." << endl;
return;
}

// Четене на броя на записаните валути
inFile.read((char*)&broiValuti, sizeof(broiValuti));

// Четене на самите валути от файла
inFile.read((char*)&valuti, sizeof(valuti) * broiValuti);

// Затваряне на файла
inFile.close();

// Известие за успешно зареждане на данните от файла
cout << "Данните са заредени от файла." << endl;
}
```

```
void namirane_nai_skupi_valuti_isoCode(const Valuta valuti[], int broiValuti)

{

setlocale(LC_ALL, "bulgarian");


// Проверка дали има налични валути

if (broiValuti == 0)

{

cout << "Няма налични валути." << endl;

return;

}


// Въвеждане на стойност за праг на наличност

double stoinost_prag;

cout << "Въведете стойност за праг, спрямо който да излизат валутите с по-голяма наличност: ";

cin >> stoinost_prag;


// Въвеждане на праг за марж в проценти

double prag_marj;

cout << "Въведете праг за маржа в проценти, от който тези с по-малък процент да се изпишат: ";

cin >> prag_marj;


// Брояч за валутите, отговарящи на критериите

int valuta_kriterii = 0;
```

```
// Обхождане на валутите и извеждане на тези, които отговарят на критериите
for (int i = 0; i < broiValuti; ++i)
{
if (valuti[i].nalichnost > stoinost_prag && (((valuti[i].kurs_prodava - valuti[i].kurs_kupuva) / valuti[i].kurs_prodava) * 100) < prag_marj)
{
cout << "Валуита отговаряща на тези критерии е: " << endl;
cout << "Наименование: " << valuti[i].ime_valuta << endl;
cout << "ISO код: " << valuti[i].isoCode << endl;
cout << "Наличност: " << valuti[i].nalichnost << endl;
cout << "Марж: " << (((valuti[i].kurs_prodava - valuti[i].kurs_kupuva) / valuti[i].kurs_prodava) * 100) << "%" << endl;
valuta_kriterii += 1;
}
}

// Извеждане на съобщение, ако няма валути, отговарящи на критериите
if (valuta_kriterii == 0) {
cout << "Няма валути отговарящи на критериите" << endl;
}
}

void podmenu_naimenovanie_valuta(Valuta valuti[], int broiValuti)
```

```
{
setlocale(LC_ALL, "bulgarian");
cout << "-----------------------------------------------------" << endl;
cout << "1. Извеждане на валутите, които се търгуват в подреден ред по наименованието на валутата." << endl;
cout << "2. Валути, които са над баланс въведен от вас и марж с процент по-нисък въведен от вас." << endl;
cout << "3. Връщане към главното меню." << endl;
}

void podredba_naimenovanie(Valuta valuti[], int broiValuti)
{
setlocale(LC_ALL, "bulgarian");
// Създаване на временен масив за сортиране
Valuta* tempValuti = new Valuta[broiValuti];
for (int i = 0; i < broiValuti; ++i)
{
tempValuti[i] = valuti[i];
}

// Сортиране на временния масив по име на валутата
for (int i = 0; i < broiValuti - 1; ++i)
{
for (int j = 0; j < broiValuti - i - 1; ++j)
```

```
{
if (tempValuti[j].ime_valuta > tempValuti[j + 1].ime_valuta)
{
Valuta tem = tempValuti[j];
tempValuti[j] = tempValuti[j + 1];
tempValuti[j + 1] = tem;
}
}
}

// Извеждане на сортираните валути по име
cout << "Валутите са подредени по наименование:" << endl;
pokaji_valuti(tempValuti, broiValuti);

// Освобождаване на заделената памет за временния масив
delete[] tempValuti;
}

void izvurshvane_tranzakciq(Valuta valuti[], int broiValuti)
{
setlocale(LC_ALL, "bulgarian");
// Въвеждане на ISO код на валутата
string isoCode;
```

```
cout << "Въведете ISO код на валутата: ";

cin >> isoCode;


// Търсене на валутата в масива

int index = -1;

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

{

if (valuti[i].isoCode == isoCode)

{

index = i;

break;

}

}


// Проверка дали валутата е намерена

if (index != -1)

{

// Проверка дали валутата се търгува

if (valuti[index].turguva_se)

{

double suma;


// Определяне на действието - покупка или продажба
```

```
if (valuti[index].kurs_prodava > valuti[index].kurs_kupuva)

{

cout << "Въведете сума за покупка: ";

cin >> suma;

}

else

{

cout << "Въведете сума за продажба: ";

cin >> suma;

}


// Проверка дали сумата е валидна и дали има достатъчна наличност

if (suma <= valuti[index].maksSumazaObmen && suma <= valuti[index].nalichnost)

{

// Извършване на транзакцията

valuti[index].nalichnost -= suma;

double razlika = valuti[index].kurs_prodava - valuti[index].kurs_kupuva;

double sumaBGN = suma / valuti[index].kurs_kupuva;

valuti[0].nalichnost += sumaBGN;


// Извеждане на съобщение за успешна транзакция

cout << "Успешна " << (valuti[index].kurs_prodava > valuti[index].kurs_kupuva ? "покупка" : "продажба") << " на " << suma << " " << valuti[index].ime_valuta << "." << endl;
```

```
}
else
{
// Извеждане на съобщение за грешка
cout << "Грешка: Невалидна сума или недостатъчна наличност." << endl;
}
}
else
{
// Извеждане на съобщение, че валутата не се търгува
cout << "Валутата не се търгува." << endl;
}
}
else
{
// Извеждане на съобщение, че валутата не е намерена
cout << "Валутата с ISO код '" << isoCode << "' не е намерена." << endl;
}
}
```