Картина, която съдържа текст, графична колекция

Описанието е генерирано автоматичноТЕХНИЧЕСКИ УНИВЕРСИТЕТ – ВАРНА

Факултет по изчислителна техника и автоматизация

Катедра „Компютърни науки и технологии“

**СЕМЕСТРИАЛНА ДОМАШНА РАБОТА**

по дисциплината „Базово програмиране”

на тема: „Продажба на компютри”

Вариант 276

|  |  |
| --- | --- |
| Изготвил: Ивайло Илиев | Проверил: |
| Специалност: КСТ |  |
| Група:4Б |  |
| Факултетен номер:22621518 |  |

2023

Съдържание

[I. Задание на проекта 3](#_Toc120821980)

[II. Анализ на решението 4](#_Toc120821981)

[1. Структура за данните в програмата 4](#_Toc120821982)

[2. Реализация на условие A 4](#_Toc120821983)

[3. Реализация на условие B 4](#_Toc120821984)

[4. Реализация на условие C 5](#_Toc120821985)

[5. Реализация на условие D 5](#_Toc120821986)

[6. Реализация на условие E 6](#_Toc120821987)

[7. Реализация на условие F 7](#_Toc120821988)

[8. Реализация на условие … - допълнение първо 7](#_Toc120821989)

[9. Реализация на условие … - допълнение второ 8](#_Toc120821990)

[10. Реализация на допълнение трето 8](#_Toc120821991)

[III. Упътване за употреба 9](#_Toc120821992)

[1. Впишете съответната част от проекта 9](#_Toc120821993)

[2. Впишете съответната част от проекта 9](#_Toc120821994)

[IV. Примерно действие на програмата 10](#_Toc120821995)

[1. Условие A 10](#_Toc120821996)

[2. Условие B 10](#_Toc120821997)

[3. Условие C 10](#_Toc120821998)

[4. Условие D 10](#_Toc120821999)

[5. Условие E 10](#_Toc120822000)

[6. Условие F 11](#_Toc120822001)

[7. Допълнение първо 11](#_Toc120822002)

[8. Допълнение второ 11](#_Toc120822003)

[9. Допълнение трето 11](#_Toc120822004)

Задание на проекта

**Продажба на компютри**

Да се напише компютърна програма, реализираща информационна система за продажба на компютърни конфигурации (сериен номер - уникален 8-цифров код, марка, модел, процесор – производител, брой ядра, RAM памет, цена, статус: в продажба/продадена). Максималният брой конфигурации, които могат да се поддържат, е 100.

**Базова задача**

1. Меню за избор на функциите от програмата

**Функции от програмата са:**

1. Добавяне на конфигурация:
   1. Добавянето трябва да позволява да се въвеждат различен брой нови конфигурации, които да се допълват в масива. Не трябва да превишава максималния брой конфигурации (100).

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

1. Извеждане на екрана
   1. Извеждане на всички конфигурации в оформен вид
2. Търсене и извеждане на екрана:
   1. Намиране и извеждане на конфигурациите с най-голяма RAM памет
   2. Търсене и извеждане на всички конфигурации от даден производител на процесор
3. Подреждане на основния масив с конфигурации, без да се извежда на екрана:

a. Подреждане на конфигурациите по нарастващ ред на тяхната цена

1. Управление на файл:
   1. Извеждане на масива с конфигурации във файл (двоичен)
   2. Въвеждане на масива с конфигурации от файл (двоичен)

**Допълнение първо (+ базова задача)**

1. Създайте подменю, в което се влиза от основното, с нови функции за:
2. Извеждане на всички непродадени конфигурации в подреден ред по сериен номер, без да се променя основния масив.
3. Търсене и извеждане на конфигурациите по въведени марка и модел.

**Допълнение второ (+ базова задача)**

1. Продажба на конфигурация
2. Търсене на подходяща конфигурация. Създайте функционалност с меню, посредством което можете да изберете по кой критерий да търсите подходяща конфигурация: марка и модел; процесор – производител; брой ядра; RAM памет или максимална цена. При зададен избор да се извеждат всички конфигурации в продажба, които съответстват на потребителския вход.
3. Продажба на конфигурацията. Извършва се по сериен номер.
   1. Ако в масива присъства конфигурация с този номер. Да се провери дали потребителя не въвежда номер на вече продадена конфигурация**:**
      1. **Конфигурацията е в продажба -** трябва да се потвърди намерението за продажбата. При потвърждение потребителят се запитва дали има код за намаление. При положителен отговор да се изисква неговото въвеждане. Ако се въведе 4-цифрено число, кратно едновременно и на 8, и на 7, да се предвиди 5% отстъпка от цената. След продажбата да се изведе съобщение **„Успешна продажба“,** да се изведат старата и новата продажна цена (без код за отстъпка те ще са с една и съща стойност), а конфигурацията да се отбележи като продадена.
      2. **Конфигурацията е продадена –** извежда се съобщение „Конфигурацията вече не е налична“.
   2. Ако серийният номер не присъства в нито една от въведените конфигурации, трябва да се изведе съобщение **„Не е открита конфигурация с посочения сериен номер“.**

**Допълнение трето (+ базова задача)**

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

Анализ на решението

Структура за данните в програмата

|  |  |
| --- | --- |
| Структура | Обяснение |
| struct ComputerConfiguration  {  Char serialnumber[8];  string brand;  string model;  string processormaker;  int corescount;  int RAM;  double price;  string status;  }; | * **serialnumber** – сериен номер(уникален за всяка конфигурация) * **brand**- марка на компютъра * **model**- модел на компютъра * **processormaker**- производител на процесора на компютъра * **corescount**-брой ядра на компютъра * **RAM**-размера на RAM в GB * **price**-продажна цена на компютъра * **status**- показва дали компютъра е в продажба(purchased) или не е(unpurchased) |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Примерни данни** | | | | | | | |
| **Сериен номер** | **Марка** | **Модел** | **Производител процесор** | **Ядра-брой** | **RAM** | **Цена** | **Статус** |
| **10000001** | **APPLE** | **STUDIOM1ULTRA64G/1TBJMW3Z** | **APPLE** | **20** | **64** | **9299** | **purchased** |
| **10000010** | **APPLE** | **PRO13,3"MNEJ3ZE/A-512GB/SPG** | **APPLE** | **8** | **8** | **3499** | **unpurchased** |
| **10000011** | **GPLAY** | **RTX3060DUAL** | **RYZEN** | **6** | **16** | **2399** | **unpurchased** |
| **10000100** | **GPLAY** | **TUFRTX2060** | **RYZEN** | **6** | **16** | **2599** | **unpurchased** |
| **10000101** | **ACER** | **PREDATORPO3-630DG.E2CEX.006** | **INTEL** | **6** | **16** | **2599** | **purchased** |
| **10000110** | **APPLE** | **IMAC24"RETINA4.5KMJVA3ZE/A** | **APPLE** | **8** | **8** | **2999** | **unpurchased** |
| **10000111** | **ASUS** | **G15DK-WB5520** | **RYZEN** | **6** | **16** | **2659** | **purchased** |
| **10001000** | **GPLAY** | **WHITEMADNESS** | **INTEL** | **6** | **16** | **3049** | **unpurchased** |
| **10001001** | **ACER** | **VERITONS2680GDT.VV2EX.004** | **INTEL** | **4** | **8** | **879** | **purchased** |
| **10001010** | **GPLAY** | **GAMEZENG** | **RYZEN** | **8** | **16** | **1149** | **unpurchased** |
| **10001011** | **ASUS** | **G10CE-2140390PF02T2-M00AN0** | **INTEL** | **6** | **16** | **1899** | **purchased** |
| **10001100** | **APPLE** | **AIR13,3"MGN93ZE/A-256GB/S** | **APPLE** | **8** | **8** | **2099** | **unpurchased** |
| **10001101** | **ASUS** | **E410MA-BV1182WS** | **INTEL** | **2** | **4** | **599** | **purchased** |
| **10001110** | **LENOVO** | **IDEAPAD315IML0581WB00HJBM** | **INTEL** | **2** | **4** | **609** | **purchased** |
| **10001111** | **ASUS** | **GX650RX-LB158X** | **RYZEN** | **8** | **32** | **8799** | **unpurchased** |
| **10010000** | **ASUS** | **G35CG-WB9931W** | **INTEL** | **8** | **32** | **7499** | **unpurchased** |

Реализация на условие A

Анализ на алгоритъма, който трябва да се реализира

1. Менюто за избор на функциите на програмата се намира в главната функция-main.
2. Реализацията и започва с въведена константа за максимален брой компютърни конфигурации, които са 100 на брой.
3. Следва задаване на две целочислени променливи.
4. Въвеждане на първата като тя е за броя на входните данни, които се ползват(15 на брой-примерни данни).При въвеждането има капсулизация, която не дава броя на данните да е по-голям от максималния брой компютърни конфигурации(100) и не по-малък от 2 на брой входни данни.
5. Следва задаване на структура-масив с максимален брой компютърни конфигурации(100)
6. Следва функцията inputData()-вкарва началните/стартовите(примерните данни) данни от файл и прави проверка дали има връзка с файла. Излизат различни съобщения при успешна и неуспешна свръзка с файла.
7. Следва отваряне и започване на цикъл do{ }while(); След стартирането му се извежда пояснителен текст за функциите на менюто и съответния номер на които отговаря за да бъде избрана
8. Въвеждане на втората променлива въведена(от т.3) която е свързана с работата на менюто и избора функция на програмата. Тя е капсулизирана като тя не може да приема стойности по-малки от 1 и по-големи от 9.
9. Начало на менюто. То има 9 опции и избора зависи от т.8:
10. AddNew();
11. ShowData();
12. SUBMENU3();
13. SortByPrice();
14. OutputDataBinary();InputDataBinary();
15. SUBMENU6();
16. SUBMENU7();
17. SUBMENU8();
18. Излизане от менюто и край на програмата
19. Като по подразбиране извежда текста „ERROR“ и излиза от програмата
20. Затваряне и край на цикъл do{ }while(); Като стойност за въртенето на цикъла е зададено да се върти докато управляващата променлива не стане 9

Блок схема на алгоритъма

Картина, която съдържа текст, екран, монитор, екранна снимка

Описанието е генерирано автоматично

Функция, с която е реализиран алгоритъма

void main()

{

const int MaxComputers = 100;

int menu,ArrayLength;

do

{

cout << "Enter the count of the Computer Configurations:";

cin >> ArrayLength;

} while (ArrayLength < 2 || ArrayLength>100);

ComputerConfiguration Store[MaxComputers];

cout << endl;

InputData(Store,ArrayLength);

do

{

cout << endl;

cout << "Menu with main functions:" << endl;

cout << "1)Add configuration" << endl;

cout << "2)Show all data" << endl;

cout << "3)Searching for data and showing it" << endl;

cout << "4)Sorting the configurations by Price" << endl;

cout << "5)File management" << endl;

cout << "6)Submenu" << endl;

cout << "7)Configuration selling" << endl;

cout << "8)Automatically fill the data from file and record it on file" << endl;

cout << "9)Exit from the program" << endl;

cout <<endl<< "Enter a number from 1 to 9: ";

do

{

cin >> menu;

} while (menu < 1 || menu>9);

cout << endl;

switch (menu)

{

case 1: AddNew(Store, ArrayLength, MaxComputers); cout << ArrayLength; break;

case 2: ShowData(Store,ArrayLength); break;

case 3: cout << "WELCOME!" << endl; SUBMENU3(Store, ArrayLength); break;

case 4: SortByPrice(Store, ArrayLength); break;

case 5: OutputDataBinary(Store, ArrayLength); InputDataBinary(Store, ArrayLength); break;

case 6: cout << "WELCOME!" << endl; SUBMENU6(Store, ArrayLength); break;

case 7: cout << "WELCOME!" << endl; SUBMENU7(Store, ArrayLength); break;

case 8: cout << "WELCOME!" << endl; SUBMENU8(Store, ArrayLength); break;

case 9: exit(1); break;

default: cout << "ERROR!"; system("PAUSE"); exit(1);

}

} while (menu != 9);

}

#### Входни данни на функцията

|  |
| --- |
| Вход: |
| MaxComputers = 100;(по условие)  ArrayLength =15 (броя на примерните данни)  menu(1-9 избира се от клавиатурата) |

#### Изходни данни на функцията или данни, които се извеждат

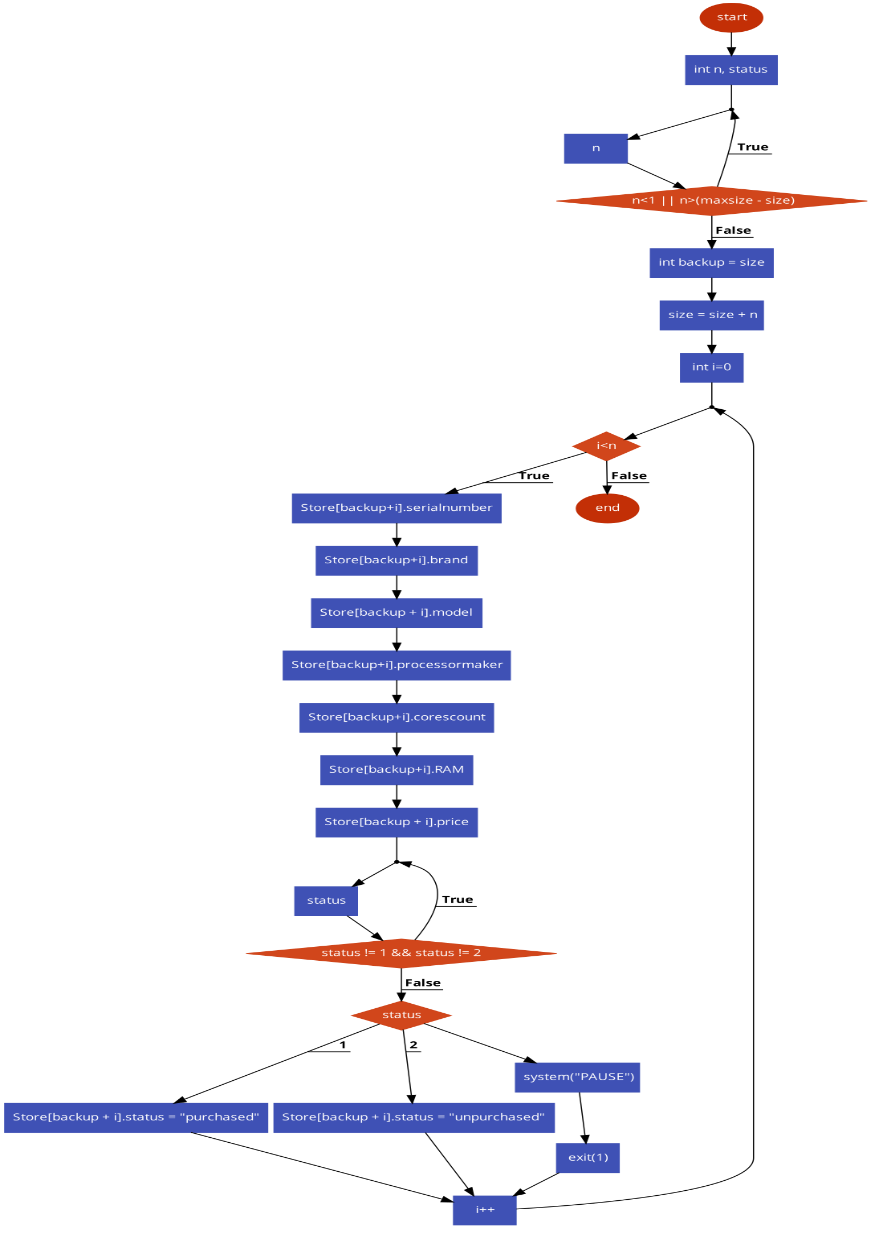
|  |  |
| --- | --- |
| (Изхода се базира на избора на стойност на menu.Крайния резултат е резултата от функцията която е избрана)  Изход: | |
| 1)AddNew();  2)ShowData();  3)SUBMENU3();  4)SortByPrice();  5)OutputDataBinary();InputDataBinary();  6)SUBMENU6();  7)SUBMENU7();  8)SUBMENU8();  9)Излизане от менюто и край на програмата |

Реализация на условие B

Анализ на алгоритъма, който трябва да се реализира

1. Добавянето трябва да позволява да се въвеждат различен брой нови конфигурации, които да се допълват в масива. Не трябва да превишава максималния брой конфигурации (100).
2. Въвежда се цяло число n, което позволява да се въведат n на брой конфигурации. n не може да надвишава свободните елементи в масива с конфигурации.
3. Началния брой на масива търпи промяна: стария брой + добавените нов = новия брой на масива

Блок схема на алгоритъма



Функция, с която е реализиран алгоритъма

void AddNew(struct ComputerConfiguration Store[],int &size,const unsigned int maxsize)//1

{

int n, status;

do { cout << "Count of NEW Configurations:"; cin >> n; } while (n<1 || n>(maxsize - size));

int backup = size;

size = size + n;

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

{

cout << "Enter Serial Number !(8 digits ONLY)!:"; cin >> Store[backup+i].serialnumber;

cout << "Enter Brand:"; getline(cin.ignore(1000,'\n'), Store[backup+i].brand);

cout << "Enter Model:"; getline(cin, Store[backup + i].model);

cout << "Enter Maker of the Processor:"; getline(cin, Store[backup+i].processormaker);

cout << "Enter Count of Cores:"; cin>>Store[backup+i].corescount;

cout << "Enter RAM:"; cin >> Store[backup+i].RAM;

cout << "Enter Price:"; cin >> Store[backup + i].price;

cout << "Select Status:"<<endl;

cout << "1) purchased" << endl;

cout << "2) unpurchased!" << endl;

do { cout << "Select 1 or 2 to continue:"; cin >> status; } while (status != 1 && status != 2);

switch (status)

{

case 1: Store[backup + i].status = "purchased"; break;

case 2: Store[backup + i].status = "unpurchased"; break;

default: cout << "ERROR!"; system("PAUSE"); exit(1);

}

}

cout << endl;

}

#### Входни данни на функцията

|  |
| --- |
| Вход: |
| maxsize=MaxComputers = 100;(по условие)  size=ArrayLength =15 (броя на примерните данни)  n=2  backup=size=15  size=size+n=15+2  for(int i=0; i<n; i++)  **Store[size+i].serialnumber** – от клавиатурата  **Store[size+i].brand**- от клавиатурата  **Store[size+i].model**- от клавиатурата  **Store[size+i].processormaker**- от клавиатурата  **Store[size+i].corescount**- от клавиатурата  **Store[size+i].RAM**- от клавиатурата  **Store[size+i].price**- от клавиатурата  **Store[size+i].status**- purchased или unpurchased |

#### Изходни данни на функцията или данни, които се извеждат

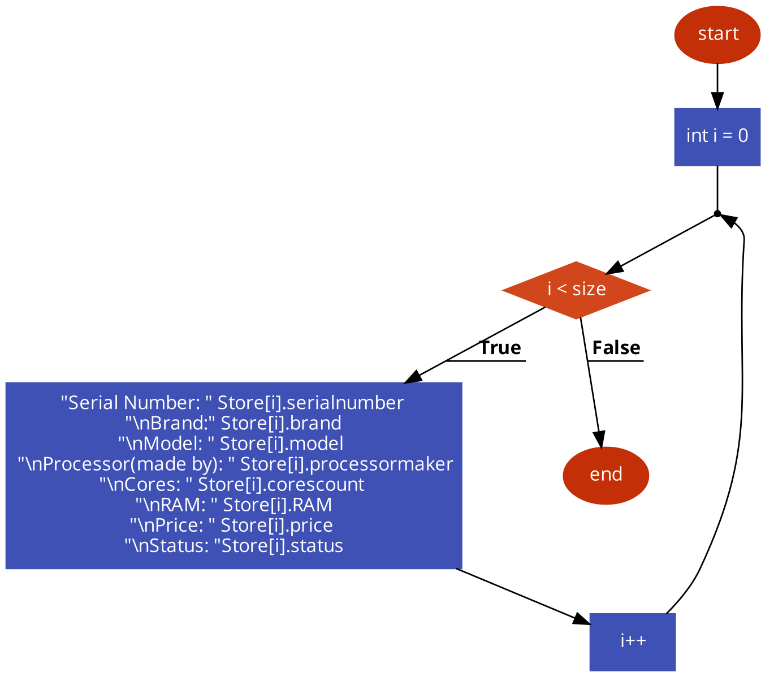
|  |
| --- |
| Изход: |
| ArrayLength=17 |

Реализация на условие C

Анализ на алгоритъма, който трябва да се реализира

1. Въвежда се масива и размера му.
2. След това се отваря цикъл for() който се върти докато не се изведат всички елементи на цикъла.
3. Форматира се начина на извеждане

Блок схема на алгоритъма



Функция, с която е реализиран алгоритъма

void ShowData(struct ComputerConfiguration Store[],const unsigned int size)//2

{

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

{

cout << "Serial Number: " << Store[i].serialnumber << "\nBrand: " << Store[i].brand << "\nModel: " << Store[i].model << "\nProcessor(made by): " << Store[i].processormaker << "\nCores: " << Store[i].corescount << "\nRAM: " << Store[i].RAM << " GB \nPrice: " << Store[i].price << "LV. \nStatus: " << Store[i].status << endl << endl;

}

cout << endl;

}

#### Входни данни на функцията

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Вход:** | | | | | | | |
| **Сериен номер** | **марка** | **модел** | **произв.-процесор** | **Ядра-брой** | **RAM** | **цена** | **Статус** |
| **10000001** | **APPLE** | **STUDIOM1ULTRA64G/1TBJMW3Z** | **APPLE** | **20** | **64** | **9299** | **purchased** |
| **10000010** | **APPLE** | **PRO13,3"MNEJ3ZE/A-512GB/SPG** | **APPLE** | **8** | **8** | **3499** | **unpurchased** |
| **10000011** | **GPLAY** | **RTX3060DUAL** | **RYZEN** | **6** | **16** | **2399** | **unpurchased** |
| **10000100** | **GPLAY** | **TUFRTX2060** | **RYZEN** | **6** | **16** | **2599** | **unpurchased** |
| **10000101** | **ACER** | **PREDATORPO3-630DG.E2CEX.006** | **INTEL** | **6** | **16** | **2599** | **purchased** |
| **10000110** | **APPLE** | **IMAC24"RETINA4.5KMJVA3ZE/A** | **APPLE** | **8** | **8** | **2999** | **unpurchased** |
| **10000111** | **ASUS** | **G15DK-WB5520** | **RYZEN** | **6** | **16** | **2659** | **purchased** |
| **10001000** | **GPLAY** | **WHITEMADNESS** | **INTEL** | **6** | **16** | **3049** | **unpurchased** |
| **10001001** | **ACER** | **VERITONS2680GDT.VV2EX.004** | **INTEL** | **4** | **8** | **879** | **purchased** |
| **10001010** | **GPLAY** | **GAMEZENG** | **RYZEN** | **8** | **16** | **1149** | **unpurchased** |
| **10001011** | **ASUS** | **G10CE-2140390PF02T2-M00AN0** | **INTEL** | **6** | **16** | **1899** | **purchased** |
| **10001100** | **APPLE** | **AIR13,3"MGN93ZE/A-256GB/S** | **APPLE** | **8** | **8** | **2099** | **unpurchased** |
| **10001101** | **ASUS** | **E410MA-BV1182WS** | **INTEL** | **2** | **4** | **599** | **purchased** |
| **10001110** | **LENOVO** | **IDEAPAD315IML0581WB00HJBM** | **INTEL** | **2** | **4** | **609** | **purchased** |
| **10001111** | **ASUS** | **GX650RX-LB158X** | **RYZEN** | **8** | **32** | **8799** | **unpurchased** |

#### Изходни данни на функцията или данни, които се извеждат

|  |
| --- |
| Изход: |
| (15 пъти се показват)  Serial Number: …  Brand: …  Model: …  Processor(made by): …  Cores: …  RAM: …  Price: … Status: ...  Пример:  Serial Number: 10001111  Brand:ASUS  Model: GX650RX-LB158X  Processor(made by): RYZEN  Cores: 8  RAM: 32 GB  Price: 8799LV.  Status: unpurchased |

Реализация на условие D

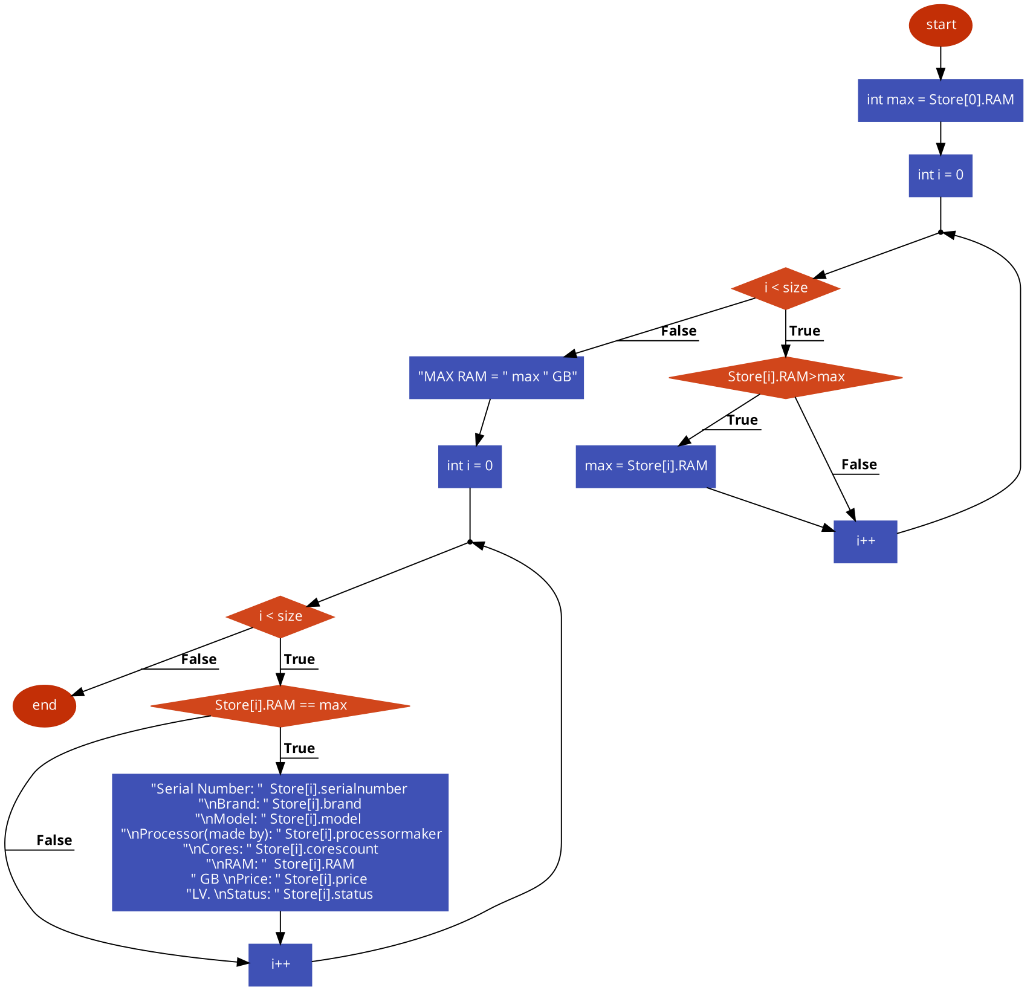
Анализ на алгоритъма, който трябва да се реализира

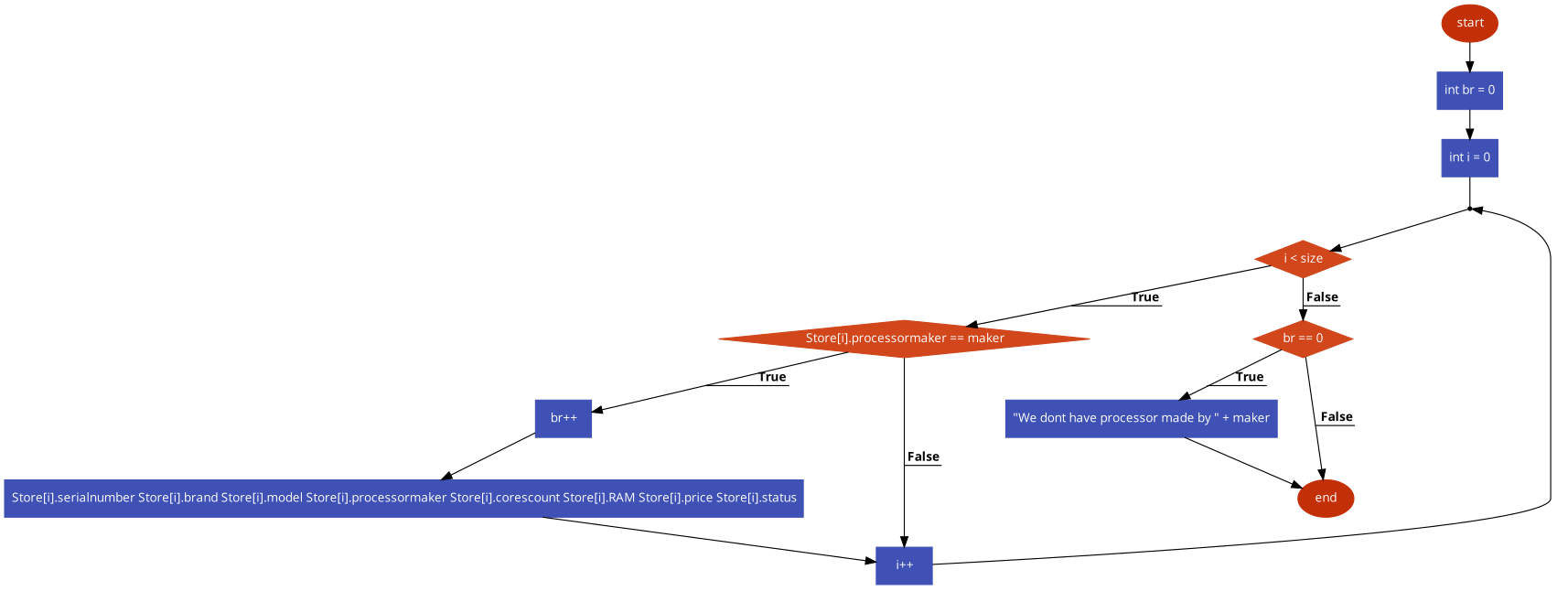
1. При menu=3 се отваря подменю 3, което дава да се изберат 3 стойности за подменюто А, В и С(контролираща променлива на подменю 3-submenu3)
2. Самото меню е в цикъл do{}while()
3. А)🡪Отваря функцията MaxRAM(), която търси компютрите с най-голяма RAM памет
4. Въвежда се променлива max която стойност е първия елемент на масива с индекс 0
5. С помощта на цикъла for обхождаме целия масив и правим проверка дали има елемент с по-голяма стойност от max, ако има то max приема неговата стойност
6. С помощта на втори цикъл for намираме всички елементи на масива с Ram равна на max
7. Следва извеждане на всички тези елементи
8. B)🡪Отваря функцията ProcessorMakerSearch(), която търси компютри с конкретен производител на процесора
9. При въвеждането на функцията тя изисква параметър за производител на процесора
10. Въвежда се променлива br която стойност 0(брояч)
11. С помощта на цикъла for обхождаме целия масив и правим проверка дали има елемент с производител на процесора съвпадащ с търсения, ако има брояча се увеличава с единица всеки път щом условието е изпълнено и извежда конкретния елемент(компютърна конфигурация)
12. Прави се проверка на брояча, ако не се е променил и си стои със стойност 0 то се извежда съобщение, че няма компютър с процесор произведен от търсения производител
13. C)🡪Излиза от подменю 3 и затваря цикъла и се връща в главното меню

Блок схема на алгоритъма

Картина, която съдържа текст, монитор, закрито, екранна снимка

Описанието е генерирано автоматично





Функция с която е реализиран алгоритъма

void MaxRAM(struct ComputerConfiguration Store[], const unsigned int size)//3a

{

int max = Store[0].RAM;

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

{

if (Store[i].RAM>max) max = Store[i].RAM;

}

cout << endl << "MAX RAM = " << max << " GB" << endl;

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

{

if (Store[i].RAM == max)

{

cout << "Serial Number: " << Store[i].serialnumber << "\nBrand:" << Store[i].brand

<< "\nModel: " << Store[i].model << "\nProcessor(made by): " << Store[i].processormaker

<< "\nCores: " << Store[i].corescount << "\nRAM: " << Store[i].RAM << " GB \nPrice: "

<< Store[i].price << "LV. \nStatus: " << Store[i].status << endl << endl;

}

}

}

void ProcessorMakerSearch(struct ComputerConfiguration Store[], const unsigned int size,string maker)//3b

{

int br = 0;

cout << endl << "Processor Maker: " << maker << endl<<endl;

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

{

if (Store[i].processormaker == maker)

{

br++;

cout << "Serial Number: " << Store[i].serialnumber << "\nBrand:" << Store[i].brand

<< "\nModel: " << Store[i].model << "\nProcessor(made by): " << Store[i].processormaker

<< "\nCores: " << Store[i].corescount << "\nRAM: " << Store[i].RAM << " GB \nPrice: "

<< Store[i].price << "LV. \nStatus: " << Store[i].status << endl << endl;

}

}

if (br == 0) cout << "We dont have processor made by " + maker<<endl<<endl;

}

void SUBMENU3(struct ComputerConfiguration Store[], const unsigned int size)//3

{

char submenu3;

string maker;

do

{

cout << endl;

cout << "Submenu 3 with functions:" << endl;

cout << "3.A) Max RAM configurations" << endl;

cout << "3.B) Search by maker of the processor" << endl;

cout << "3.C) Exit from Submenu 3" << endl;

do

{

cout << endl << "Choose A, B or C: "; cin >> submenu3;

} while (submenu3 != 'A' && submenu3 != 'B' && submenu3 != 'C');

switch (submenu3)

{

case 'A': MaxRAM(Store,size); break;

case 'B': cout << "Search by Processor Maker:";

getline(cin.ignore(1000,'\n'), maker); ProcessorMakerSearch(Store,size,maker); break;

case 'C': break;

default: cout << "ERROR!"; system("PAUSE"); exit(1);

}

} while (submenu3 != 'C');

}

#### Входни данни на функцията

|  |
| --- |
| 3.Вход: |
| Submenu3=приема стойност от клавиатурата:  А) MaxRAM();  В) maker=приема стойност за клавиатурата; ProcessorMakerSearch;  С) Край на цикъла, излизане от подменю 3 и връщане в главното |

#### Изходни данни на функцията или данни, които се извеждат

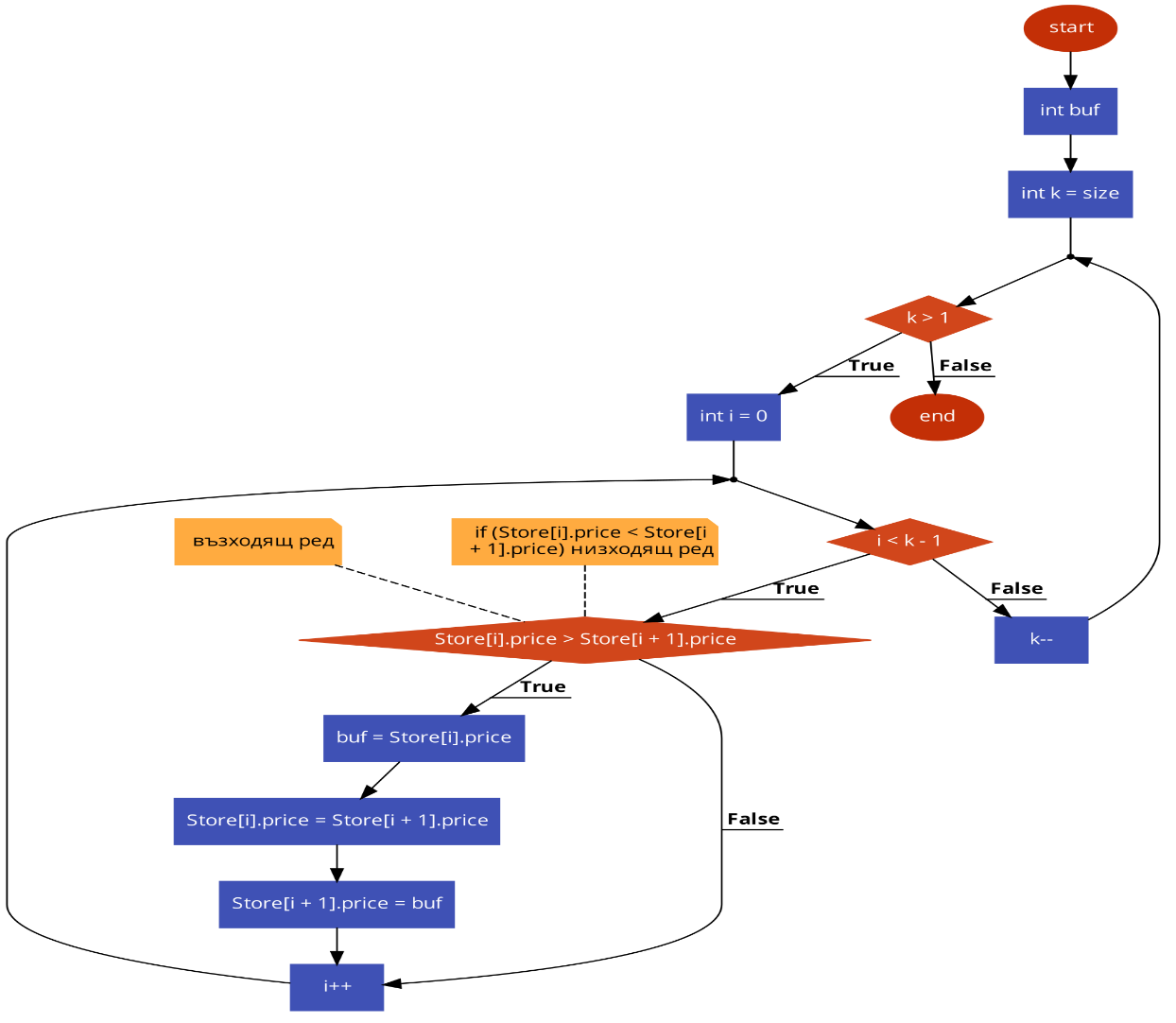
|  |
| --- |
| Изход: |
| А) Извежда всички компютри с макс RAM(64GB-)  В)Извежда всички компютри с търсения производител на процесора, ако няма връща съобщение с подходящ отговор |

Реализация на условие E

Анализ на алгоритъма, който трябва да се реализира

1. Подреждане на конфигурациите по нарастващ ред на тяхната цена
2. Задаване на параметър buf
3. Отваряне на първия for
4. Отваряне на втория for
5. Започва проверка на съседни елементи относно това кои е с по-голяма стойност
6. Buf приема стойността на елемента с по-голяма стойност
7. Елемента с по-голяма стойност приемна стойността на елемента с по-малка стойност
8. Елемента с по-малка стойност приема стойността на buf
9. След края на цикъла елементите са сортирани по възходящ ред

Блок схема на алгоритъма



Функция с която е реализиран алгоритъма

void SortByPrice(struct ComputerConfiguration Store[], const unsigned int size)//4

{

ComputerConfiguration buf;

for (int k = size; k > 1; k--)

{

for (int i = 0; i < k - 1; i++)

{

//if (Store[i].price < Store[i + 1].price) //низходящ ред

if (Store[i].price > Store[i + 1].price) //възходящ ред

{

buf = Store[i];

Store[i] = Store[i + 1];

Store[i + 1] = buf;

}

}

}

}

#### Входни данни на функцията



#### Изходни данни на функцията или данни, които се извеждат

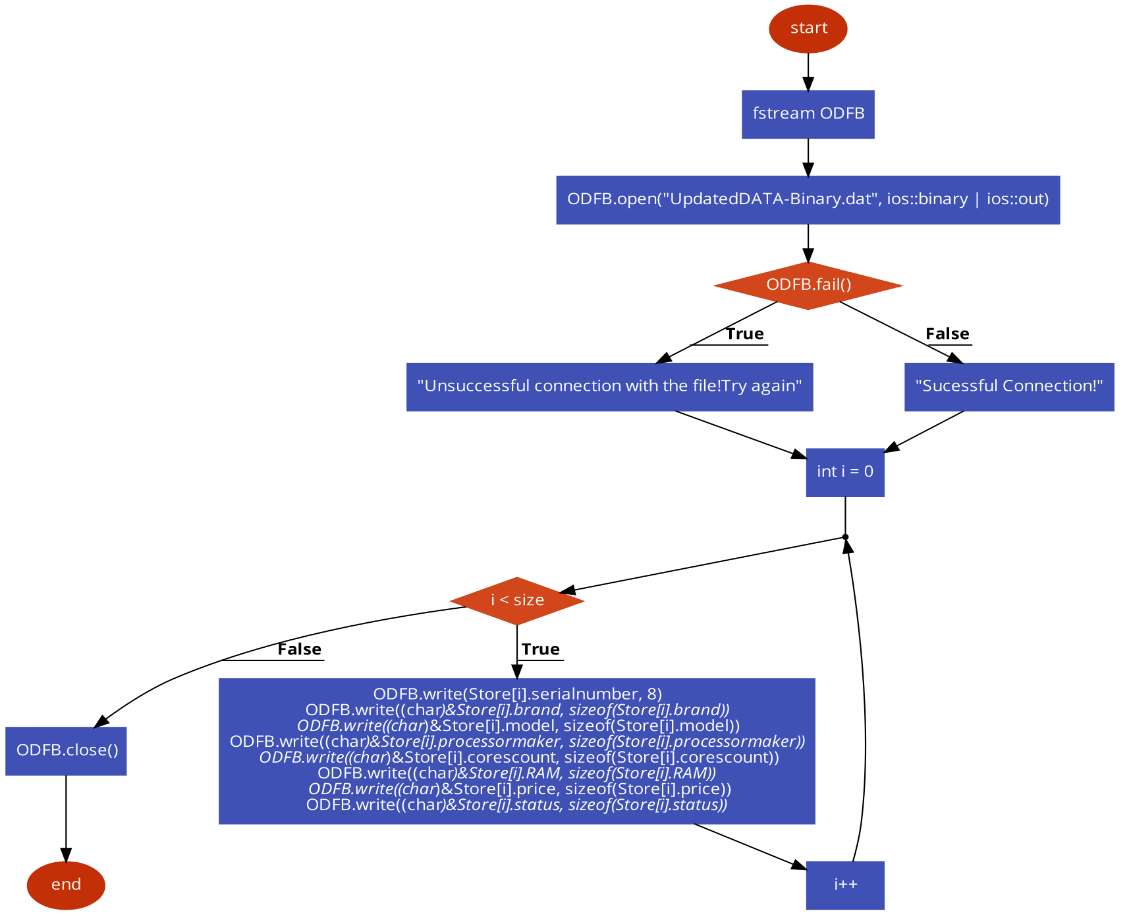


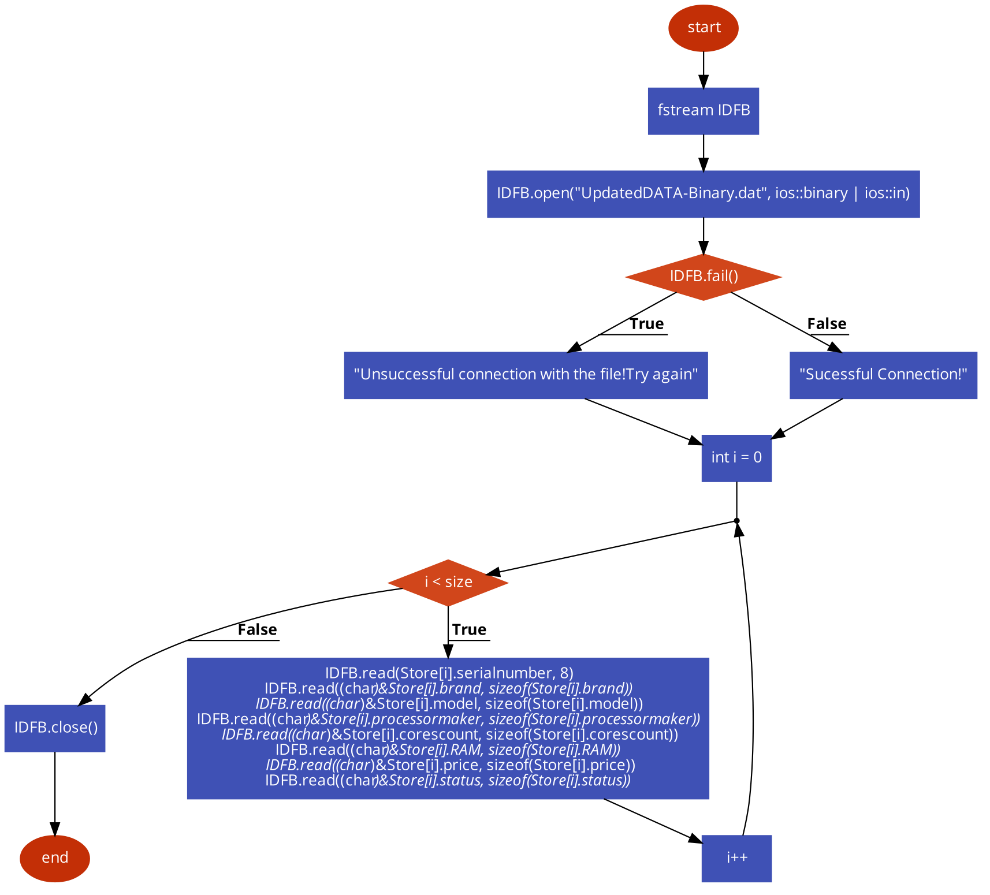
Реализация на условие F

Анализ на алгоритъма, който трябва да се реализира

1. Създаване на входно-изходна файлова променлива и в двете функции за четене и записване
2. Отваря се файл със статус ios::binary | ios::in/ios::out
3. Прави се проверка дали има връзка или не
4. Отваря се цикъл for за обхождане на масива
5. Записване/Четене на данните от масива и преобразуването им
6. Затваряне на файла

Блок схема на алгоритъма





Функция с която е реализиран алгоритъма

void OutputDataBinary(struct ComputerConfiguration Store[], const unsigned int size)//5

{

fstream ODFB;

ODFB.open("UpdatedDATA-Binary.dat", ios::binary | ios::out);

if (ODFB.fail()) cout << "Unsuccessful connection with the file!Try again" << endl;

else cout << "Sucessful Connection!" << endl;

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

{

ODFB.write(Store[i].serialnumber, 8);

ODFB.write((char\*)&Store[i].brand, sizeof(Store[i].brand));

ODFB.write((char\*)&Store[i].model, sizeof(Store[i].model));

ODFB.write((char\*)&Store[i].processormaker, sizeof(Store[i].processormaker));

ODFB.write((char\*)&Store[i].corescount, sizeof(Store[i].corescount));

ODFB.write((char\*)&Store[i].RAM, sizeof(Store[i].RAM));

ODFB.write((char\*)&Store[i].price, sizeof(Store[i].price));

ODFB.write((char\*)&Store[i].status, sizeof(Store[i].status));

}

ODFB.close();

}

void InputDataBinary(struct ComputerConfiguration Store[], const unsigned int size)//5

{

fstream IDFB;

IDFB.open("UpdatedDATA-Binary.dat", ios::binary | ios::in);

if (IDFB.fail()) cout << "Unsuccessful connection with the file!Try again" << endl;

else cout << "Sucessful Connection!" << endl;

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

{

IDFB.read(Store[i].serialnumber, 8);

IDFB.read((char\*)&Store[i].brand, sizeof(Store[i].brand));

IDFB.read((char\*)&Store[i].model, sizeof(Store[i].model));

IDFB.read((char\*)&Store[i].processormaker, sizeof(Store[i].processormaker));

IDFB.read((char\*)&Store[i].corescount, sizeof(Store[i].corescount));

IDFB.read((char\*)&Store[i].RAM, sizeof(Store[i].RAM));

IDFB.read((char\*)&Store[i].price, sizeof(Store[i].price));

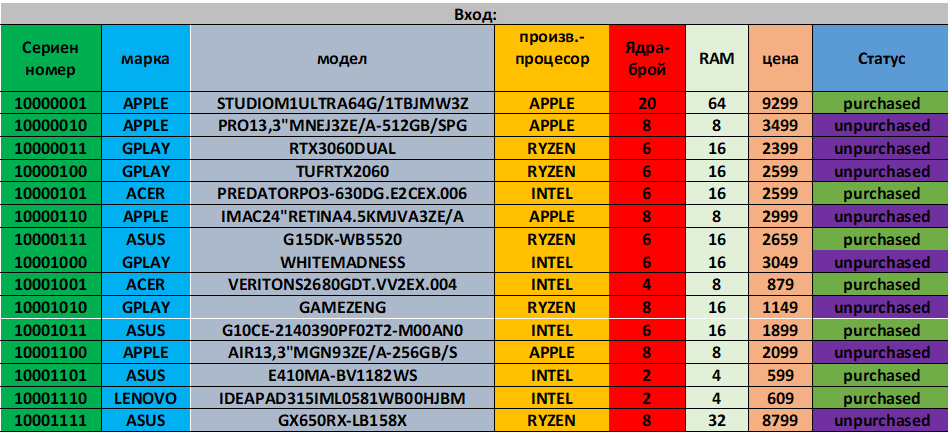
IDFB.read((char\*)&Store[i].status, sizeof(Store[i].status));

}

IDFB.close();

}

#### Входни данни на функцията



#### Изходни данни на функцията или данни, които се извеждат

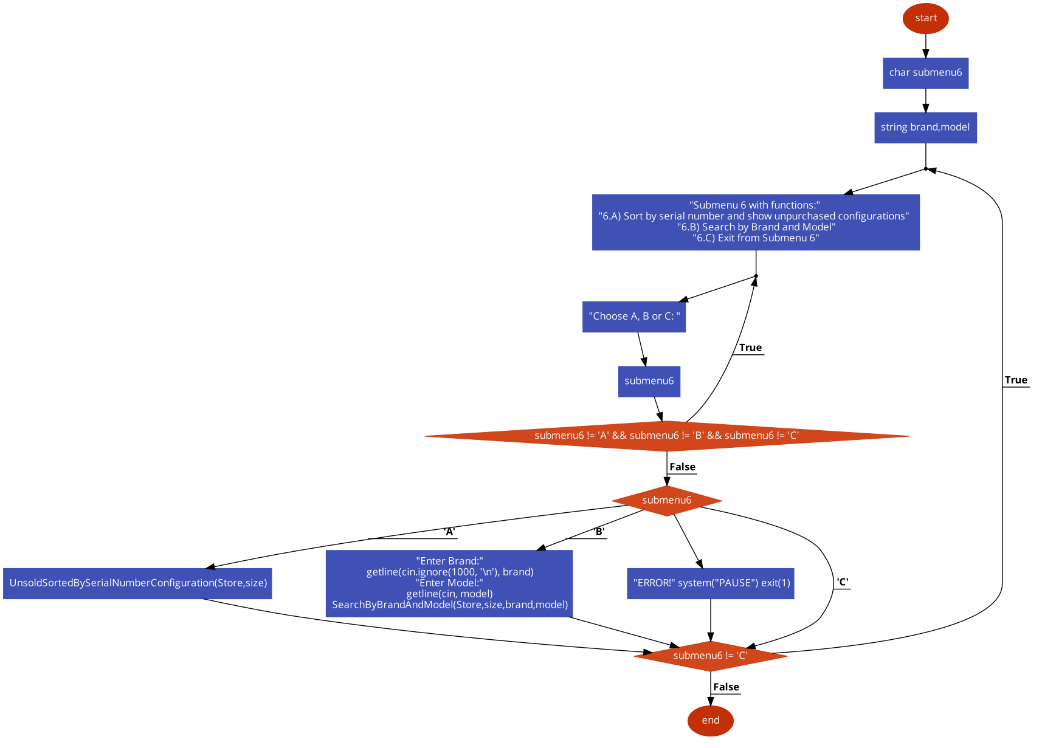
Резултата е записани данните в двоичен файл и взимане на данните от файла

Реализация на условие … - допълнение първо

Анализ на алгоритъма, който трябва да се реализира

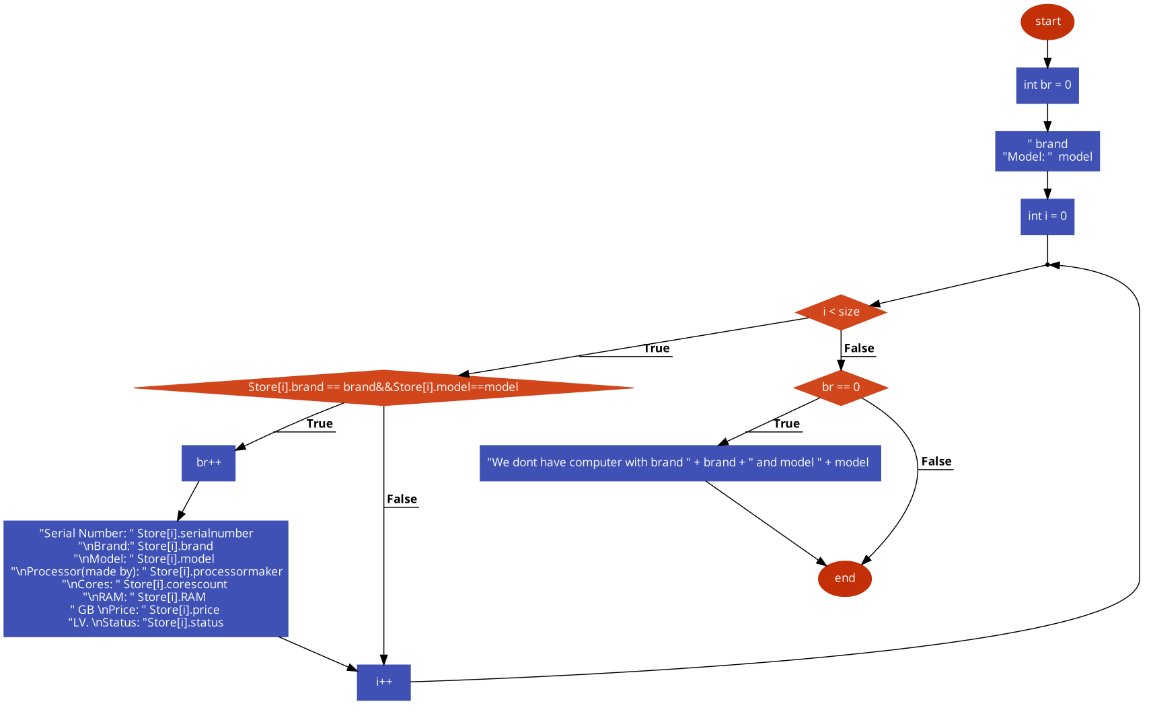
1. При menu=6 се отваря подменю 6, което дава да се изберат 3 стойности за подменюто А, В и С(контролираща променлива на подменю 6-submenu6)
2. Самото меню е в цикъл do{}while()
3. А)🡪Отваря функцията UnsoldSortedBySerialNumberConfiguration(), която сортира масива по сериен номер във възходящ ред и извежда всички непродадени компютри
4. Подреждане на конфигурациите по нарастващ ред на тяхната цена
5. Задаване на параметър buf(като структура)
6. Отваряне на първия for
7. Отваряне на втория for
8. Започва проверка на съседни елементи относно това кои е с по-голяма стойност
9. Buf приема стойността на елемента с по-голяма стойност
10. Елемента с по-голяма стойност приемна стойността на елемента с по-малка стойност
11. Елемента с по-малка стойност приема стойността на buf
12. След края на цикъла елементите са сортирани по възходящ ред
13. Отваря се нов цикъл за обхождане на масива
14. Прави се проверка по статус ако са в продажба се извеждат
15. B)🡪 brand-въвежда се от клавиатурата; model-въвежда се от клавиатурата; SearchByBrandAndModel();, която търси компютри по конкретно въведени марка и модел
16. При въвеждането на функцията тя изисква параметри за модел и марка
17. Въвежда се променлива br която стойност 0(брояч)
18. С помощта на цикъла for обхождаме целия масив и правим проверка дали има елемент с модел и марка съвпадащи с търсените и дали е в продажба, ако има брояча се увеличава с единица всеки път щом условието е изпълнено и извежда конкретния елемент(компютърна конфигурация)
19. Прави се проверка на брояча, ако не се е променил и си стои със стойност 0 то се извежда съобщение, че няма компютър с търсените модел и марка или е бил купен
20. C)🡪Излиза от подменю 6 и затваря цикъла и се връща в главното меню

Блок схема на алгоритъма



Картина, която съдържа текст, екран, екранна снимка

Описанието е генерирано автоматично



Функция с която е реализиран алгоритъма

void UnsoldSortedBySerialNumberConfiguration(struct ComputerConfiguration Store[], const unsigned int size)//6a

{

cout << endl;

cout << "Unpurchased computers sorted by serial number" << endl;

ComputerConfiguration buf;

for (int k = size; k > 1; k--)

{

for (int i = 0; i < k - 1; i++)

{

//if (strcmp(Store[i].serialnumber,Store[i+1].serialnumber)<0) //низходящ ред

if (strcmp(Store[i].serialnumber,Store[i+1].serialnumber)>0) //възходящ ред

{

buf = Store[i];

Store[i] = Store[i + 1];

Store[i + 1] = buf;

}

}

}

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

{

if (Store[i].status == "unpurchased")

{

cout << "Serial Number: " << Store[i].serialnumber << "\nBrand:" << Store[i].brand

<< "\nModel: " << Store[i].model << "\nProcessor(made by): " << Store[i].processormaker

<< "\nCores: " << Store[i].corescount << "\nRAM: " << Store[i].RAM << " GB \nPrice: "

<< Store[i].price << "LV. \nStatus: " << Store[i].status << endl << endl;

}

}

}

void SearchByBrandAndModel(struct ComputerConfiguration Store[], const unsigned int size,string brand,string model)//6b

{

int br = 0;

cout << "Brand: " << brand;

cout << endl << "Model: " << model << endl << endl;

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

{

if (Store[i].brand == brand&&Store[i].model==model)

{

br++;

cout << "Serial Number: " << Store[i].serialnumber << "\nBrand:" << Store[i].brand

<< "\nModel: " << Store[i].model << "\nProcessor(made by): " << Store[i].processormaker

<< "\nCores: " << Store[i].corescount << "\nRAM: " << Store[i].RAM << " GB \nPrice: "

<< Store[i].price << "LV. \nStatus: " << Store[i].status << endl << endl;

}

}

if (br == 0) cout << "We dont have computer with brand " + brand + " and model " + model << endl;

}

void SUBMENU6(struct ComputerConfiguration Store[], const unsigned int size)//6

{

char submenu6;

string brand,model;

do

{

cout << endl;

cout << "Submenu 6 with functions:" << endl;

cout << "6.A) Sort by serial number and show unpurchased configurations" << endl;

cout << "6.B) Search by Brand and Model" << endl;

cout << "6.C) Exit from Submenu 6" << endl;

do

{

cout << endl << "Choose A, B or C: "; cin >> submenu6;

} while (submenu6 != 'A' && submenu6 != 'B' && submenu6 != 'C');

switch (submenu6)

{

case 'A': UnsoldSortedBySerialNumberConfiguration(Store,size); break;

case 'B': cout << "Enter Brand:";

getline(cin.ignore(1000, '\n'), brand);

cout << "Enter Model:";

getline(cin, model);

SearchByBrandAndModel(Store,size,brand,model); break;

case 'C': break;

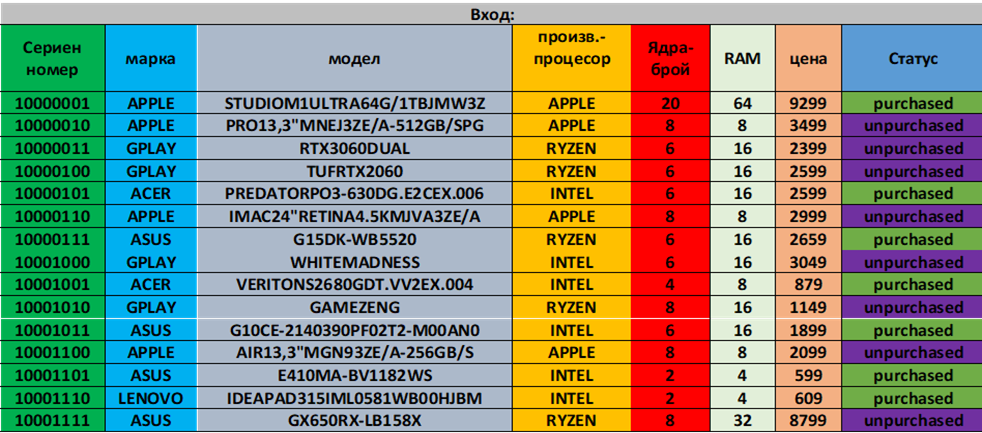
default: cout << "ERROR!"; system("PAUSE"); exit(1);

}

} while (submenu6 != 'C');

}

#### Входни данни на функцията



|  |
| --- |
| 3.Вход: |
| Submenu6=приема стойност от клавиатурата:  А) UnsoldSortedBySerialNumberConfiguration();  В) brand=APPLE; model=PRO13,3"MNEJ3ZE/A-512GB/SPG; SearchByBrandAndModel();  С) Край на цикъла, излизане от подменю 3 и връщане в главното |

#### Изходни данни на функцията или данни, които се извеждат

|  |
| --- |
| Изход: |
| А)   |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | | 10000010 | APPLE | PRO13,3"MNEJ3ZE/A-512GB/SPG | APPLE | 8 | 8 | 3499 | unpurchased | | 10000011 | GPLAY | RTX3060DUAL | RYZEN | 6 | 16 | 2399 | unpurchased | | 10000100 | GPLAY | TUFRTX2060 | RYZEN | 6 | 16 | 2599 | unpurchased | | 10000110 | APPLE | IMAC24"RETINA4.5KMJVA3ZE/A | APPLE | 8 | 8 | 2999 | unpurchased | | 10001000 | GPLAY | WHITEMADNESS | INTEL | 6 | 16 | 3049 | unpurchased | | 10001010 | GPLAY | GAMEZENG | RYZEN | 8 | 16 | 1149 | unpurchased | | 10001100 | APPLE | AIR13,3"MGN93ZE/A-256GB/S | APPLE | 8 | 8 | 2099 | unpurchased | | 10001111 | ASUS | GX650RX-LB158X | RYZEN | 8 | 32 | 8799 | unpurchased |   В)  Serial Number: 10010010  Brand:APPLE  Model: PRO13,3"MNEJ3ZE/A-512GB/SPG  Processor(made by): APPLE  Cores: 8  RAM: 8 GB  Price: 3499LV.  Status: unpurchased |

Реализация на условие … - допълнение второ

Анализ на алгоритъма, който трябва да се реализира

1. При menu=7 се отваря подменю 7, което дава да се изберат 3 стойности за подменюто А, В и С(контролираща променлива на подменю 7-submenu7)
2. Самото меню е в цикъл do{}while()
3. А)🡪 SearchingForTheRightConfigurationMenu();-Търсене на подходяща конфигурация. Меню,посредством което можете да изберете по кой критерий да търсите подходяща конфигурация: марка и модел; процесор – производител; брой ядра; RAM памет или максимална цена. При зададен избор да се извеждат всички конфигурации в продажба, които съответстват на потребителския вход.
4. Отваря подподменю 7.1
5. Задаване на параметри subsubmenu7(контролираща меню 7.1), brand, model,maker, cores, RAM;
6. Самото меню е в цикъл do{}while()
7. brand-въвежда се от клавиатурата; model-въвежда се от клавиатурата; SearchByBrandAndModel2();, която търси компютри по конкретно въведени марка и модел, които са в продажба
8. Отваря функцията ProcessorMakerSearch2(), която търси компютри с конкретен производител на процесора в продажба
9. Въвеждане на стойност на cores; SearchByCoresCount(); търси компютри с въведения брой ядра(в продажба)
10. Въвеждане на стойност на RAM; SearchByRAM(); търси компютри с избраната RAM(в продажба )
11. MaxPrice();-Извежда най-скъпия компютър в продажба
12. Отваряне на втория for Излиза от подменю 7.1 и затваря цикъла и се връща в подменю 7
13. B)🡪 Selling();

Продажба на конфигурацията. Извършва се по сериен номер.

1. Ако в масива присъства конфигурация с този номер. Да се провери дали потребителя не въвежда номер на вече продадена конфигурация:
2. Конфигурацията е в продажба - трябва да се потвърди намерението за продажбата. При потвърждение потребителят се запитва дали има код за намаление. При положителен отговор да се изисква неговото въвеждане. Ако се въведе 4-цифрено число, кратно едновременно и на 8, и на 7, да се предвиди 5% отстъпка от цената. След продажбата да се изведе съобщение „Успешна продажба“, да се изведат старата и новата родажна цена (без код за отстъпка те ще са с една и съща стойност), а конфигурацията да се отбележи като продадена.
3. Конфигурацията е продадена – извежда се съобщение „Конфигурацията не е налична“.
4. Ако серийният номер не присъства в нито една от въведените конфигурации, трябва да се изведе съобщение „Не е открита конфигурация с посочения сериен номер“.

За реализация на функцията се използва булева променлива, низ, 3 целочислени променливи, 3 проверки с If и едно меню с подменю,което в себе си прави изчисленията за продажбата

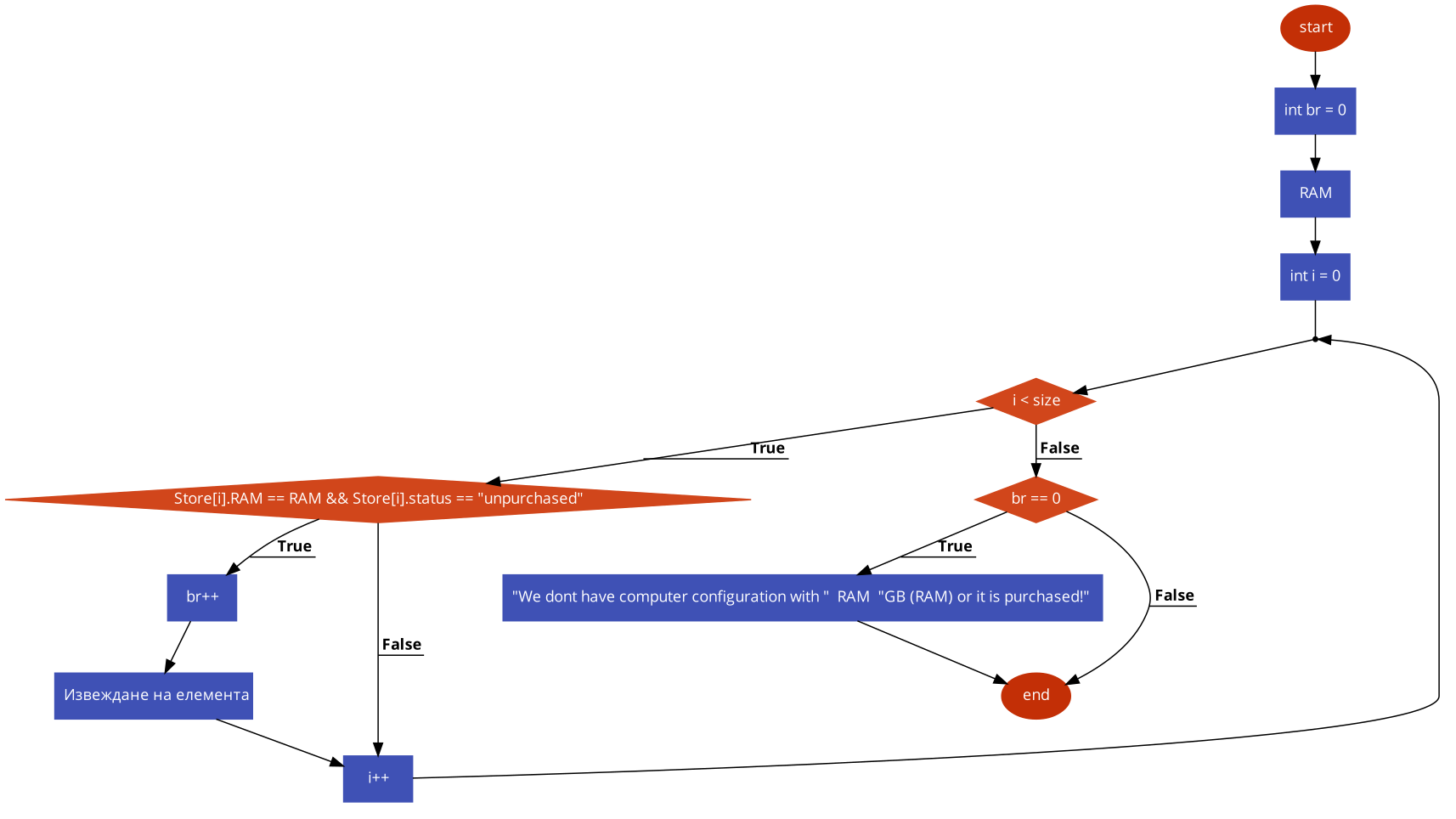
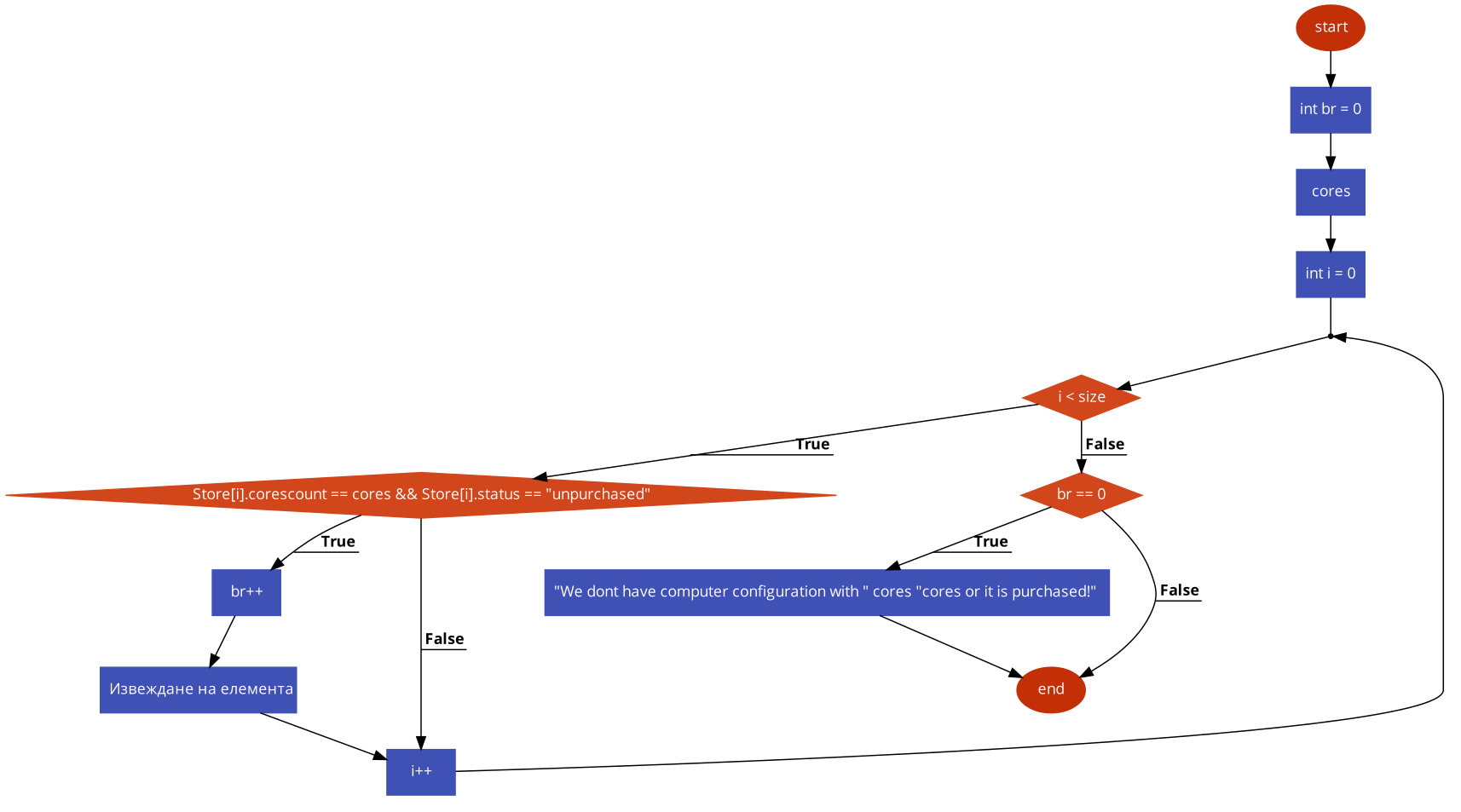
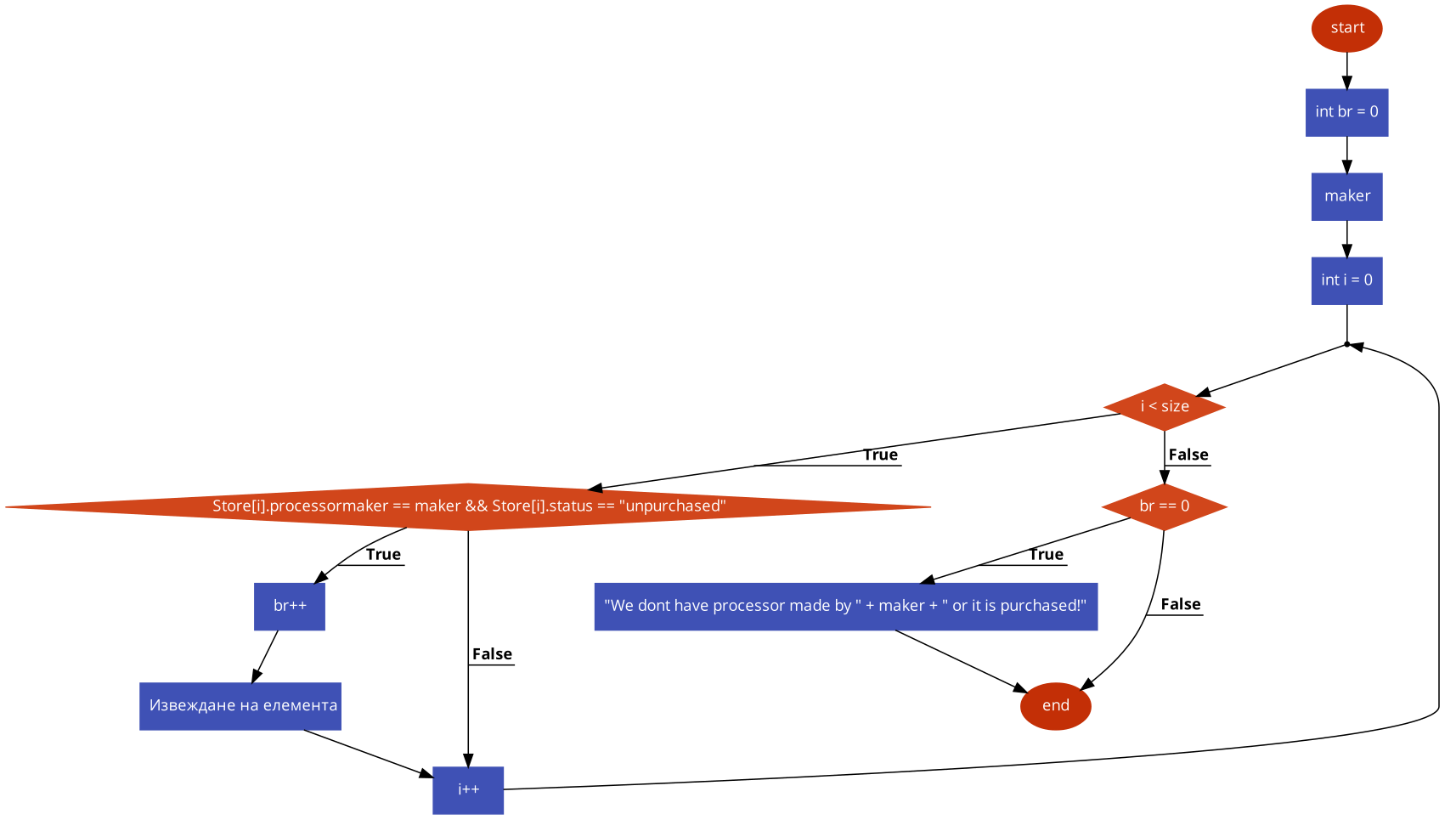
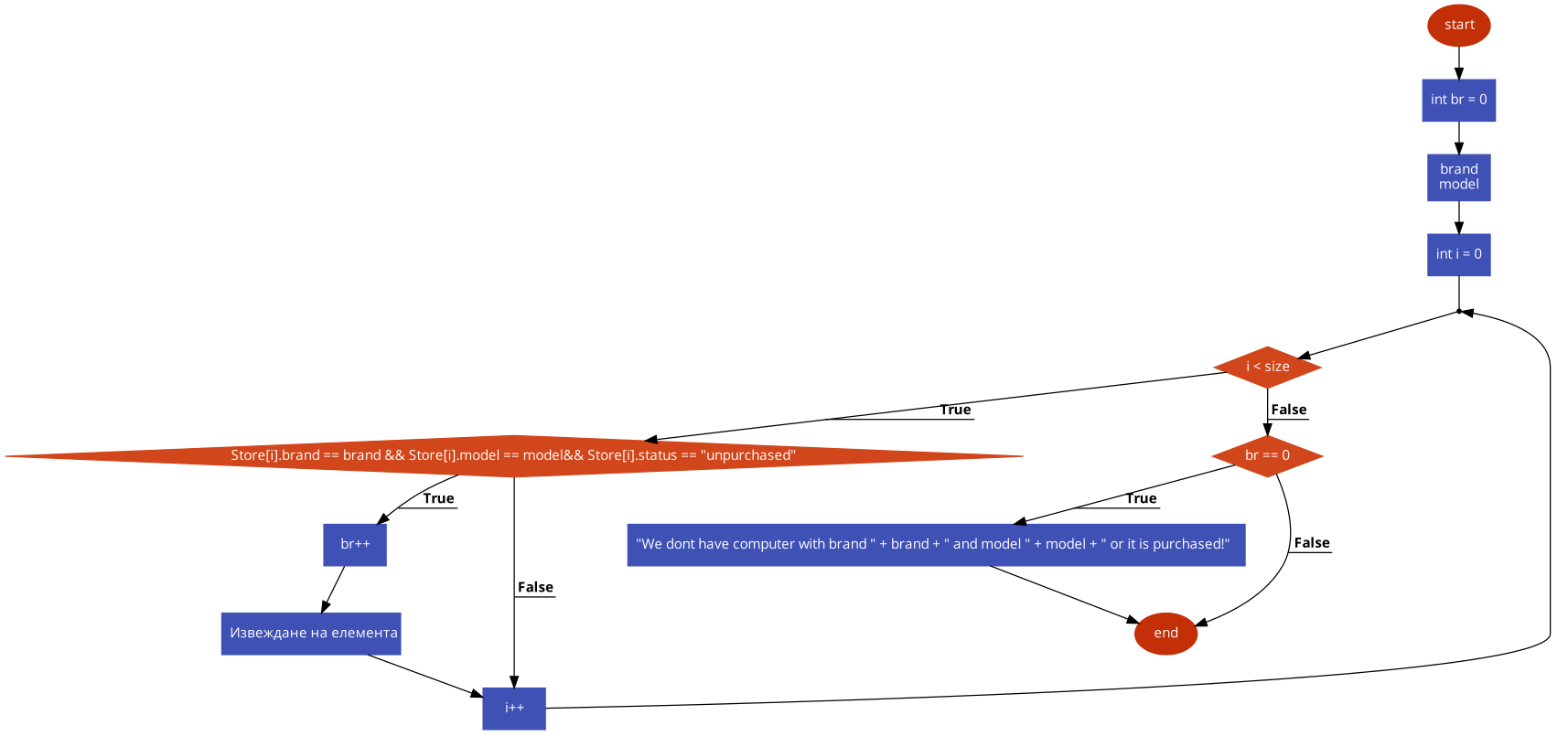
1. C)🡪Излиза от подменю 7 и затваря цикъла и се връща в главното меню

Блок схема на алгоритъма

Картина, която съдържа текст, монитор, екран, екранна снимка

Описанието е генерирано автоматичноКартина, която съдържа текст, монитор, екран, екранна снимка

Описанието е генерирано автоматично

Картина, която съдържа текст, монитор, екран, тъмен

Описанието е генерирано автоматично

Функция с която е реализиран алгоритъма

void SearchByBrandAndModel2(struct ComputerConfiguration Store[], const unsigned int size,string brand, string model)//7a

{

int br = 0;

cout << "Brand: " << brand;

cout << endl << "Model: " << model << endl << endl;

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

{

if (Store[i].brand == brand && Store[i].model == model&& Store[i].status == "unpurchased")

{

br++;

cout << "Serial Number: " << Store[i].serialnumber << "\nBrand:" << Store[i].brand

<< "\nModel: " << Store[i].model << "\nProcessor(made by): " << Store[i].processormaker

<< "\nCores: " << Store[i].corescount << "\nRAM: " << Store[i].RAM << " GB \nPrice: "

<< Store[i].price << "LV. \nStatus: " << Store[i].status << endl << endl;

}

}

if (br == 0) cout << "We dont have computer with brand " + brand + " and model " + model + " or it is purchased!" << endl;;

}

void ProcessorMakerSearch2(struct ComputerConfiguration Store[], const unsigned int size,string maker)//7a

{

int br = 0;

cout << endl << "Processor Maker: " << maker << endl << endl;

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

{

if (Store[i].processormaker == maker && Store[i].status == "unpurchased")

{

br++;

cout << "Serial Number: " << Store[i].serialnumber << "\nBrand:" << Store[i].brand

<< "\nModel: " << Store[i].model << "\nProcessor(made by): " << Store[i].processormaker

<< "\nCores: " << Store[i].corescount << "\nRAM: " << Store[i].RAM << " GB \nPrice: "

<< Store[i].price << "LV. \nStatus: " << Store[i].status << endl << endl;

}

}

if (br == 0) cout << "We dont have processor made by " + maker + " or it is purchased!" << endl << endl;

}

void SearchByCoresCount(struct ComputerConfiguration Store[], const unsigned int size,int cores)//7a

{

int br = 0;

cout << endl << "Cores: " << cores << endl << endl;

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

{

if (Store[i].corescount == cores && Store[i].status == "unpurchased")

{

br++;

cout << "Serial Number: " << Store[i].serialnumber << "\nBrand:" << Store[i].brand

<< "\nModel: " << Store[i].model << "\nProcessor(made by): " << Store[i].processormaker

<< "\nCores: " << Store[i].corescount << "\nRAM: " << Store[i].RAM << " GB \nPrice: "

<< Store[i].price << "LV. \nStatus: " << Store[i].status << endl << endl;

}

}

if (br == 0) cout << "We dont have computer configuration with " <<cores<<" cores or it is purchased!" << endl << endl;

}

void SearchByRAM(struct ComputerConfiguration Store[], const unsigned int size,int RAM)//7a

{

int br = 0;

cout << endl << "RAM: " << RAM << endl << endl;

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

{

if (Store[i].RAM == RAM && Store[i].status == "unpurchased")

{

br++;

cout << "Serial Number: " << Store[i].serialnumber << "\nBrand:" << Store[i].brand

<< "\nModel: " << Store[i].model << "\nProcessor(made by): " << Store[i].processormaker

<< "\nCores: " << Store[i].corescount << "\nRAM: " << Store[i].RAM << " GB \nPrice: "

<< Store[i].price << "LV. \nStatus: " << Store[i].status << endl << endl;

}

}

if (br == 0) cout << "We dont have computer configuration with " << RAM << "GB (RAM) or it is purchased!" << endl << endl;

}

void MaxPrice(struct ComputerConfiguration Store[], const unsigned int size)//7a

{

double max=0;

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

{

if (Store[i].status == "unpurchased") max = Store[i].price; break;

}

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

{

if (Store[i].price > max && Store[i].status=="unpurchased") max = Store[i].price;

}

cout << endl << "MAX Price = " << max << " LV" << endl;

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

{

if (Store[i].price == max && Store[i].status == "unpurchased")

{

cout << "Serial Number: " << Store[i].serialnumber << "\nBrand:" << Store[i].brand

<< "\nModel: " << Store[i].model << "\nProcessor(made by): " << Store[i].processormaker

<< "\nCores: " << Store[i].corescount << "\nRAM: " << Store[i].RAM << " GB \nPrice: "

<< Store[i].price << "LV. \nStatus: " << Store[i].status << endl;

}

}

}

void SearchingForTheRightConfigurationMenu(struct ComputerConfiguration Store[], const unsigned int size)//7a-menu

{

int submenu7;

string brand, model,maker;

int cores, RAM;

do

{

cout << endl;

cout << "Subsubmenu 7.1 with functions --> !(Unpurchased Computer Configurations ONLY)! Search By:" << endl;

cout << "7.1.(1) Brand and Model " << endl;

cout << "7.1.(2) Maker of the processor" << endl;

cout << "7.1.(3) Count of the cores " << endl;

cout << "7.1.(4) RAM" << endl;

cout << "7.1.(5) MAX Price" << endl;

cout << "7.1.(6) Exit from Submenu 7.1" << endl;

do

{

cout << endl << "Choose a number from 1 to 6: "; cin >> submenu7;

} while (submenu7 < 1 || submenu7>6);

switch (submenu7)

{

case 1: cout << "Enter Brand:";

getline(cin.ignore(1000, '\n'), brand);

cout << "Enter Model:";

getline(cin, model);

SearchByBrandAndModel2(Store,size,brand, model); break;

case 2: cout << "Search by Processor Maker:";

getline(cin.ignore(1000, '\n'), maker); ProcessorMakerSearch2(Store,size,maker); break;

case 3: cout << "Enter cores count: "; cin >> cores;

SearchByCoresCount(Store,size,cores); break;

case 4: cout << "Enter RAM: "; cin >> RAM;

SearchByRAM(Store,size,RAM); break;

case 5: MaxPrice(Store,size); break;

case 6: break;

default: cout << "ERROR!"; system("PAUSE"); exit(1);

}

} while (submenu7 != 6);

}

void Selling(struct ComputerConfiguration Store[], const unsigned int size)//7b

{

bool flag = false;

string wantednumber;

int buy,discount,discountcode;

cout << "Enter Serial Number !(8 digits ONLY)!:"; getline(cin.ignore(1000,'\n'),wantednumber);

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

{

if (Store[i].serialnumber==wantednumber)//!(strcmp(Store[i].serialnumber, wantednumber))

{

flag = true;

if (Store[i].status == "unpurchased")

{

cout << "Serial Number: " << Store[i].serialnumber << "\nBrand:" << Store[i].brand

<< "\nModel: " << Store[i].model << "\nProcessor(made by): " << Store[i].processormaker

<< "\nCores: " << Store[i].corescount << "\nRAM: " << Store[i].RAM << " GB \nPrice: "

<< Store[i].price << "LV. \nStatus: " << Store[i].status << endl;

cout << "Do you want to buy it?" << endl;

cout << "1) YES!" << endl;

cout << "2) NO!" << endl;

do { cout << "Select 1 or 2 to continue:"; cin >> buy; } while (buy != 1 && buy != 2);

switch (buy)

{

case 1:

{

cout << endl;

cout << "Do you have a discount code?" << endl;

cout << "1) YES!" << endl;

cout << "2) NO!" << endl;

do { cout << "Select 1 or 2 to continue:"; cin >> discount; } while (discount != 1 && discount != 2);

switch (discount)

{

case 1:

{

cout << "Enter discount code:"; cin >> discountcode;

if (sizeof(discountcode) == 4 && discountcode % 8 == 0 && discountcode % 7 == 0)

{

cout << "Old Price: " << Store[i].price << " LV." << endl;

cout << "New Price: " << Store[i].price - (Store[i].price \* 0.05) << " LV." << endl;

cout << ":) !SUCCESSFUL PURCHASE! :)" << endl;

Store[i].status = "purchased";

}

else

{

cout << "Price: " << Store[i].price << " LV." << endl;

cout << ":) !SUCCESSFUL PURCHASE! :)" << endl;

Store[i].status = "purchased";

}

break;

}

case 2:

{

cout << "Price: " << Store[i].price << " LV." << endl;

cout << ":) !SUCCESSFUL PURCHASE! :)" << endl;

Store[i].status = "purchased"; break;

}

default: cout << "ERROR!"; system("PAUSE"); exit(1);

}

break;

}

case 2: cout << "Thank you!Come again!" << endl; break;

default: cout << "ERROR!"; system("PAUSE"); exit(1);

}

}

else cout << " The Computer Configuration has been sold!" << endl << endl;

}

}

if (flag == false)

{

cout << "Is isn't found a Computer Configuration with the specified Serial Number!"

<< endl << endl;

}

}

void SUBMENU7(struct ComputerConfiguration Store[], const unsigned int size)//7

{

char submenu7;

do

{

cout << endl;

cout << "Submenu 7 with functions:" << endl;

cout << "7.A) Searching For The Right Configuration(Menu)" << endl;

cout << "7.B) Buying a Computer Configuration?" << endl;

cout << "7.C) Exit from Submenu 7" << endl;

do

{

cout << endl << "Choose A, B or C: "; cin >> submenu7;

} while (submenu7 != 'A' && submenu7 != 'B' && submenu7 != 'C');

switch (submenu7)

{

case 'A': SearchingForTheRightConfigurationMenu(Store,size); break;

case 'B': Selling(Store,size); break;

case 'C': break;

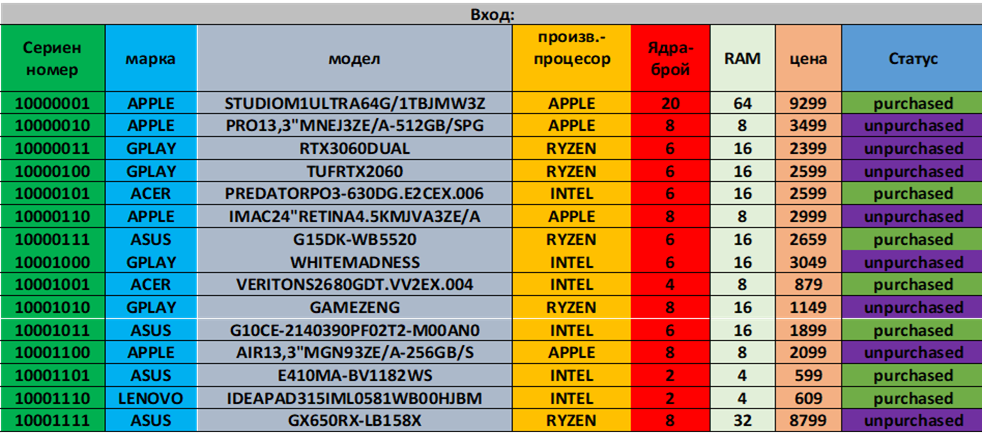
default: cout << "ERROR!"; system("PAUSE"); exit(1);

}

} while (submenu7 != 'C');

}

#### Входни данни на функцията



#### Изходни данни на функцията или данни, които се извеждат



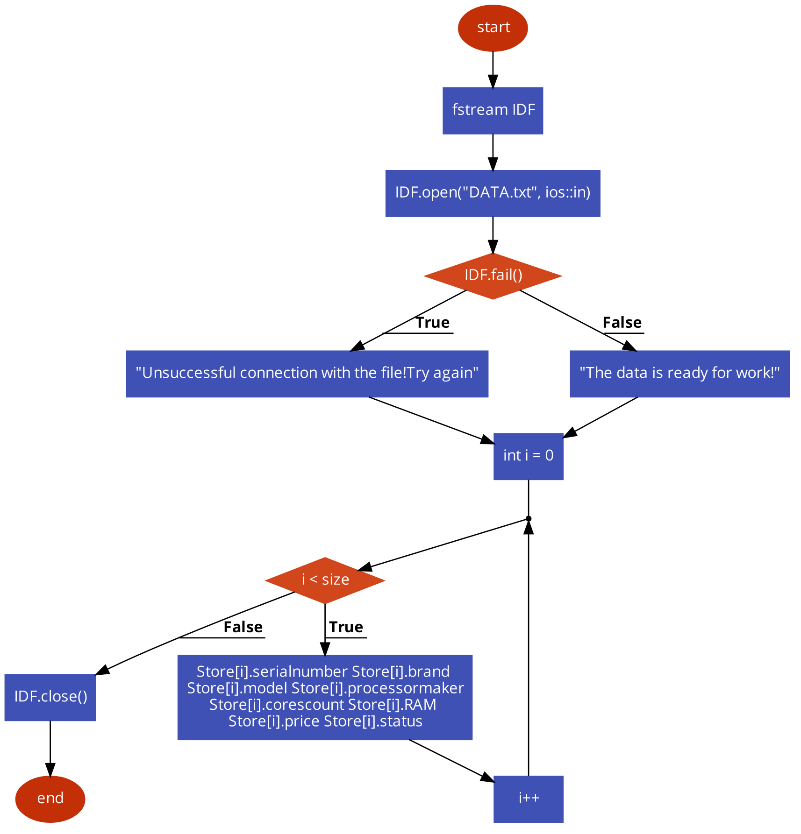
Реализация на допълнение трето

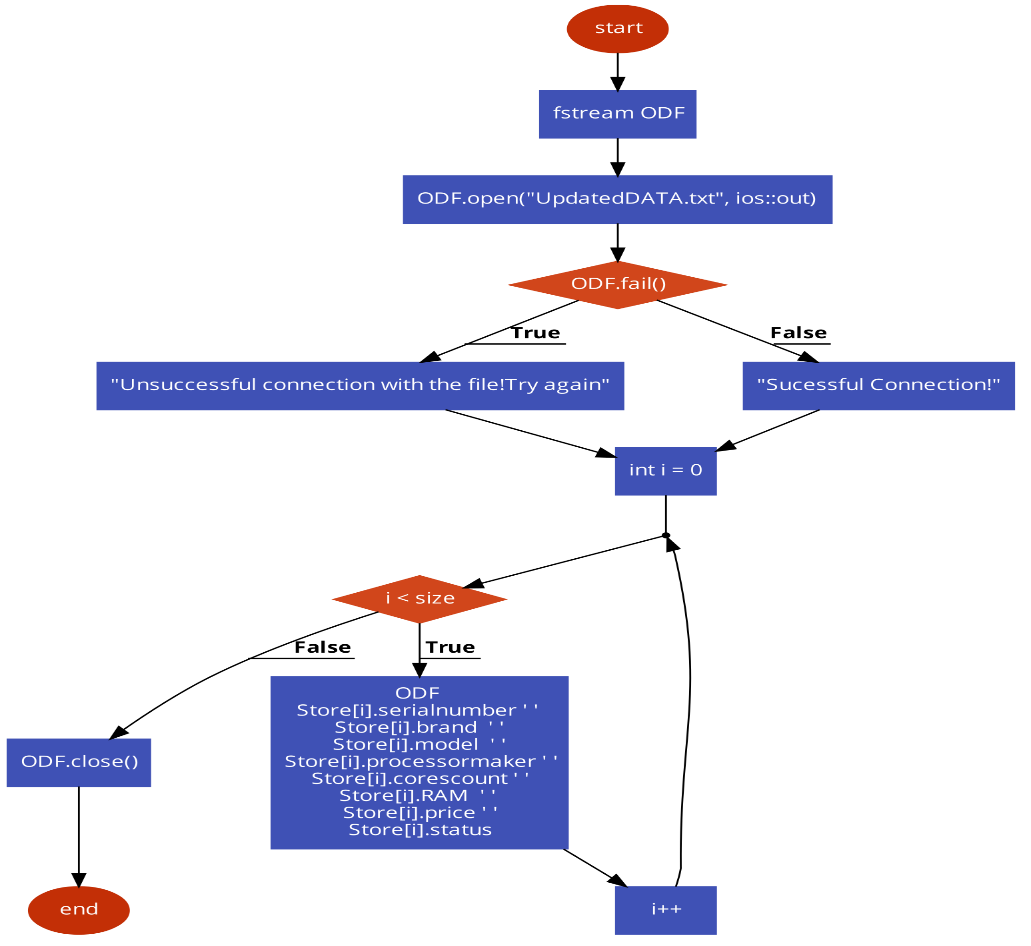
Анализ на алгоритъма, който трябва да се реализира

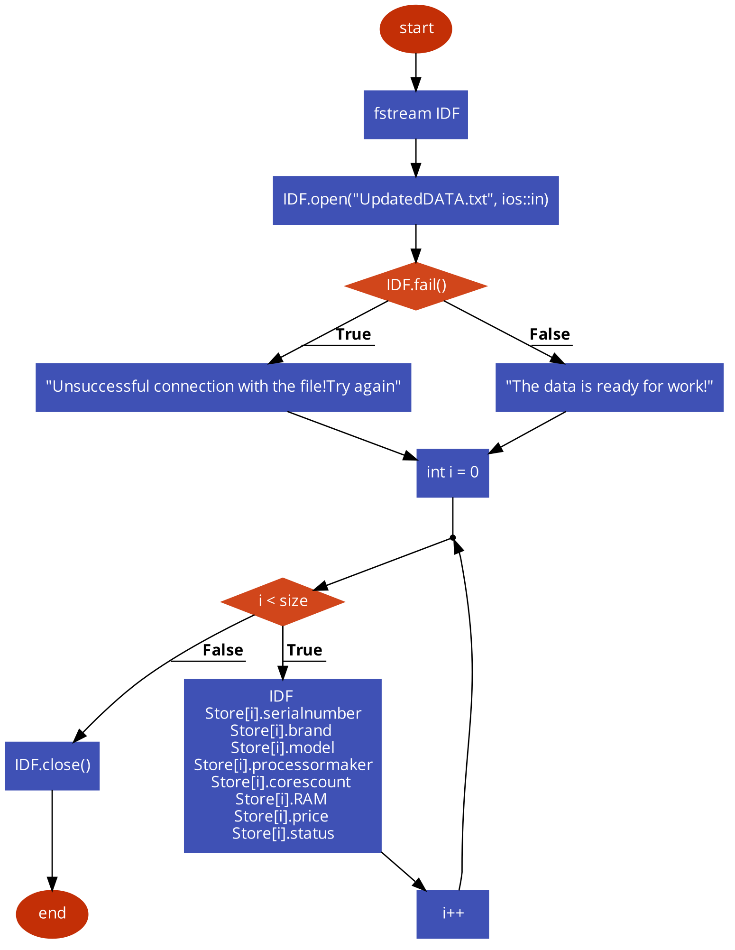
1. При menu=8 се отваря подменю 8, което дава да се изберат 4 стойности за подменюто - 1,2,3 и 4(контролираща променлива на подменю 8-submenu8)
2. Самото меню е в цикъл do{}while()
3. 1)🡪InputData()
4. Създаване на входно-изходна файлова променлива
5. Отваря се файл със статус ios::in
6. Прави се проверка дали има връзка или не
7. Отваря се цикъл for за обхождане на масива
8. Четене на данни и записването им в масив от структури
9. Затваряне на файла
10. 2)🡪 OutputData ()
11. Създаване на входно-изходна файлова променлива
12. Отваря се файл със статус ios::out
13. Прави се проверка дали има връзка или не
14. Отваря се цикъл for за обхождане на масива
15. Записване на данни от масива във файла
16. Затваряне на файла
17. 3)🡪 InputUpdatedData();
18. Създаване на входно-изходна файлова променлива
19. Отваря се файл със статус ios::in
20. Прави се проверка дали има връзка или не
21. Отваря се цикъл for за обхождане на масива
22. Четене на данни и записването им в масив от структури
23. Затваряне на файла
24. 4)🡪Излиза от подменю 8 и затваря цикъла и се връща в главното меню

Картина, която съдържа текст, монитор, екран, екранна снимка

Описанието е генерирано автоматичноБлок схема на алгоритъма







Функция с която е реализиран алгоритъма

void InputData(struct ComputerConfiguration Store[], const unsigned int size)

{

fstream IDF;

IDF.open("DATA.txt", ios::in);

if (IDF.fail()) cout << "Unsuccessful connection with the file!Try again" << endl;

else cout << "The data is ready for work!" << endl;

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

{

IDF >> Store[i].serialnumber >> Store[i].brand >> Store[i].model >> Store[i].processormaker

>> Store[i].corescount >> Store[i].RAM >> Store[i].price >> Store[i].status;

}

IDF.close();

}

void OutputData(struct ComputerConfiguration Store[], const unsigned int size)

{

fstream ODF;

ODF.open("UpdatedDATA.txt", ios::out);

if (ODF.fail()) cout << "Unsuccessful connection with the file!Try again" << endl;

else cout << "Sucessful Connection!" << endl;

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

{

ODF << Store[i].serialnumber <<' '<< Store[i].brand << ' ' << Store[i].model << ' ' << Store[i].processormaker

<< ' ' << Store[i].corescount << ' ' << Store[i].RAM << ' ' << Store[i].price << ' ' << Store[i].status<<endl;

}

ODF.close();

}

void InputUpdatedData(struct ComputerConfiguration Store[], const unsigned int size)

{

fstream IDF;

IDF.open("UpdatedDATA.txt", ios::in);

if (IDF.fail()) cout << "Unsuccessful connection with the file!Try again" << endl;

else cout << "The data is ready for work!" << endl;

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

{

IDF >> Store[i].serialnumber >> Store[i].brand >> Store[i].model >> Store[i].processormaker

>> Store[i].corescount >> Store[i].RAM >> Store[i].price >> Store[i].status;

}

IDF.close();

}

void SUBMENU8(struct ComputerConfiguration Store[], int &size)//8

{

int submenu8;

do

{

cout << endl;

cout << "Submenu 8 with functions:" << endl;

cout << "1. InputData - starting data for the program (from a file)" << endl;

cout << "2. OutputData - recording the new and updated data on a file" << endl;

cout << "3. InputUpdatedData - reading and taking the new and updated data from the file from point (2)" << endl;

cout << "4. Exit from Submenu 8" << endl;

do

{

cout << endl << "Choose 1,2,3 or 4: "; cin >> submenu8;

} while (submenu8 < 1 || submenu8>4);

switch (submenu8)

{

case 1:

{

InputData(Store, size);

do

{

cout << "Enter the count of the Computer Configurations:";

cin >> size;

} while (size < 2 || size>100); break;

}

case 2: OutputData(Store, size); break;

case 3:

{

do

{

cout << "Enter the count of the Computer Configurations:";

cin >> size;

} while (size < 2 || size>100);

InputUpdatedData(Store, size);

break;

}

case 4: break;

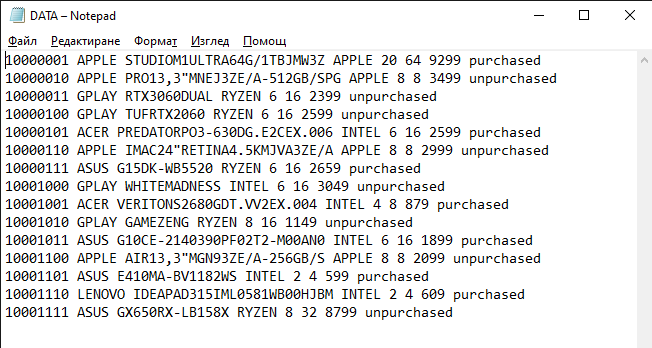
default: cout << "ERROR!"; system("PAUSE"); exit(1);

}

} while (submenu8 != 4);

}

#### Входни данни на функцията



#### Изходни данни на функцията или данни, които се извеждат

Картина, която съдържа текст, вестник, екранна снимка

Описанието е генерирано автоматично

Упътване за употреба

**За правилна работа на програмата се изисква да се спазват стриктно типа на данните!!!**

#include<iostream>

#include<iomanip>

#include<string>

#include<fstream>

#include<cstdlib>

using namespace std;

struct ComputerConfiguration

{

char serialnumber[8];

string brand;

string model;

string processormaker;

int corescount;

int RAM;

double price;

string status;

};

void AddNew(struct ComputerConfiguration Store[],int &size,const unsigned int maxsize)//1

{

int n, status;

do { cout << "Count of NEW Configurations:"; cin >> n; } while (n<1 || n>(maxsize - size));

int backup = size;

size = size + n;

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

{

cout << "Enter Serial Number !(8 digits ONLY)!:"; cin >> Store[backup+i].serialnumber;

cout << "Enter Brand:"; getline(cin.ignore(1000,'\n'), Store[backup+i].brand);

cout << "Enter Model:"; getline(cin, Store[backup + i].model);

cout << "Enter Maker of the Processor:"; getline(cin, Store[backup+i].processormaker);

cout << "Enter Count of Cores:"; cin>>Store[backup+i].corescount;

cout << "Enter RAM:"; cin >> Store[backup+i].RAM;

cout << "Enter Price:"; cin >> Store[backup + i].price;

cout << "Select Status:"<<endl;

cout << "1) purchased" << endl;

cout << "2) unpurchased!" << endl;

do { cout << "Select 1 or 2 to continue:"; cin >> status; } while (status != 1 && status != 2);

switch (status)

{

case 1: Store[backup + i].status = "purchased"; break;

case 2: Store[backup + i].status = "unpurchased"; break;

default: cout << "ERROR!"; system("PAUSE"); exit(1);

}

}

cout << endl;

}

void ShowData(struct ComputerConfiguration Store[],const unsigned int size)//2

{

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

{

cout << "Serial Number: " << Store[i].serialnumber << "\nBrand: " << Store[i].brand

<< "\nModel: " << Store[i].model << "\nProcessor(made by): " << Store[i].processormaker

<< "\nCores: " << Store[i].corescount << "\nRAM: " << Store[i].RAM << " GB \nPrice: "

<< Store[i].price << "LV. \nStatus: " << Store[i].status << endl << endl;

}

cout << endl;

}

void MaxRAM(struct ComputerConfiguration Store[], const unsigned int size)//3a

{

int max = Store[0].RAM;

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

{

if (Store[i].RAM>max) max = Store[i].RAM;

}

cout << endl << "MAX RAM = " << max << " GB" << endl;

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

{

if (Store[i].RAM == max)

{

cout << "Serial Number: " << Store[i].serialnumber << "\nBrand:" << Store[i].brand

<< "\nModel: " << Store[i].model << "\nProcessor(made by): " << Store[i].processormaker

<< "\nCores: " << Store[i].corescount << "\nRAM: " << Store[i].RAM << " GB \nPrice: "

<< Store[i].price << "LV. \nStatus: " << Store[i].status << endl << endl;

}

}

}

void ProcessorMakerSearch(struct ComputerConfiguration Store[], const unsigned int size,string maker)//3b

{

int br = 0;

cout << endl << "Processor Maker: " << maker << endl<<endl;

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

{

if (Store[i].processormaker == maker)

{

br++;

cout << "Serial Number: " << Store[i].serialnumber << "\nBrand:" << Store[i].brand

<< "\nModel: " << Store[i].model << "\nProcessor(made by): " << Store[i].processormaker

<< "\nCores: " << Store[i].corescount << "\nRAM: " << Store[i].RAM << " GB \nPrice: "

<< Store[i].price << "LV. \nStatus: " << Store[i].status << endl << endl;

}

}

if (br == 0) cout << "We dont have processor made by " + maker<<endl<<endl;

}

void SUBMENU3(struct ComputerConfiguration Store[], const unsigned int size)//3

{

char submenu3;

string maker;

do

{

cout << endl;

cout << "Submenu 3 with functions:" << endl;

cout << "3.A) Max RAM configurations" << endl;

cout << "3.B) Search by maker of the processor" << endl;

cout << "3.C) Exit from Submenu 3" << endl;

do

{

cout << endl << "Choose A, B or C: "; cin >> submenu3;

} while (submenu3 != 'A' && submenu3 != 'B' && submenu3 != 'C');

switch (submenu3)

{

case 'A': MaxRAM(Store,size); break;

case 'B': cout << "Search by Processor Maker:";

getline(cin.ignore(1000,'\n'), maker); ProcessorMakerSearch(Store,size,maker); break;

case 'C': break;

default: cout << "ERROR!"; system("PAUSE"); exit(1);

}

} while (submenu3 != 'C');

}

void SortByPrice(struct ComputerConfiguration Store[], const unsigned int size)//4

{

ComputerConfiguration buf;

for (int k = size; k > 1; k--)

{

for (int i = 0; i < k - 1; i++)

{

//if (Store[i].price < Store[i + 1].price) //низходящ ред

if (Store[i].price > Store[i + 1].price) //възходящ ред

{

buf = Store[i];

Store[i] = Store[i + 1];

Store[i + 1] = buf;

}

}

}

}

void OutputDataBinary(struct ComputerConfiguration Store[], const unsigned int size)//5

{

fstream ODFB;

ODFB.open("UpdatedDATA-Binary.dat", ios::binary | ios::out);

if (ODFB.fail()) cout << "Unsuccessful connection with the file!Try again" << endl;

else cout << "Sucessful Connection!" << endl;

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

{

ODFB.write(Store[i].serialnumber, 8);

ODFB.write((char\*)&Store[i].brand, sizeof(Store[i].brand));

ODFB.write((char\*)&Store[i].model, sizeof(Store[i].model));

ODFB.write((char\*)&Store[i].processormaker, sizeof(Store[i].processormaker));

ODFB.write((char\*)&Store[i].corescount, sizeof(Store[i].corescount));

ODFB.write((char\*)&Store[i].RAM, sizeof(Store[i].RAM));

ODFB.write((char\*)&Store[i].price, sizeof(Store[i].price));

ODFB.write((char\*)&Store[i].status, sizeof(Store[i].status));

}

ODFB.close();

}

void InputDataBinary(struct ComputerConfiguration Store[], const unsigned int size)//5

{

fstream IDFB;

IDFB.open("UpdatedDATA-Binary.dat", ios::binary | ios::in);

if (IDFB.fail()) cout << "Unsuccessful connection with the file!Try again" << endl;

else cout << "Sucessful Connection!" << endl;

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

{

IDFB.read(Store[i].serialnumber, 8);

IDFB.read((char\*)&Store[i].brand, sizeof(Store[i].brand));

IDFB.read((char\*)&Store[i].model, sizeof(Store[i].model));

IDFB.read((char\*)&Store[i].processormaker, sizeof(Store[i].processormaker));

IDFB.read((char\*)&Store[i].corescount, sizeof(Store[i].corescount));

IDFB.read((char\*)&Store[i].RAM, sizeof(Store[i].RAM));

IDFB.read((char\*)&Store[i].price, sizeof(Store[i].price));

IDFB.read((char\*)&Store[i].status, sizeof(Store[i].status));

}

IDFB.close();

}

void UnsoldSortedBySerialNumberConfiguration(struct ComputerConfiguration Store[], const unsigned int size)//6a

{

cout << endl;

cout << "Unpurchased computers sorted by serial number" << endl;

ComputerConfiguration buf;

for (int k = size; k > 1; k--)

{

for (int i = 0; i < k - 1; i++)

{

//if (strcmp(Store[i].serialnumber,Store[i+1].serialnumber)<0) //низходящ ред

if (strcmp(Store[i].serialnumber,Store[i+1].serialnumber)>0) //възходящ ред

{

buf = Store[i];

Store[i] = Store[i + 1];

Store[i + 1] = buf;

}

}

}

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

{

if (Store[i].status == "unpurchased")

{

cout << "Serial Number: " << Store[i].serialnumber << "\nBrand:" << Store[i].brand

<< "\nModel: " << Store[i].model << "\nProcessor(made by): " << Store[i].processormaker

<< "\nCores: " << Store[i].corescount << "\nRAM: " << Store[i].RAM << " GB \nPrice: "

<< Store[i].price << "LV. \nStatus: " << Store[i].status << endl << endl;

}

}

}

void SearchByBrandAndModel(struct ComputerConfiguration Store[], const unsigned int size,string brand,string model)//6b

{

int br = 0;

cout << "Brand: " << brand;

cout << endl << "Model: " << model << endl << endl;

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

{

if (Store[i].brand == brand&&Store[i].model==model)

{

br++;

cout << "Serial Number: " << Store[i].serialnumber << "\nBrand:" << Store[i].brand

<< "\nModel: " << Store[i].model << "\nProcessor(made by): " << Store[i].processormaker

<< "\nCores: " << Store[i].corescount << "\nRAM: " << Store[i].RAM << " GB \nPrice: "

<< Store[i].price << "LV. \nStatus: " << Store[i].status << endl << endl;

}

}

if (br == 0) cout << "We dont have computer with brand " + brand + " and model " + model << endl;

}

void SUBMENU6(struct ComputerConfiguration Store[], const unsigned int size)//6

{

char submenu6;

string brand,model;

do

{

cout << endl;

cout << "Submenu 6 with functions:" << endl;

cout << "6.A) Sort by serial number and show unpurchased configurations" << endl;

cout << "6.B) Search by Brand and Model" << endl;

cout << "6.C) Exit from Submenu 6" << endl;

do

{

cout << endl << "Choose A, B or C: "; cin >> submenu6;

} while (submenu6 != 'A' && submenu6 != 'B' && submenu6 != 'C');

switch (submenu6)

{

case 'A': UnsoldSortedBySerialNumberConfiguration(Store,size); break;

case 'B': cout << "Enter Brand:";

getline(cin.ignore(1000, '\n'), brand);

cout << "Enter Model:";

getline(cin, model);

SearchByBrandAndModel(Store,size,brand,model); break;

case 'C': break;

default: cout << "ERROR!"; system("PAUSE"); exit(1);

}

} while (submenu6 != 'C');

}

void SearchByBrandAndModel2(struct ComputerConfiguration Store[], const unsigned int size,string brand, string model)//7a

{

int br = 0;

cout << "Brand: " << brand;

cout << endl << "Model: " << model << endl << endl;

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

{

if (Store[i].brand == brand && Store[i].model == model&& Store[i].status == "unpurchased")

{

br++;

cout << "Serial Number: " << Store[i].serialnumber << "\nBrand:" << Store[i].brand

<< "\nModel: " << Store[i].model << "\nProcessor(made by): " << Store[i].processormaker

<< "\nCores: " << Store[i].corescount << "\nRAM: " << Store[i].RAM << " GB \nPrice: "

<< Store[i].price << "LV. \nStatus: " << Store[i].status << endl << endl;

}

}

if (br == 0) cout << "We dont have computer with brand " + brand + " and model " + model + " or it is purchased!" << endl;

}

void ProcessorMakerSearch2(struct ComputerConfiguration Store[], const unsigned int size,string maker)//7a

{

int br = 0;

cout << endl << "Processor Maker: " << maker << endl << endl;

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

{

if (Store[i].processormaker == maker && Store[i].status == "unpurchased")

{

br++;

cout << "Serial Number: " << Store[i].serialnumber << "\nBrand:" << Store[i].brand

<< "\nModel: " << Store[i].model << "\nProcessor(made by): " << Store[i].processormaker

<< "\nCores: " << Store[i].corescount << "\nRAM: " << Store[i].RAM << " GB \nPrice: "

<< Store[i].price << "LV. \nStatus: " << Store[i].status << endl << endl;

}

}

if (br == 0) cout << "We dont have processor made by " + maker + " or it is purchased!" << endl << endl;

}

void SearchByCoresCount(struct ComputerConfiguration Store[], const unsigned int size,int cores)//7a

{

int br = 0;

cout << endl << "Cores: " << cores << endl << endl;

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

{

if (Store[i].corescount == cores && Store[i].status == "unpurchased")

{

br++;

cout << "Serial Number: " << Store[i].serialnumber << "\nBrand:" << Store[i].brand

<< "\nModel: " << Store[i].model << "\nProcessor(made by): " << Store[i].processormaker

<< "\nCores: " << Store[i].corescount << "\nRAM: " << Store[i].RAM << " GB \nPrice: "

<< Store[i].price << "LV. \nStatus: " << Store[i].status << endl << endl;

}

}

if (br == 0) cout << "We dont have computer configuration with " <<cores<<" cores or it is purchased!" << endl << endl;

}

void SearchByRAM(struct ComputerConfiguration Store[], const unsigned int size,int RAM)//7a

{

int br = 0;

cout << endl << "RAM: " << RAM << endl << endl;

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

{

if (Store[i].RAM == RAM && Store[i].status == "unpurchased")

{

br++;

cout << "Serial Number: " << Store[i].serialnumber << "\nBrand:" << Store[i].brand

<< "\nModel: " << Store[i].model << "\nProcessor(made by): " << Store[i].processormaker

<< "\nCores: " << Store[i].corescount << "\nRAM: " << Store[i].RAM << " GB \nPrice: "

<< Store[i].price << "LV. \nStatus: " << Store[i].status << endl << endl;

}

}

if (br == 0) cout << "We dont have computer configuration with " << RAM << "GB (RAM) or it is purchased!" << endl << endl;

}

void MaxPrice(struct ComputerConfiguration Store[], const unsigned int size)//7a

{

double max=0;

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

{

if (Store[i].status == "unpurchased") max = Store[i].price; break;

}

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

{

if (Store[i].price > max && Store[i].status=="unpurchased") max = Store[i].price;

}

cout << endl << "MAX Price = " << max << " LV" << endl;

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

{

if (Store[i].price == max && Store[i].status == "unpurchased")

{

cout << "Serial Number: " << Store[i].serialnumber << "\nBrand:" << Store[i].brand

<< "\nModel: " << Store[i].model << "\nProcessor(made by): " << Store[i].processormaker

<< "\nCores: " << Store[i].corescount << "\nRAM: " << Store[i].RAM << " GB \nPrice: "

<< Store[i].price << "LV. \nStatus: " << Store[i].status << endl;

}

}

}

void SearchingForTheRightConfigurationMenu(struct ComputerConfiguration Store[], const unsigned int size)//7a-menu

{

int submenu7;

string brand, model,maker;

int cores, RAM;

do

{

cout << endl;

cout << "Subsubmenu 7.1 with functions --> !(Unpurchased Computer Configurations ONLY)! Search By:" << endl;

cout << "7.1.(1) Brand and Model " << endl;

cout << "7.1.(2) Maker of the processor" << endl;

cout << "7.1.(3) Count of the cores " << endl;

cout << "7.1.(4) RAM" << endl;

cout << "7.1.(5) MAX Price" << endl;

cout << "7.1.(6) Exit from Submenu 7.1" << endl;

do

{

cout << endl << "Choose a number from 1 to 6: "; cin >> submenu7;

} while (submenu7 < 1 || submenu7>6);

switch (submenu7)

{

case 1: cout << "Enter Brand:";

getline(cin.ignore(1000, '\n'), brand);

cout << "Enter Model:";

getline(cin, model);

SearchByBrandAndModel2(Store,size,brand, model); break;

case 2: cout << "Search by Processor Maker:";

getline(cin.ignore(1000, '\n'), maker); ProcessorMakerSearch2(Store,size,maker); break;

case 3: cout << "Enter cores count: "; cin >> cores;

SearchByCoresCount(Store,size,cores); break;

case 4: cout << "Enter RAM: "; cin >> RAM;

SearchByRAM(Store,size,RAM); break;

case 5: MaxPrice(Store,size); break;

case 6: break;

default: cout << "ERROR!"; system("PAUSE"); exit(1);

}

} while (submenu7 != 6);

}

void Selling(struct ComputerConfiguration Store[], const unsigned int size)//7b

{

bool flag = false;

string wantednumber;

int buy,discount,discountcode;

cout << "Enter Serial Number !(8 digits ONLY)!:"; getline(cin.ignore(1000,'\n'),wantednumber);

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

{

if (Store[i].serialnumber==wantednumber)//!(strcmp(Store[i].serialnumber, wantednumber))

{

flag = true;

if (Store[i].status == "unpurchased")

{

cout << "Serial Number: " << Store[i].serialnumber << "\nBrand:" << Store[i].brand

<< "\nModel: " << Store[i].model << "\nProcessor(made by): " << Store[i].processormaker

<< "\nCores: " << Store[i].corescount << "\nRAM: " << Store[i].RAM << " GB \nPrice: "

<< Store[i].price << "LV. \nStatus: " << Store[i].status << endl;

cout << "Do you want to buy it?" << endl;

cout << "1) YES!" << endl;

cout << "2) NO!" << endl;

do { cout << "Select 1 or 2 to continue:"; cin >> buy; } while (buy != 1 && buy != 2);

switch (buy)

{

case 1:

{

cout << endl;

cout << "Do you have a discount code?" << endl;

cout << "1) YES!" << endl;

cout << "2) NO!" << endl;

do { cout << "Select 1 or 2 to continue:"; cin >> discount; } while (discount != 1 && discount != 2);

switch (discount)

{

case 1:

{

cout << "Enter discount code:"; cin >> discountcode;

if (sizeof(discountcode) == 4 && discountcode % 8 == 0 && discountcode % 7 == 0)

{

cout << "Old Price: " << Store[i].price << " LV." << endl;

cout << "New Price: " << Store[i].price - (Store[i].price \* 0.05) << " LV." << endl;

cout << ":) !SUCCESSFUL PURCHASE! :)" << endl;

Store[i].status = "purchased";

}

else

{

cout << "Price: " << Store[i].price << " LV." << endl;

cout << ":) !SUCCESSFUL PURCHASE! :)" << endl;

Store[i].status = "purchased";

}

break;

}

case 2:

{

cout << "Price: " << Store[i].price << " LV." << endl;

cout << ":) !SUCCESSFUL PURCHASE! :)" << endl;

Store[i].status = "purchased"; break;

}

default: cout << "ERROR!"; system("PAUSE"); exit(1);

}

break;

}

case 2: cout << "Thank you!Come again!" << endl; break;

default: cout << "ERROR!"; system("PAUSE"); exit(1);

}

}

else cout << " The Computer Configuration has been sold!" << endl << endl;

}

}

if (flag == false)

{

cout << "It isn't found a Computer Configuration with the specified Serial Number!"

<< endl << endl;

}

}

void SUBMENU7(struct ComputerConfiguration Store[], const unsigned int size)//7

{

char submenu7;

do

{

cout << endl;

cout << "Submenu 7 with functions:" << endl;

cout << "7.A) Searching For The Right Configuration(Menu)" << endl;

cout << "7.B) Buying a Computer Configuration?" << endl;

cout << "7.C) Exit from Submenu 7" << endl;

do

{

cout << endl << "Choose A, B or C: "; cin >> submenu7;

} while (submenu7 != 'A' && submenu7 != 'B' && submenu7 != 'C');

switch (submenu7)

{

case 'A': SearchingForTheRightConfigurationMenu(Store,size); break;

case 'B': Selling(Store,size); break;

case 'C': break;

default: cout << "ERROR!"; system("PAUSE"); exit(1);

}

} while (submenu7 != 'C');

}

void InputData(struct ComputerConfiguration Store[], const unsigned int size)

{

fstream IDF;

IDF.open("DATA.txt", ios::in);

if (IDF.fail()) cout << "Unsuccessful connection with the file!Try again" << endl;

else cout << "The data is ready for work!" << endl;

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

{

IDF >> Store[i].serialnumber >> Store[i].brand >> Store[i].model >> Store[i].processormaker

>> Store[i].corescount >> Store[i].RAM >> Store[i].price >> Store[i].status;

}

IDF.close();

}

void OutputData(struct ComputerConfiguration Store[], const unsigned int size)

{

fstream ODF;

ODF.open("UpdatedDATA.txt", ios::out);

if (ODF.fail()) cout << "Unsuccessful connection with the file!Try again" << endl;

else cout << "Sucessful Connection!" << endl;

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

{

ODF << Store[i].serialnumber <<' '<< Store[i].brand << ' ' << Store[i].model << ' ' << Store[i].processormaker

<< ' ' << Store[i].corescount << ' ' << Store[i].RAM << ' ' << Store[i].price << ' ' << Store[i].status<<endl;

}

ODF.close();

}

void InputUpdatedData(struct ComputerConfiguration Store[], const unsigned int size)

{

fstream IDF;

IDF.open("UpdatedDATA.txt", ios::in);

if (IDF.fail()) cout << "Unsuccessful connection with the file!Try again" << endl;

else cout << "The data is ready for work!" << endl;

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

{

IDF >> Store[i].serialnumber >> Store[i].brand >> Store[i].model >> Store[i].processormaker

>> Store[i].corescount >> Store[i].RAM >> Store[i].price >> Store[i].status;

}

IDF.close();

}

void SUBMENU8(struct ComputerConfiguration Store[], int &size)//8

{

int submenu8;

do

{

cout << endl;

cout << "Submenu 8 with functions:" << endl;

cout << "1. InputData - starting data for the program (from a file)" << endl;

cout << "2. OutputData - recording the new and updated data on a file" << endl;

cout << "3. InputUpdatedData - reading and taking the new and updated data from the file from point (2)" << endl;

cout << "4. Exit from Submenu 8" << endl;

do

{

cout << endl << "Choose 1,2,3 or 4: "; cin >> submenu8;

} while (submenu8 < 1 || submenu8>4);

switch (submenu8)

{

case 1:

{

InputData(Store, size);

do

{

cout << "Enter the count of the Computer Configurations:";

cin >> size;

} while (size < 2 || size>100); break;

}

case 2: OutputData(Store, size); break;

case 3:

{

do

{

cout << "Enter the count of the Computer Configurations:";

cin >> size;

} while (size < 2 || size>100);

InputUpdatedData(Store, size);

break;

}

case 4: break;

default: cout << "ERROR!"; system("PAUSE"); exit(1);

}

} while (submenu8 != 4);

}

void main()

{

const int MaxComputers = 100;

int menu,ArrayLength;

do

{

cout << "Enter the count of the Computer Configurations:";

cin >> ArrayLength;

} while (ArrayLength < 2 || ArrayLength>100);

ComputerConfiguration Store[MaxComputers];

cout << endl;

InputData(Store,ArrayLength);

do

{

cout << endl;

cout << "Menu with main functions:" << endl;

cout << "1)Add configuration" << endl;

cout << "2)Show all data" << endl;

cout << "3)Searching for data and showing it" << endl;

cout << "4)Sorting the configurations by Price" << endl;

cout << "5)File management" << endl;

cout << "6)Submenu" << endl;

cout << "7)Configuration selling" << endl;

cout << "8)Automatically fill the data from file and record it on file" << endl;

cout << "9)Exit from the program" << endl;

cout <<endl<< "Enter a number from 1 to 9: ";

do

{

cin >> menu;

} while (menu < 1 || menu>9);

cout << endl;

switch (menu)

{

case 1: AddNew(Store, ArrayLength, MaxComputers); break;

case 2: ShowData(Store,ArrayLength); break;

case 3: cout << "WELCOME!" << endl; SUBMENU3(Store, ArrayLength); break;

case 4: SortByPrice(Store, ArrayLength); break;

case 5: OutputDataBinary(Store, ArrayLength); InputDataBinary(Store, ArrayLength); break;

case 6: cout << "WELCOME!" << endl; SUBMENU6(Store, ArrayLength); break;

case 7: cout << "WELCOME!" << endl; SUBMENU7(Store, ArrayLength); break;

case 8: cout << "WELCOME!" << endl; SUBMENU8(Store, ArrayLength); break;

case 9: exit(1); break;

default: cout << "ERROR!"; system("PAUSE"); exit(1);

}

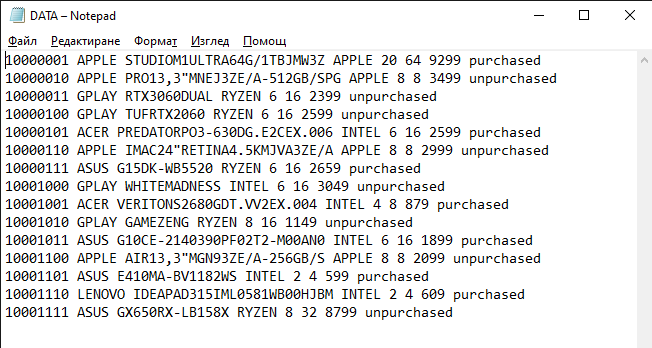
} while (menu != 9);

}

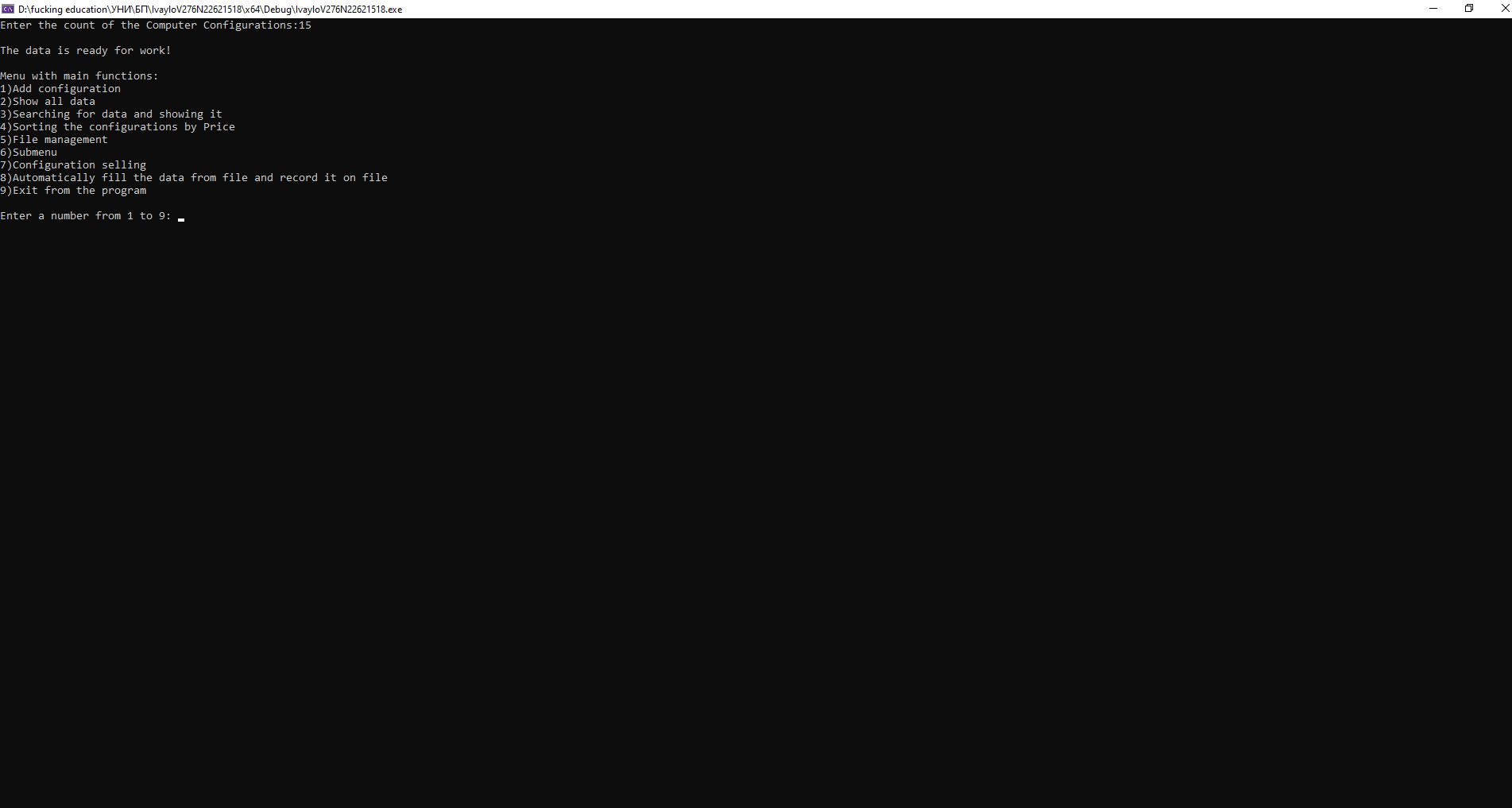
Примерно действие на програмата

Условие A

Снимка на изгледа с примерни входни данни



Снимка на изгледа с примерни изходни данни



Условие B

Снимка на изгледа с примерни входни данни

Картина, която съдържа текст

Описанието е генерирано автоматично

Снимка на изгледа с примерни изходни данни

Картина, която съдържа текст

Описанието е генерирано автоматичноКартина, която съдържа текст

Описанието е генерирано автоматичноКартина, която съдържа текст

Описанието е генерирано автоматичноКартина, която съдържа текст

Описанието е генерирано автоматично

Условие C

Снимка на изгледа с примерни входни данни

В т.2

Снимка на изгледа с примерни изходни данни

В т.2

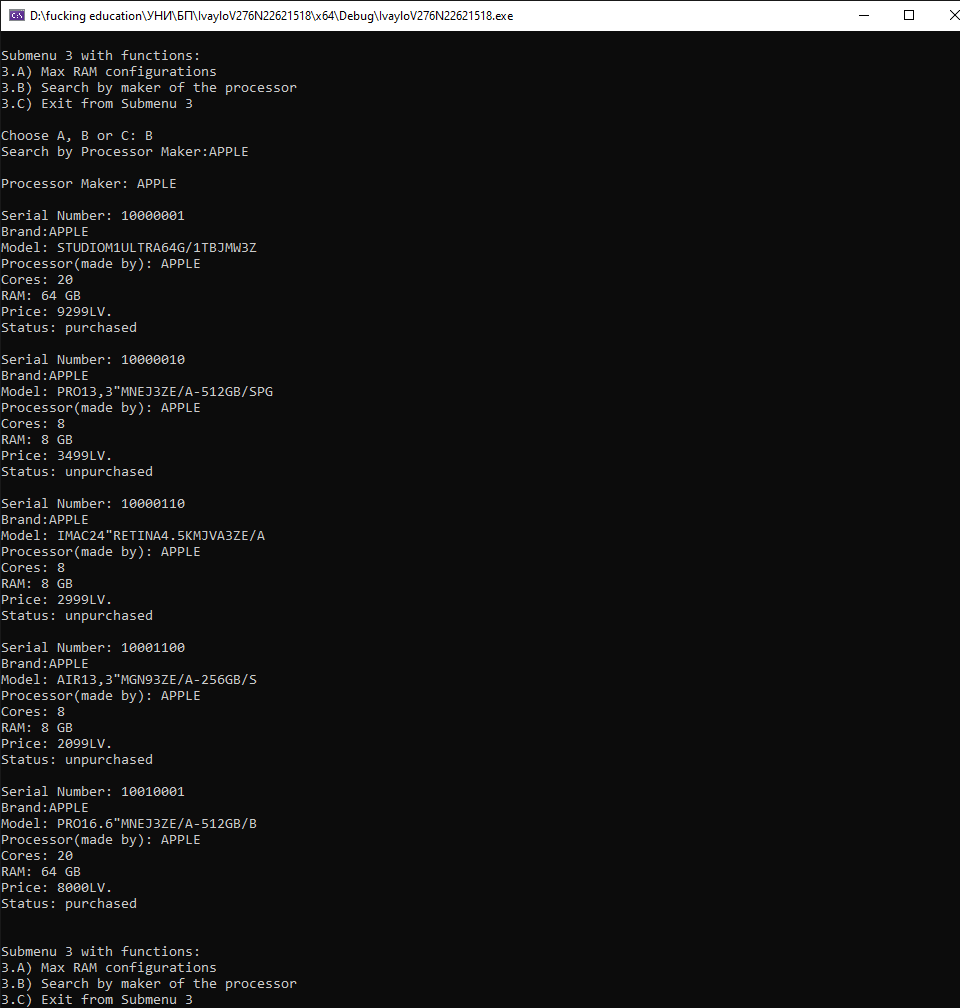
Условие D

Снимка на изгледа с примерни входни данни

Снимка на изгледа с примерни изходни данниКартина, която съдържа текст

Описанието е генерирано автоматично

Картина, която съдържа текст

Описанието е генерирано автоматичноКартина, която съдържа текст

Описанието е генерирано автоматично

Условие E

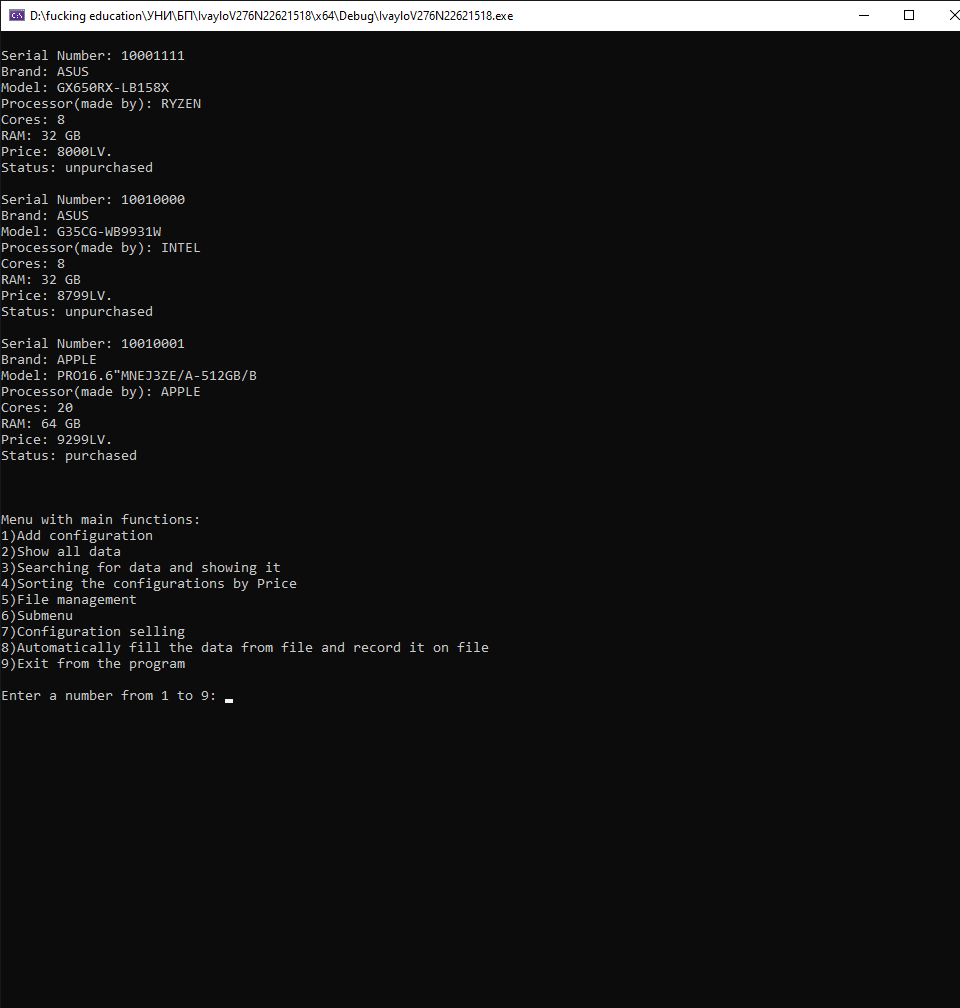
Снимка на изгледа с примерни входни данни

Снимка на изгледа с примерни изходни данниКартина, която съдържа текст

Описанието е генерирано автоматично

Картина, която съдържа текст

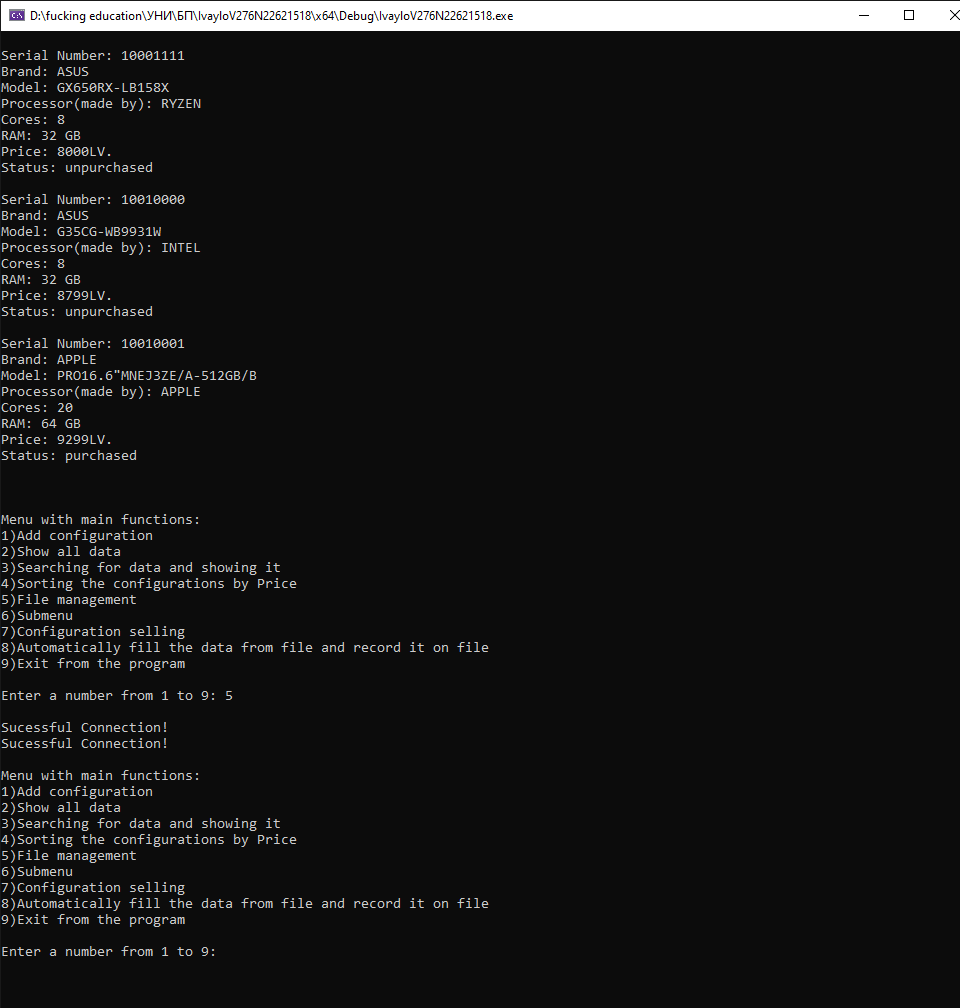
Описанието е генерирано автоматичноКартина, която съдържа текст

Описанието е генерирано автоматично

Условие F

Снимка на изгледа с примерни входни данни

Снимка на изгледа с примерни изходни данни



Допълнение първо

Снимка на изгледа с примерни входни данни

Снимка на изгледа с примерни изходни данниКартина, която съдържа текст

Описанието е генерирано автоматично

Картина, която съдържа текст

Описанието е генерирано автоматично

Картина, която съдържа текст

Описанието е генерирано автоматичноКартина, която съдържа текст

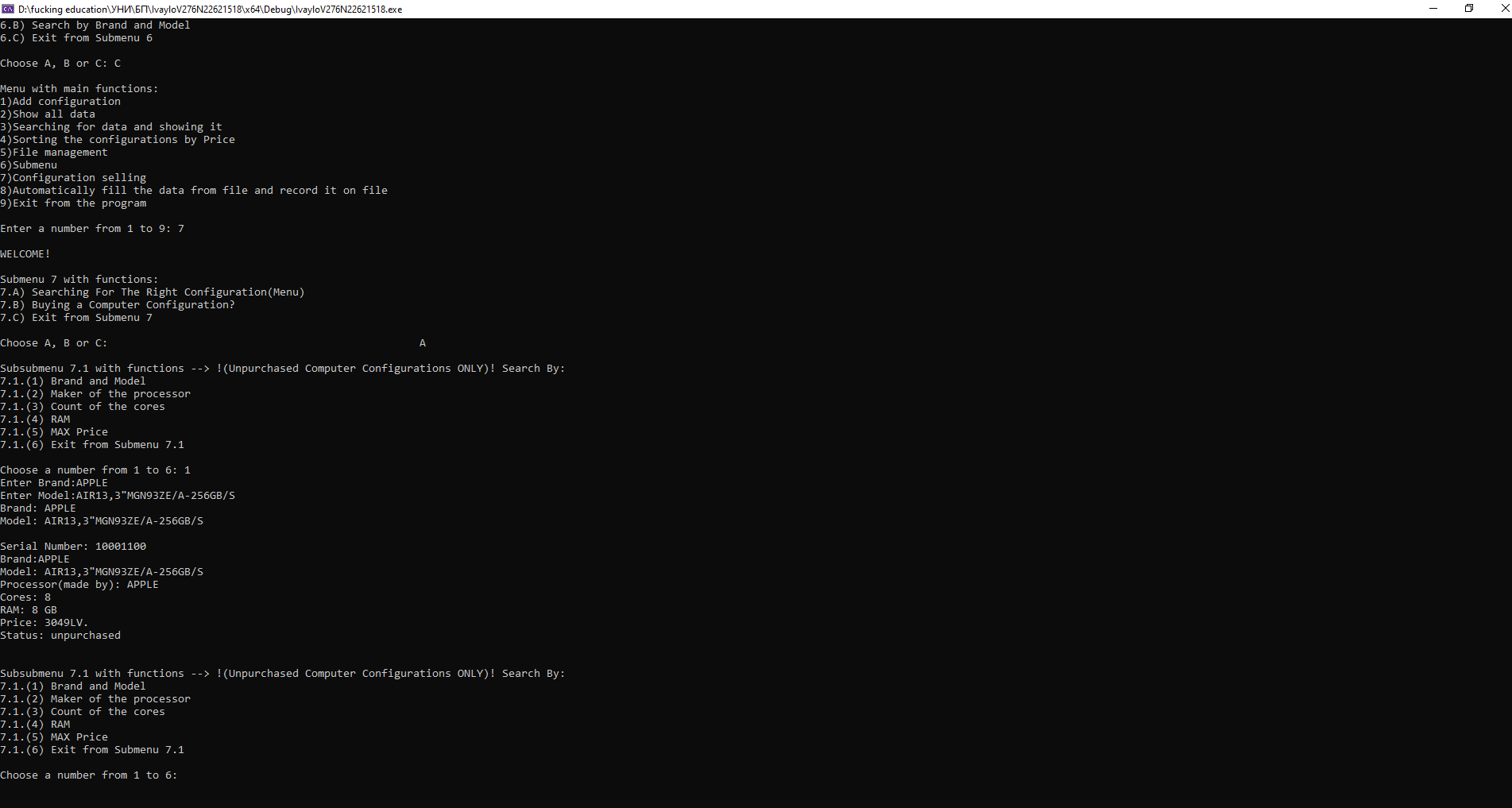
Описанието е генерирано автоматично

Допълнение второ

Снимка на изгледа с примерни входни данни

Снимка на изгледа с примерни изходни данни

Картина, която съдържа текст

Описанието е генерирано автоматичноКартина, която съдържа текст

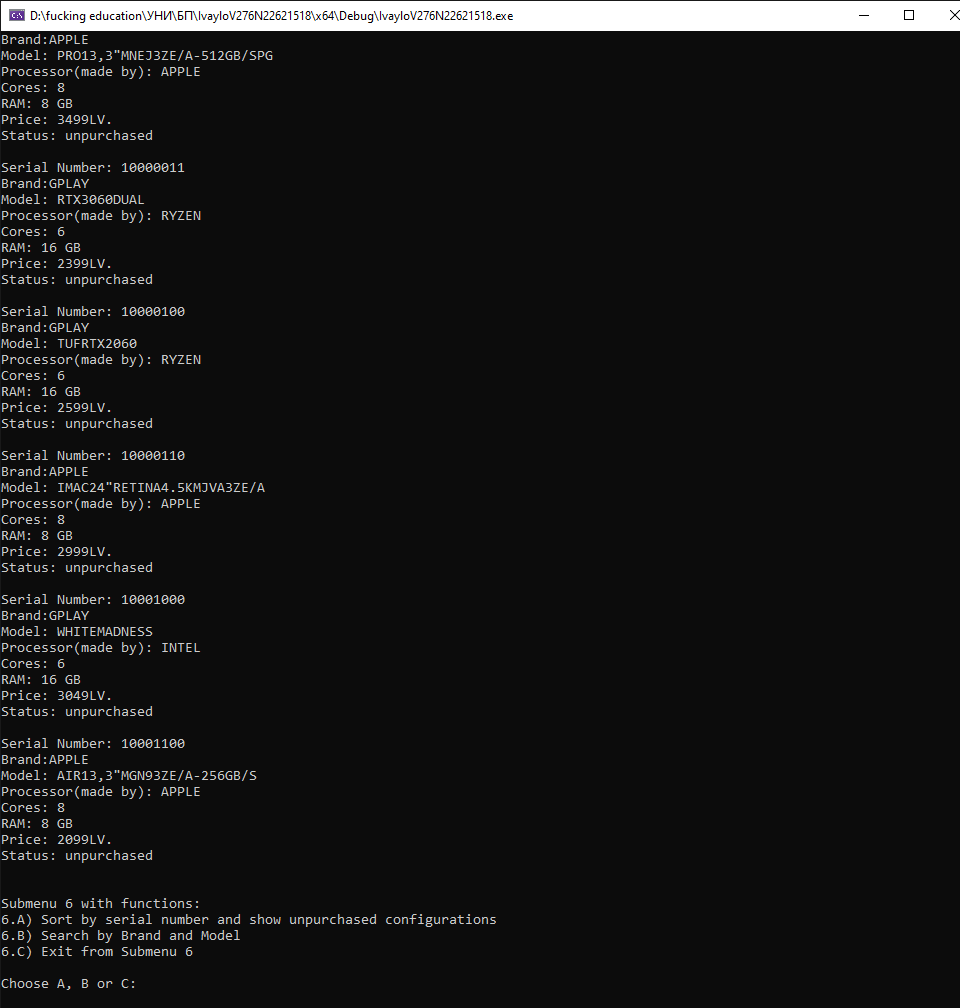
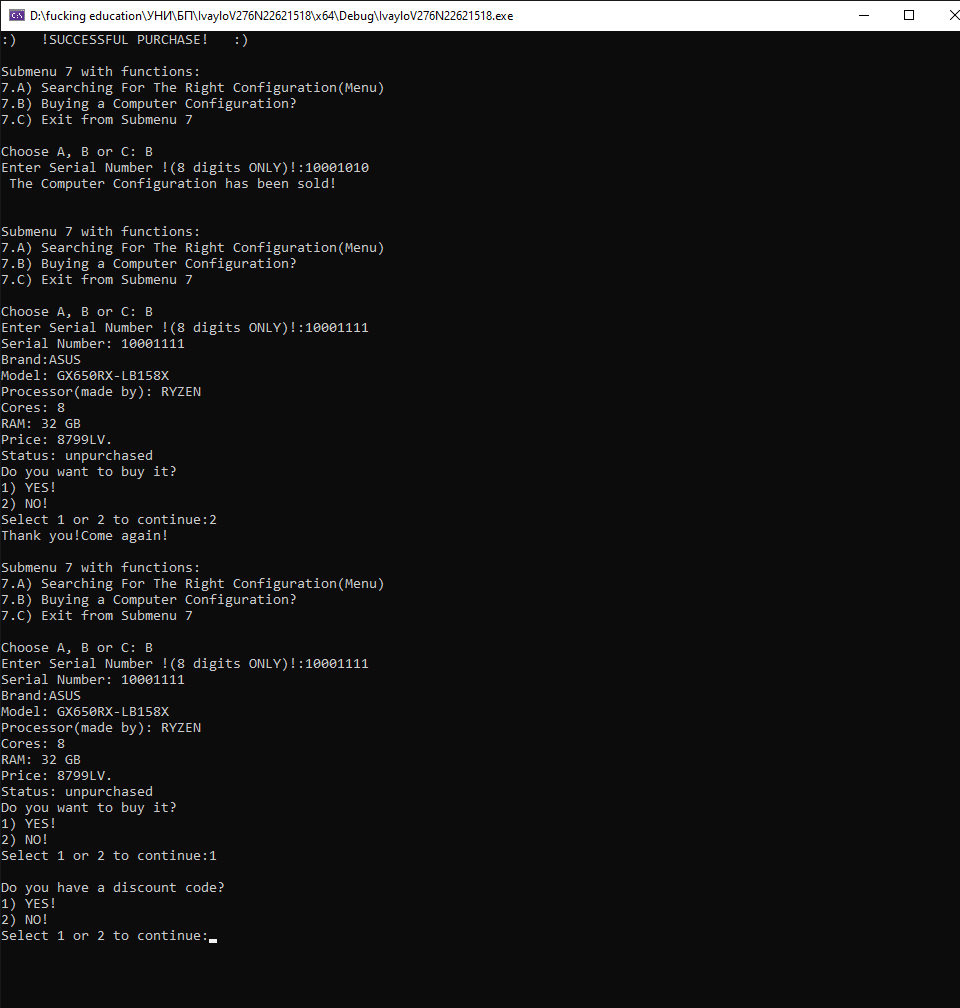
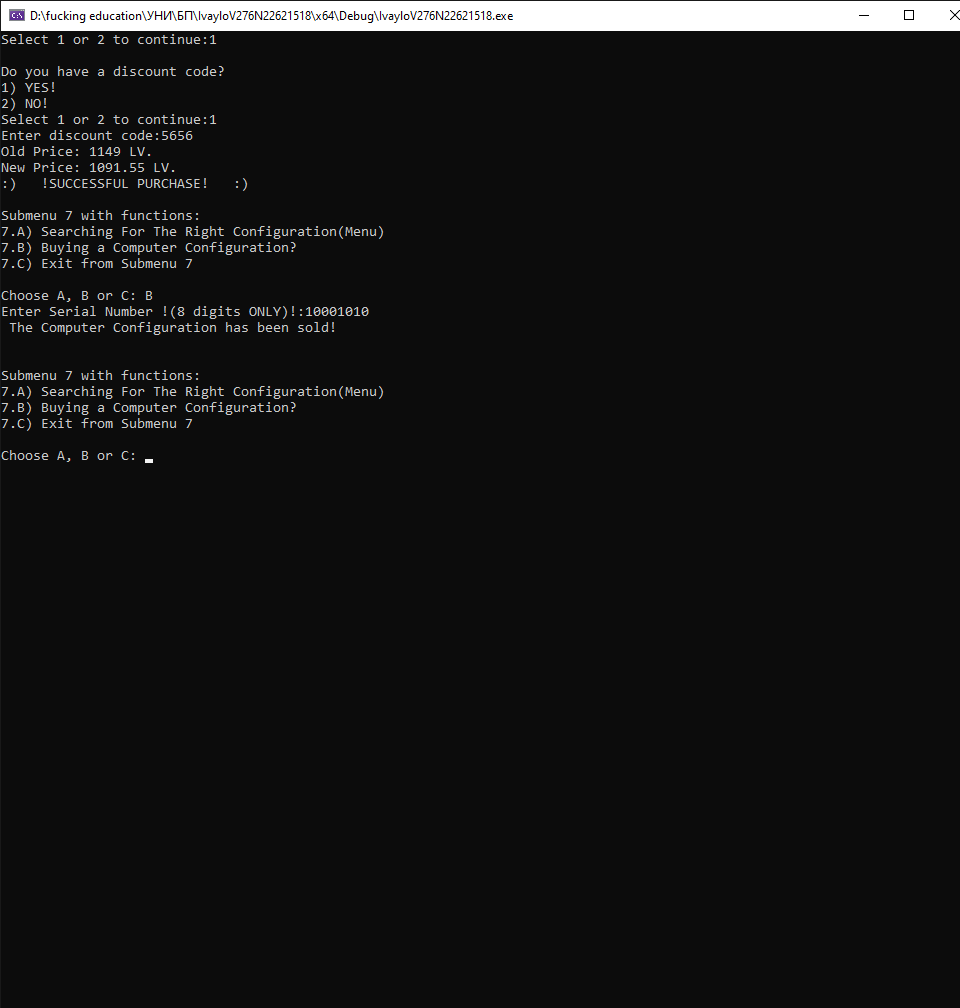
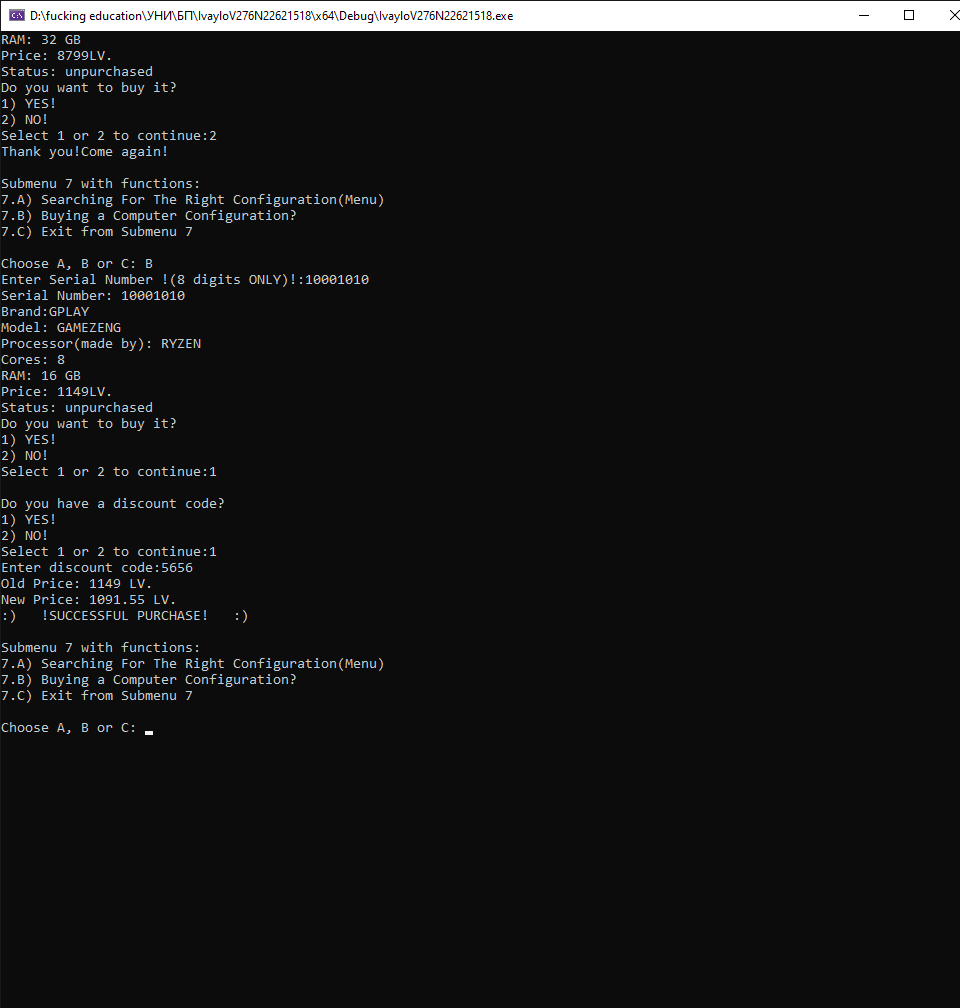
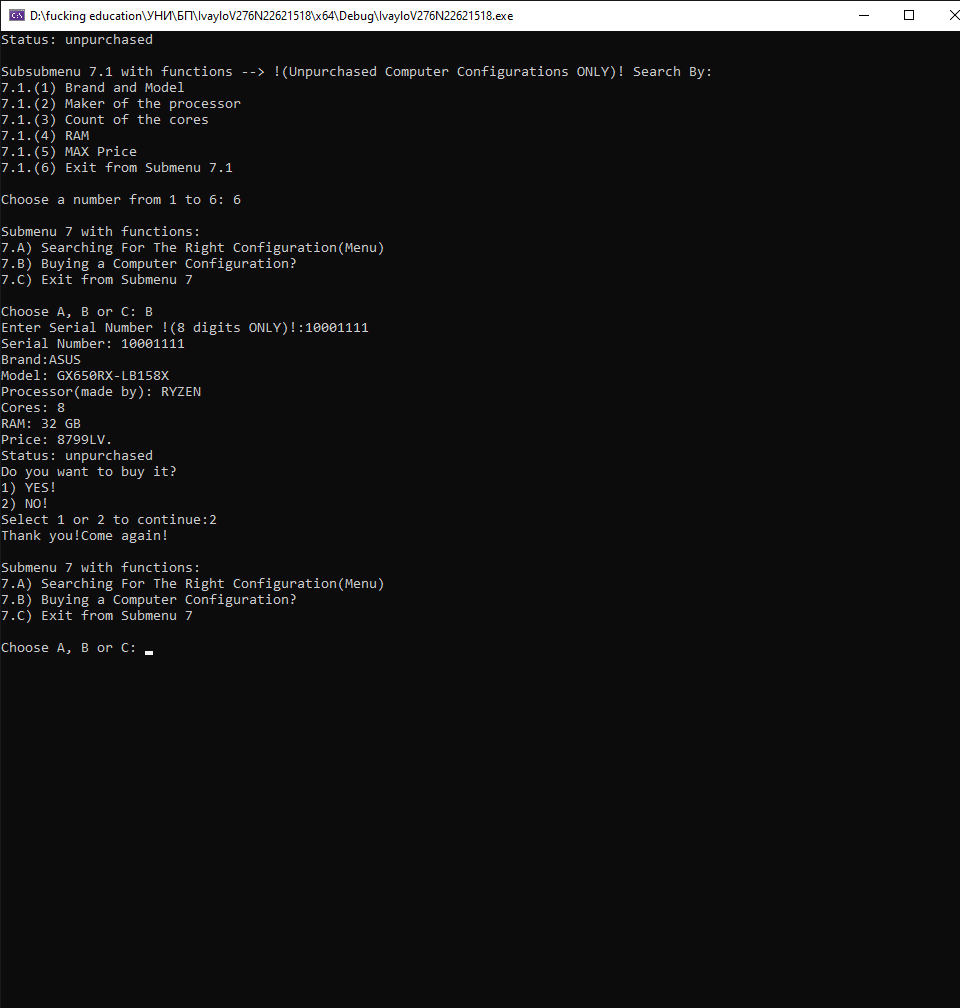
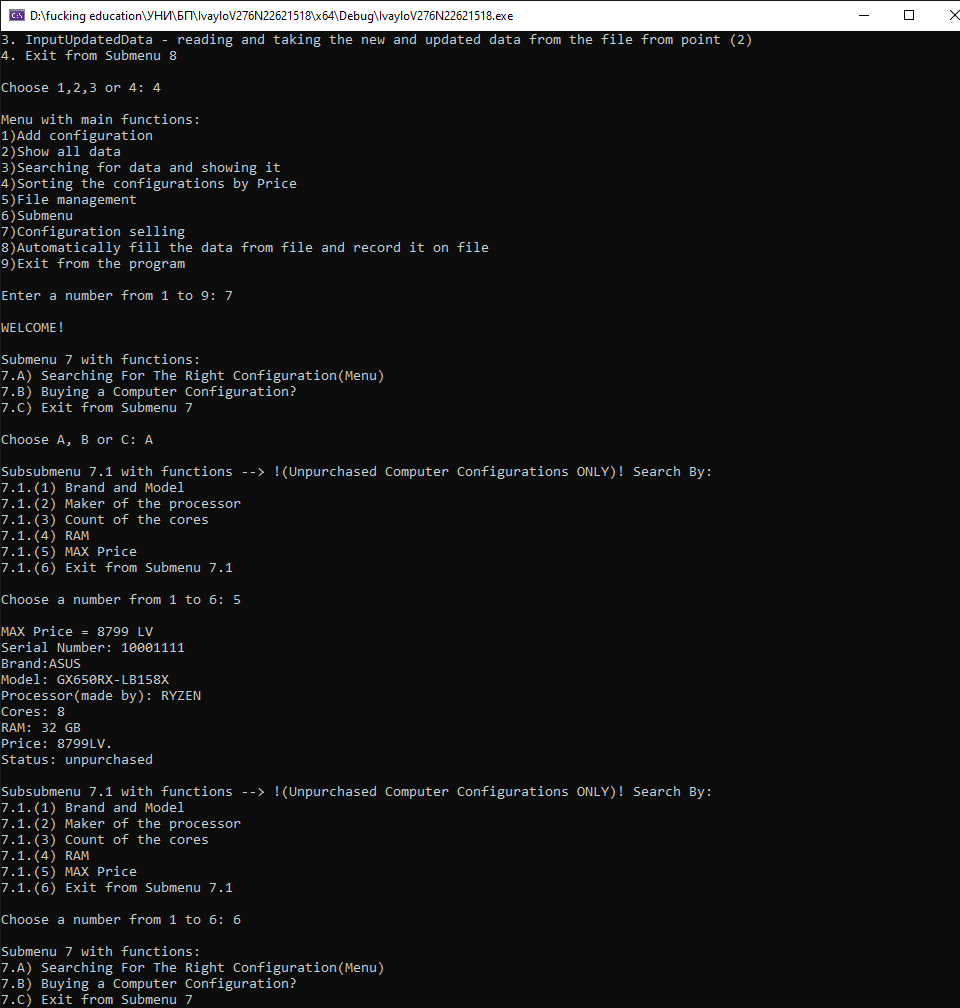
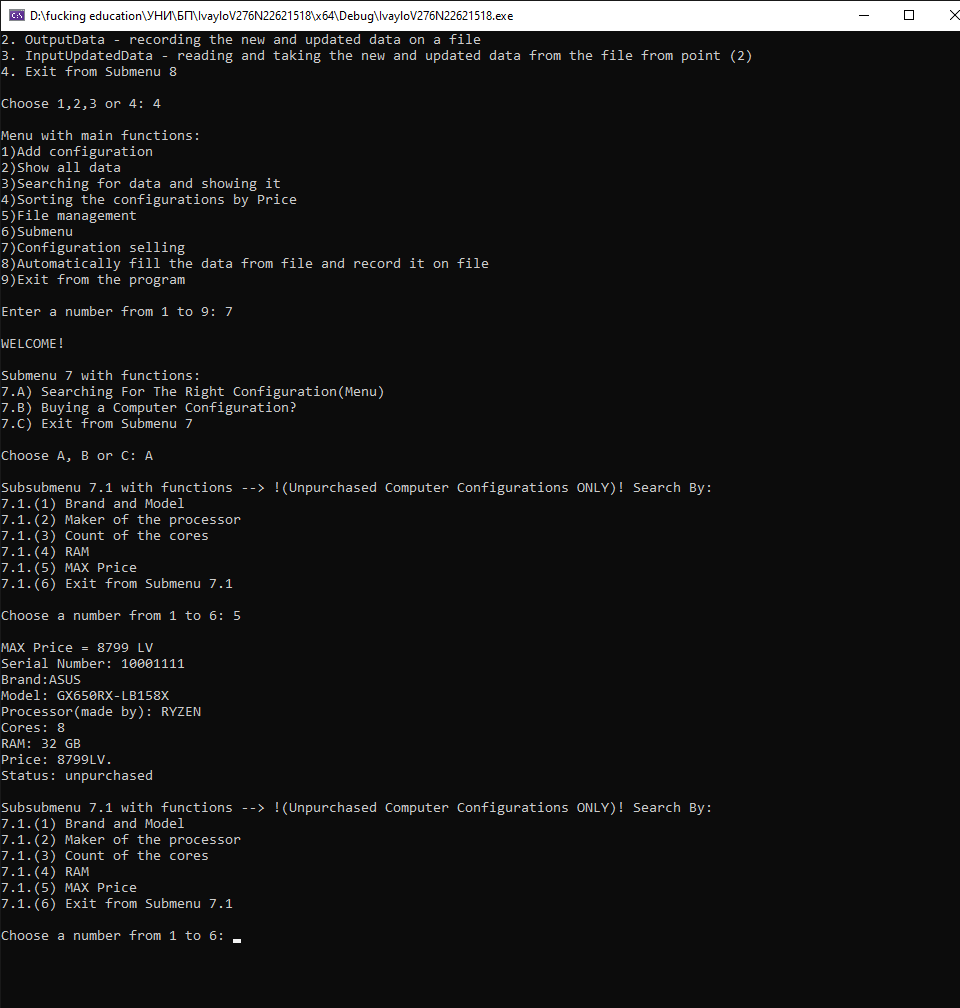
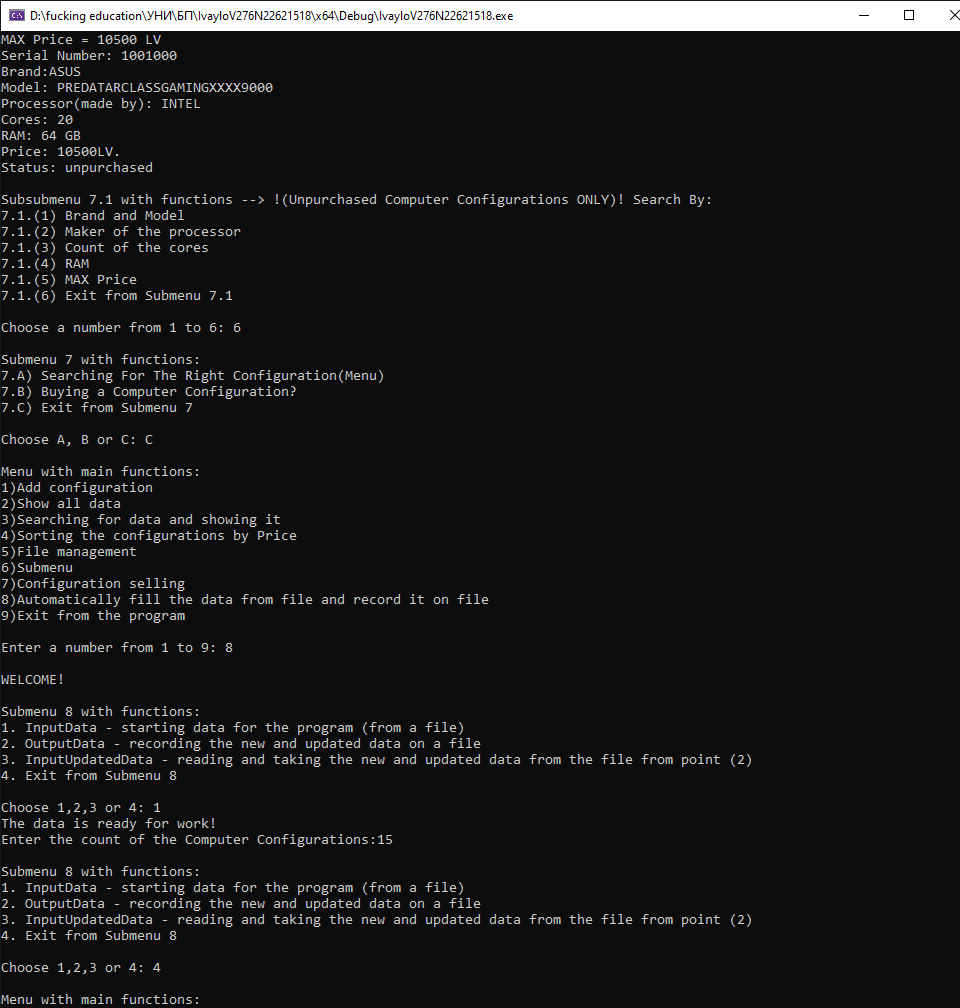
Описанието е генерирано автоматичноКартина, която съдържа текст

Описанието е генерирано автоматичноКартина, която съдържа текст

Описанието е генерирано автоматичноКартина, която съдържа текст

Описанието е генерирано автоматичноКартина, която съдържа текст

Описанието е генерирано автоматично



Допълнение трето

Снимка на изгледа с примерни входни данни

Картина, която съдържа текст, вестник, екранна снимка

Описанието е генерирано автоматично

Снимка на изгледа с примерни изходни данни

Картина, която съдържа текст

Описанието е генерирано автоматично

