**ТУ-ВАРНА**

**КУРСОВА РАБОТА**

ПО

БАЗОВО ПРОГРАМИРАНЕ – I част

Изработил: Проверил:

Михаил Бисеров Георгиев /ас. В. Колесниченко/

I курс, 1 група, VI поток, спец. СИТ

Факултетен номер: 19621820

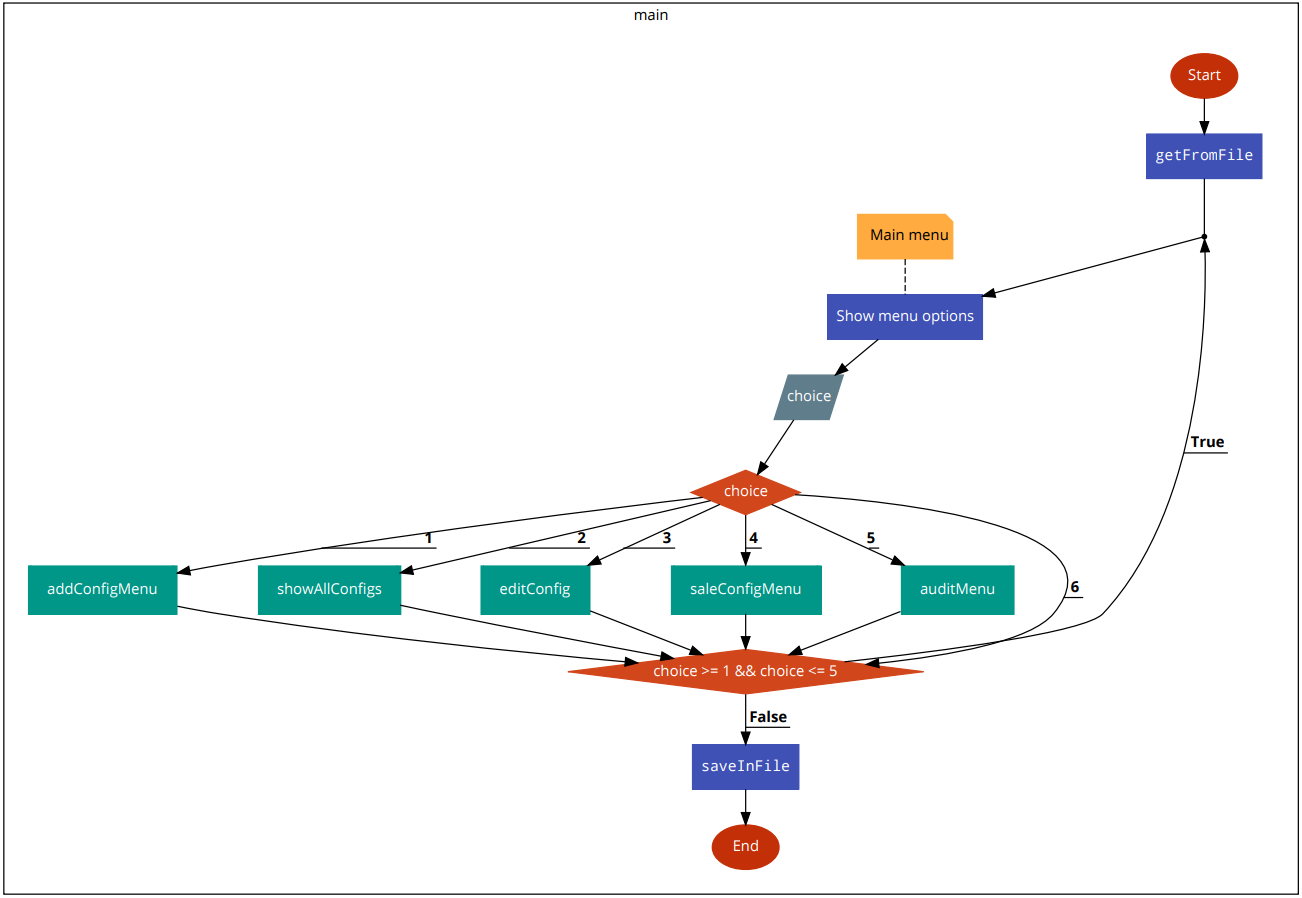
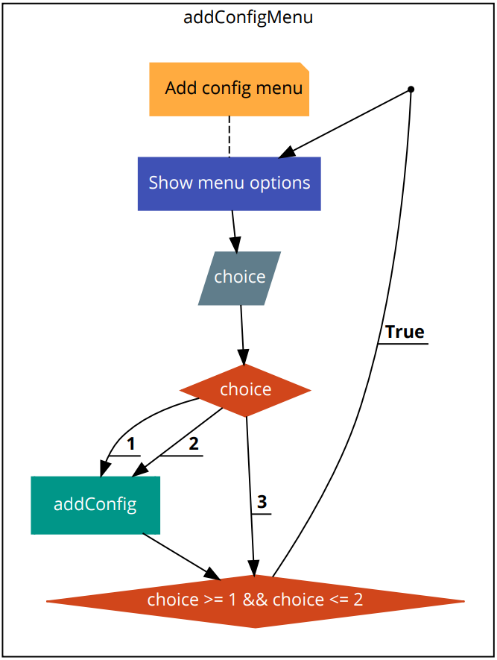
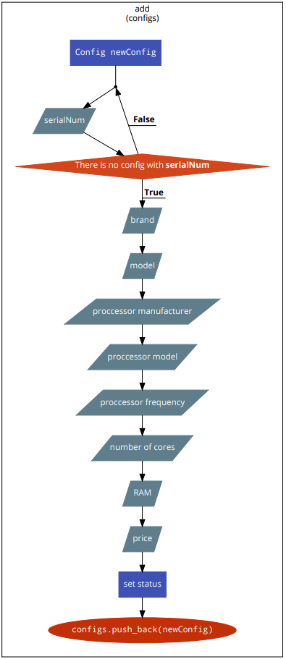
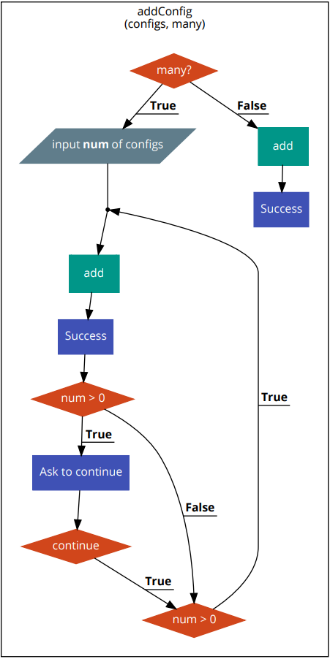
**Курсова работа – Продажба на компютри** Вариант|**184**

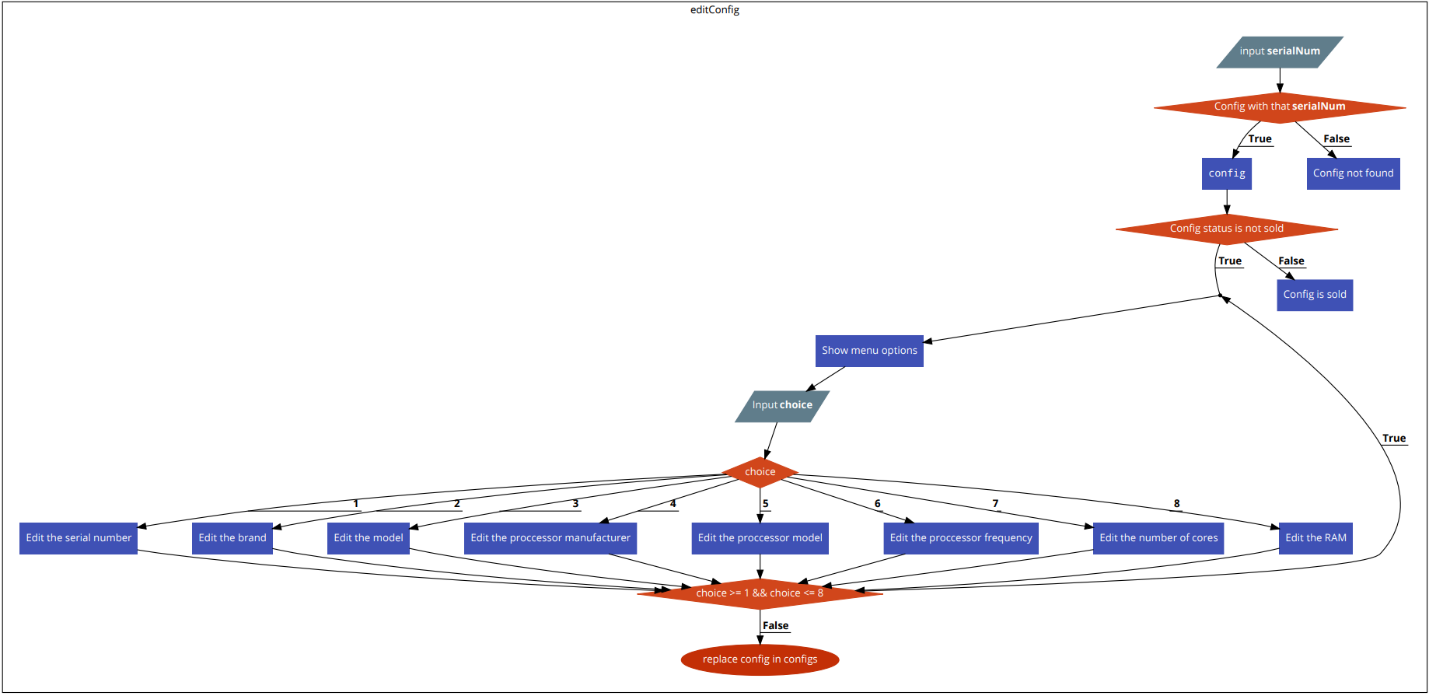
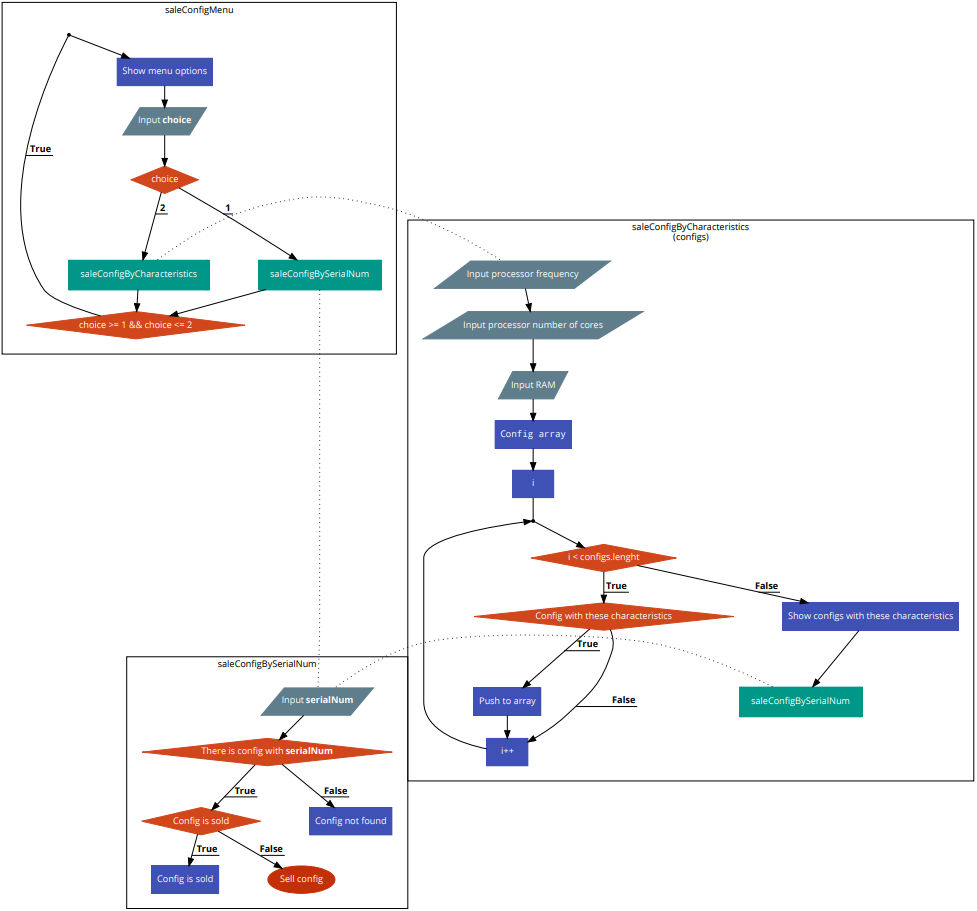
**Условие**

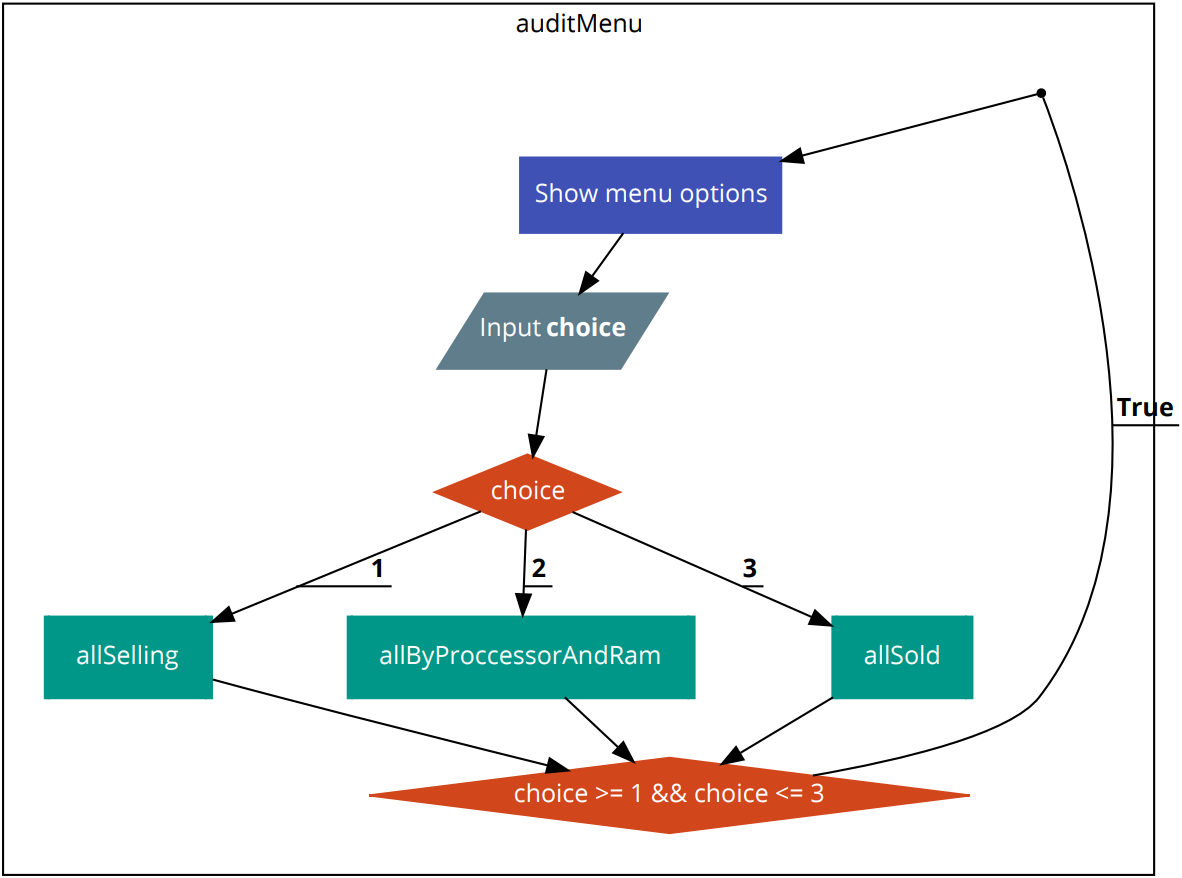
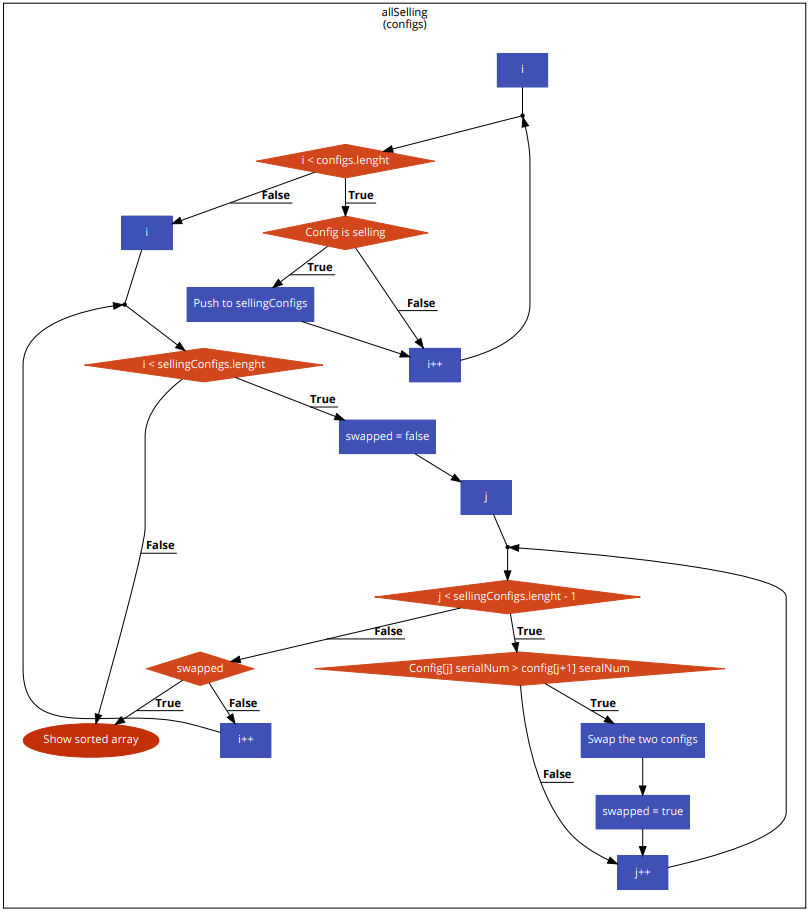
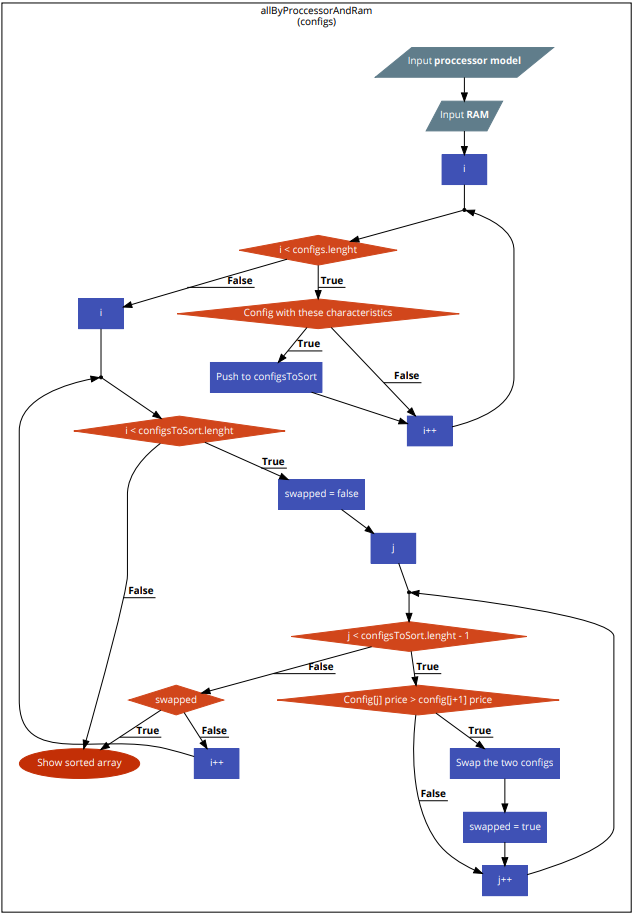
Да се напише компютърна програма, реализираща информационна система за продажба на компютърни конфигурации (сериен номер- уникален, марка, модел, процесор – производител, модел, тактова честота, брой ядра; RAM памет, цена, статус: в продажба/продадена). Максималният брой конфигурации, които могат да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

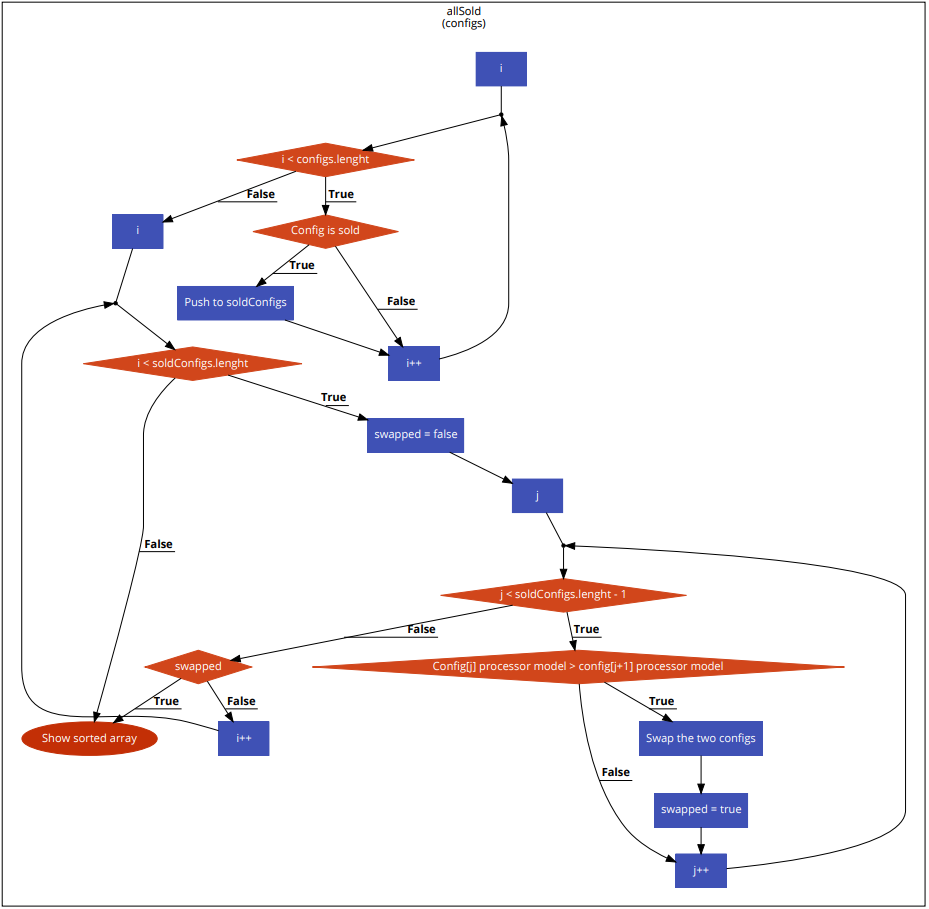
Програмата съдържа:

1. Меню с избор на функциите в програмата (7 седмица)
2. Добавяне на нови мобилни телефони (7-8 седмица)
   1. Добавяне на една нова конфигурация
   2. Добавяне на списък от конфигурации. Въвежда се цяло число n и след него n на брой нови конфигурации
3. Извеждане на всички конфигурации на екрана (8-9 седмица)
4. Корекция на данни за конфигурация
   1. Въвежда се сериен номер и данни за корекция
   2. Ако конфигурацията е продадена, не може да се прави корекция.
5. Продажба на компютърна конфигурация
   1. Въвежда се сериен номер и цената на конфигурацията.
   2. Въвеждат се характеристики и след това се избира конфигурацията за продажба
6. Одит на конфигурациите в под меню(11-12 седмица)
   1. Извеждане на всички конфигурации, които са в продажба, сортирани по сериен номер.
   2. Извеждане на всички конфигурации, с даден модел процесор и RAM памет, сортирани по цена от най-скъпия към най-евтиния.
   3. Извеждане на продадените конфигурации, сортирани по модел на процесора.
7. Данните в програмата да могат да се запазват във файл между две стартирания на програмата.
8. Допълнителни условия
   1. за точка А: при добавянето на нови компютри, ако в системата има вече въведена информация, извежда подходящо съобщение
   2. За точка B: да се реализира добавяне със запитване за нов запис и прекъсване на въвеждането.

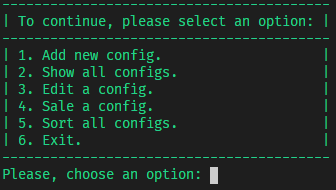
******Блок схема**





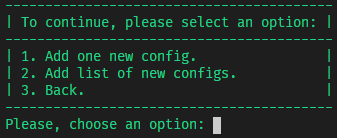


**Меню с избор на функциите**

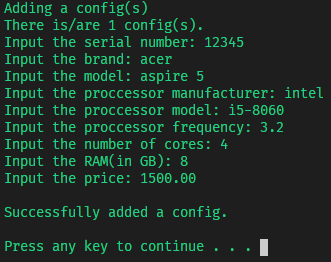


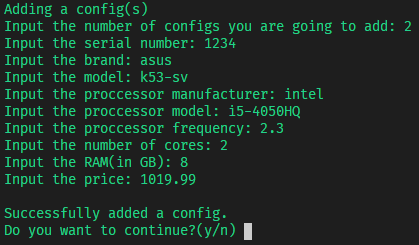
**addConfigMenu()**

С избиране на опция 1 – Добавяне на нова конфигурация програмата дава право на избор на потребителя да въведе 1 или 2 като опция, където 1 е да се добави само 1 конфигурация, а 2 е да се добавят няколко конфигурации. Функцията приема като параметри vector от типа на структурата.



**addConfig()**

С избиране на опция 1 – Добавяне на една нова конфигурация, потребителя трябва да въведе нужните данни искани от програмата. На функцията се подава bool false, за да отбележи че ще се добавя една конфигурация.

С избиране на опция 2 – Добавяне на много нова конфигурация, потребителя трябва да въведе броя на конфигурациите, които иска да въведе и след това да въведе нужните данни искани от програмата за всяка конфигурация. На функцията се подава bool true, за да отбележи че ще се добавят много конфигурация. След всяка добавена конфигурация, програмата пита потребителя дали иска да продължи да добавя.

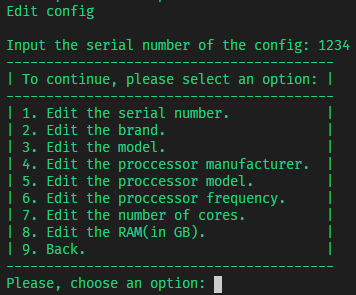
Функцията приема като параметри vector от типа на структурата и bool, за да разбиране дали ще се добавя само една или много конфигурации.

**showConfigs()**

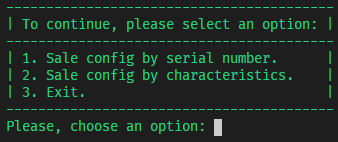
С избиране на опция 2 – Извеждане на всички конфигурация програмата показва в табличен вид всички добавени конфигурации. Функцията приема като параметър vector от типа на структурата.

**editConfig()**

С избиране на опция 3 – Редактиране на конфигурация програмата кара потребителя да въведе серийния номер на конфигурацията и след това представя опциите за редактиране под формата на меню, ако съществува такава конфигурация. Функцията приема като параметър vector от типа на структурата.



**saleConfigMenu()**

С избиране на опция 4 – Продажба на конфигурация програмата представя на потребителя меню с опции за продажба по сериен номер и продажба по характеристики. Функцията приема като параметър vector от типа на структурата.

С избиране на първата опция, програмата кара потребителя да въведе серийния номер на вече избрана от него конфигурация, с избиране на втората опция, потребителя първо трябва да въведе характеристики и след това му се показват конфигурациите които отговарят на въведените от него неща и той отново трябва да въведе серийния номер на конфигурацията.

**auditMenu()**

С избиране на опция 5 – Одит на конфигурации програмата представя на потребителя меню с опции за извеждане на всички конфигурации, които са в продажба, сортирани по сериен номер, извеждане на всички конфигурации, с даден модел процесор и RAM памет, сортирани по цена от най-скъпия към най-евтиния и за извеждане на продадените конфигурации, сортирани по модел на процесора. Функцията приема като параметър vector от типа на структурата.

**getFromFile()**

Функцията приема като параметър vector от типа на структурата и служи за записване на vector-а във двоичен файл за запазване на данните между две стартирания.

**saveInFile()**

Функцията приема като параметър vector от типа на структурата и служи за четене на vector-а от двоичния файл за запазване на данните между две стартирания.

**Source code:**

config.h

#ifndef Config\_H

#define Config\_H

#include <string>

#define configsCount 100

struct Processor

{

std::string manufacturer, model;

float frequency;

int core\_count;

};

struct Config

{

int serial\_number, ram;

std::string model, brand;

float price;

Processor processor;

enum Status

{

sold,

selling

} status;

bool operator==(const Config &right)

{

return serial\_number == right.serial\_number &&

brand == right.brand &&

model == right.model &&

price == right.price &&

ram == right.ram &&

status == right.status &&

processor.model == right.processor.model &&

processor.manufacturer == right.processor.manufacturer &&

processor.frequency == right.processor.frequency &&

processor.core\_count == right.processor.core\_count;

}

};

#endif

main.cpp

#include <iostream>

#include <vector>

#include <iomanip>

#include <fstream>

#include "main.h"

using namespace std;

int main()

{

int choice;

vector<Config> configs;

getFromFile(configs);

system("mode con: cols=132 lines=24 && color 0a");

system("printf '\e[8;24;132t' && stty rows 24 && stty cols 132");

do

{

system("clear");

system("cls");

cout << setw(41) << setfill('-') << "" << endl;

cout << "| To continue, please select an option: |\n";

cout << setw(41) << setfill('-') << "" << endl;

cout << setw(40) << setfill(' ') << left << "| 1. Add new config."

<< "|\n";

cout << setw(40) << setfill(' ') << left << "| 2. Show all configs."

<< "|\n";

cout << setw(40) << setfill(' ') << left << "| 3. Edit a config."

<< "|\n";

cout << setw(40) << setfill(' ') << left << "| 4. Sale a config."

<< "|\n";

cout << setw(40) << setfill(' ') << left << "| 5. Sort all configs."

<< "|\n";

cout << setw(40) << setfill(' ') << left << "| 6. Exit."

<< "|\n";

cout << setw(41) << setfill('-') << "" << endl;

cout << "Please, choose an option: ";

cin >> choice;

switch (choice)

{

case 1:

addConfigMenu(configs);

break;

case 2:

showConfigs(configs);

break;

case 3:

editConfig(configs);

break;

case 4:

saleConfigMenu(configs);

break;

case 5:

auditMenu(configs);

break;

case 6:

break;

}

} while (choice >= 1 && choice <= 5);

saveInFile(configs);

}

main.h

#include "config.h"

#include "add.h"

#include "show.h"

#include "edit.h"

#include "sale.h"

#include "audit.h"

#include "files.h"

add.cpp

#include <iostream>

#include <iomanip>

#include "add.h"

using namespace std;

void addConfigMenu(vector<Config> &configs)

{

int choice;

do

{

system("clear");

system("cls");

cout << setw(41) << setfill('-') << "" << endl;

cout << "| To continue, please select an option: |\n";

cout << setw(41) << setfill('-') << "" << endl;

cout << setw(40) << setfill(' ') << left << "| 1. Add one new config."

<< "|\n";

cout << setw(40) << setfill(' ') << left << "| 2. Add list of new configs."

<< "|\n";

cout << setw(40) << setfill(' ') << left << "| 3. Back."

<< "|\n";

cout << setw(41) << setfill('-') << "" << endl;

cout << "Please, choose an option: ";

cin >> choice;

switch (choice)

{

case 1:

addConfig(configs, false);

break;

case 2:

addConfig(configs, true);

break;

case 3:

break;

}

} while (choice >= 1 && choice <= 2);

}

void addConfig(vector<Config> &configs, bool many)

{

system("clear");

system("cls");

cout << "Adding a config(s)\n";

int len = configs.size();

if (len != 0)

{

cout << "There is/are " << len << " config(s).\n";

}

if (len == configsCount)

{

cout << "You can't add more than 100 configs";

}

else

{

if (many)

{

int num;

cout << "Input the number of configs you are going to add: ";

cin >> num;

do

{

char choice;

add(configs);

num--;

cout << "\nSuccessfully added a config.\n";

if (num > 0)

{

cout << "Do you want to continue?(y/n) ";

cin >> choice;

if (choice == 'n')

break;

}

} while (num > 0);

}

else

{

add(configs);

cout << "\nSuccessfully added a config.\n\n";

}

}

system("pause");

}

void add(vector<Config> &configs)

{

Config newConfig;

int serialNumber;

bool found;

do

{

cout << "Input the serial number: ";

cin >> serialNumber;

found = false;

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

{

if (configs[i].serial\_number == serialNumber)

{

found = true;

break;

}

}

if (found)

cout << "There is a config with that serial number. Try again.";

} while (found);

newConfig.serial\_number = serialNumber;

cin.ignore(1);

cout << "Input the brand: ";

getline(cin, newConfig.brand);

cout << "Input the model: ";

getline(cin, newConfig.model);

cout << "Input the proccessor manufacturer: ";

getline(cin, newConfig.processor.manufacturer);

cout << "Input the proccessor model: ";

getline(cin, newConfig.processor.model);

cout << "Input the proccessor frequency: ";

cin >> newConfig.processor.frequency;

cout << "Input the number of cores: ";

cin >> newConfig.processor.core\_count;

cout << "Input the RAM(in GB): ";

cin >> newConfig.ram;

cout << "Input the price: ";

cin >> newConfig.price;

newConfig.status = Config::Status::selling;

configs.push\_back(newConfig);

}

add.h

#include <vector>

#include "config.h"

using namespace std;

void add(vector<Config> &);

void addConfig(vector<Config> &, bool);

void addConfigMenu(vector<Config> &);

show.cpp

#include <iostream>

#include <vector>

#include <iomanip>

#include "show.h"

using namespace std;

ostream &operator<<(ostream &out, const Config::Status &x)

{

switch (x)

{

case Config::Status::selling:

out << "selling";

break;

case Config::Status::sold:

out << "sold";

break;

default:

throw out\_of\_range("Invalid value for type Config::Status");

}

return out;

}

void showConfigs(vector<Config> configs)

{

int len = configs.size();

cout << setw(131) << setfill('-') << "" << endl; // Line 1

cout << "|" << setw(15) << setfill(' ') << ""

<< "|"

<< setw(10) << setfill(' ') << ""

<< "|"

<< setw(10) << setfill(' ') << ""

<< "|"

<< setw(23) << setfill(' ') << ""

<< "Proccessor" << setw(22) << setfill(' ') << ""

<< "|" << setw(13) << setfill(' ') << ""

<< "|"

<< setw(10) << setfill(' ') << ""

<< "|"

<< setw(10) << setfill(' ') << ""

<< "|" << endl; // Line 2

cout << setw(16) << setfill(' ') << "| Serial number"

<< "|"

<< setw(10) << setfill(' ') << " Brand"

<< "|"

<< setw(10) << setfill(' ') << " Model"

<< "|"

<< setw(55) << setfill('-') << "" << setw(14) << setfill(' ') << "| RAM (in GB)"

<< "|"

<< setw(10) << setfill(' ') << " Price"

<< "|"

<< setw(10) << setfill(' ') << " Status"

<< "|" << endl; //Line 3

cout << "|" << setw(15) << setfill(' ') << ""

<< "|"

<< setw(10) << setfill(' ') << ""

<< "|"

<< setw(10) << setfill(' ') << ""

<< "|"

<< setw(15) << setfill(' ') << " Manufacturer"

<< "|"

<< setw(10) << setfill(' ') << " Model"

<< "|"

<< setw(20) << setfill(' ') << " Frequency (in GHz)"

<< "|"

<< setw(7) << setfill(' ') << " Cores"

<< "|" << setw(13) << setfill(' ') << ""

<< "|"

<< setw(10) << setfill(' ') << ""

<< "|"

<< setw(10) << setfill(' ') << ""

<< "|" << endl; // Line 4

if (len == 0)

{

cout << setw(131) << setfill('-') << "" << endl;

cout << "|" << setw(51) << setfill(' ') << ""

<< "There are no configs added." << setw(51) << setfill(' ') << ""

<< "|" << endl;

}

else

{

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

{

cout << setw(131) << setfill('-') << "" << endl;

cout << "| " << setw(13) << setfill(' ') << left << configs[i].serial\_number;

cout << " | " << setw(8) << setfill(' ') << left << configs[i].brand;

cout << " | " << setw(8) << setfill(' ') << left << configs[i].model;

cout << " | " << setw(13) << setfill(' ') << left << configs[i].processor.manufacturer;

cout << " | " << setw(8) << setfill(' ') << left << configs[i].processor.model;

cout << " | " << setw(18) << setfill(' ') << left << setprecision(1) << fixed << configs[i].processor.frequency;

cout << " | " << setw(5) << setfill(' ') << left << configs[i].processor.core\_count;

cout << " | " << setw(11) << setfill(' ') << left << configs[i].ram;

cout << " | " << setw(8) << setfill(' ') << left << setprecision(2) << fixed << configs[i].price;

cout << " | " << setw(8) << setfill(' ') << left << configs[i].status << " |\n";

}

}

cout << setw(131) << setfill('-') << "" << endl;

system("pause");

}

show.h

#include "config.h"

#include <vector>

using namespace std;

void showConfigs(vector<Config> configs);

edit.cpp

#include <iostream>

#include <vector>

#include <iomanip>

#include <algorithm>

#include "edit.h"

using namespace std;

void editConfig(vector<Config> &configs)

{

cout << "Edit config\n\n";

int serialNum, index(-1);

cout << "Input the serial number of the config: ";

cin >> serialNum;

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

{

if (configs[i].serial\_number == serialNum)

{

index = i;

break;

}

}

if (index != -1)

{

Config config = configs[index];

if (!config.status)

{

cout << "The config is sold.\n";

system("pause");

}

else

{

int choice;

do

{

cout << setw(41) << setfill('-') << "" << endl;

cout << "| To continue, please select an option: |\n";

cout << setw(41) << setfill('-') << "" << endl;

cout << setw(40) << setfill(' ') << left << "| 1. Edit the serial number."

<< "|\n";

cout << setw(40) << setfill(' ') << left << "| 2. Edit the brand."

<< "|\n";

cout << setw(40) << setfill(' ') << left << "| 3. Edit the model."

<< "|\n";

cout << setw(40) << setfill(' ') << left << "| 4. Edit the proccessor manufacturer."

<< "|\n";

cout << setw(40) << setfill(' ') << left << "| 5. Edit the proccessor model."

<< "|\n";

cout << setw(40) << setfill(' ') << left << "| 6. Edit the proccessor frequency."

<< "|\n";

cout << setw(40) << setfill(' ') << left << "| 7. Edit the number of cores."

<< "|\n";

cout << setw(40) << setfill(' ') << left << "| 8. Edit the RAM(in GB)."

<< "|\n";

cout << setw(40) << setfill(' ') << left << "| 9. Back."

<< "|\n";

cout << setw(41) << setfill('-') << "" << endl;

cout << "Please, choose an option: ";

cin >> choice;

switch (choice)

{

case 1:

int serialNumber;

bool found;

do

{

cout << "Input the new serial number: ";

cin >> serialNumber;

found = false;

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

{

if (configs[i].serial\_number == serialNumber)

{

found = true;

break;

}

}

if (found)

cout << "There is a config with that serial number. Try again.";

} while (found);

config.serial\_number = serialNumber;

break;

case 2:

cout << "Input the new brand: ";

cin >> config.brand;

break;

case 3:

cout << "Input the new model: ";

cin >> config.model;

break;

case 4:

cout << "Input the new proccessor manufacturer: ";

cin >> config.processor.manufacturer;

break;

case 5:

cout << "Input the new proccessor model: ";

cin >> config.processor.model;

break;

case 6:

cout << "Input the new proccessor frequency: ";

cin >> config.processor.frequency;

break;

case 7:

cout << "Input the new number of cores: ";

cin >> config.processor.core\_count;

break;

case 8:

cout << "Input the new RAM(in GB): ";

cin >> config.ram;

break;

}

} while (choice >= 1 && choice <= 8);

configs[index] = config;

replace(configs.begin(), configs.end(), configs[index], config);

}

}

else

{

cout << "No config found with that serial number.\n\n";

system("pause");

}

}

edit.h

#include <vector>

using namespace std;

#include "config.h"

void editConfig(vector<Config> &);

sale.cpp

#include <iostream>

#include <iomanip>

#include <vector>

#include <algorithm>

#include "sale.h"

#include "show.h"

using namespace std;

void saleConfigMenu(vector<Config> &configs)

{

int choice;

do

{

cout << setw(41) << setfill('-') << "" << endl;

cout << "| To continue, please select an option: |\n";

cout << setw(41) << setfill('-') << "" << endl;

cout << setw(40) << setfill(' ') << left << "| 1. Sale config by serial number."

<< "|\n";

cout << setw(40) << setfill(' ') << left << "| 2. Sale config by characteristics."

<< "|\n";

cout << setw(40) << setfill(' ') << left << "| 3. Exit."

<< "|\n";

cout << setw(41) << setfill('-') << "" << endl;

cout << "Please, choose an option: ";

cin >> choice;

switch (choice)

{

case 1:

cout << "Sale config by serial number\n\n";

saleConfigBySerialNum(configs);

break;

case 2:

cout << "Sale config by processor frequency and cores and RAM.\n\n";

saleConfigByCharacteristics(configs);

break;

case 3:

break;

}

} while (choice >= 1 && choice <= 2);

}

void saleConfigBySerialNum(vector<Config> &configs)

{

int serialNum, index(-1);

cout << "Input the serial number of the config: ";

cin >> serialNum;

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

{

if (configs[i].serial\_number == serialNum)

{

index = i;

break;

}

}

if (index != -1)

{

Config config = configs[index];

if (!config.status)

{

cout << "The config is sold.";

}

else

{

config.status = Config::Status::sold;

replace(configs.begin(), configs.end(), configs[index], config);

cout << "The config was sold successfuly.";

}

}

}

void saleConfigByCharacteristics(vector<Config> &configs)

{

int cores, ram, index(-1), serialNum;

float frequency;

vector<Config> sortedConfigs;

cout << "Input the processor frequency: ";

cin >> frequency;

cout << "Input the processor number of cores: ";

cin >> cores;

cout << "Input the RAM(in GB): ";

cin >> ram;

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

{

if (configs[i].processor.frequency == frequency && configs[i].processor.core\_count == cores && configs[i].ram == ram)

{

sortedConfigs.push\_back(configs[i]);

}

}

showConfigs(sortedConfigs);

saleConfigBySerialNum(configs);

}

sale.h

#include <vector>

#include "config.h"

using namespace std;

void saleConfigMenu(vector<Config> &);

void saleConfigBySerialNum(vector<Config> &);

void saleConfigByCharacteristics(vector<Config> &);

audit.cpp

#include <iostream>

#include <iomanip>

#include "audit.h"

using namespace std;

void auditMenu(vector<Config> &configs)

{

int choice;

do

{

system("clear");

system("cls");

cout << setw(58) << setfill('-') << "" << endl;

cout << "| To continue, please select an option: |\n";

cout << setw(58) << setfill('-') << "" << endl;

cout << setw(57) << setfill(' ') << left << "| 1. Sort all curently selling configs by serial number."

<< "|\n";

cout << setw(57) << setfill(' ') << left << "| 2. Sort all configs by proccessor model and RAM."

<< "|\n";

cout << setw(57) << setfill(' ') << left << "| 3. Sort all sold configs by proccessor model."

<< "|\n";

cout << setw(57) << setfill(' ') << left << "| 4. Back."

<< "|\n";

cout << setw(58) << setfill('-') << "" << endl;

cout << "Please, choose an option: ";

cin >> choice;

switch (choice)

{

case 1:

allSelling(configs);

break;

case 2:

allByProccessorAndRam(configs);

break;

case 3:

allSold(configs);

break;

case 4:

break;

}

} while (choice >= 1 && choice <= 3);

}

void allSelling(vector<Config> &configs)

{

vector<Config> sellingConfigs;

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

{

if (configs[i].status)

{

sellingConfigs.push\_back(configs[i]);

}

}

Config temp;

bool swapped;

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

{

swapped = false;

for (int j = 0; j < sellingConfigs.size() - 1; j++)

{

if (sellingConfigs[j].serial\_number > sellingConfigs[j + 1].serial\_number)

{

temp = sellingConfigs[j];

sellingConfigs[j] = sellingConfigs[j + 1];

sellingConfigs[j + 1] = temp;

swapped = true;

}

}

if (!swapped)

break;

}

showConfigs(sellingConfigs);

}

void allByProccessorAndRam(vector<Config> &configs)

{

vector<Config> configsToSort;

string model;

int ram;

cout << "Input the proccessor model: ";

cin >> model;

cout << "Input the RAM: ";

cin >> ram;

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

{

if (configs[i].processor.model == model && configs[i].ram == ram)

{

configsToSort.push\_back(configs[i]);

}

}

Config temp;

bool swapped;

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

{

swapped = false;

for (int j = 0; j < configsToSort.size() - 1; j++)

{

if (configsToSort[j].price < configsToSort[j + 1].price)

{

temp = configsToSort[j];

configsToSort[j] = configsToSort[j + 1];

configsToSort[j + 1] = temp;

swapped = true;

}

}

if (!swapped)

break;

}

showConfigs(configsToSort);

}

void allSold(vector<Config> &configs)

{

vector<Config> soldConfigs;

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

{

if (!configs[i].status)

{

soldConfigs.push\_back(configs[i]);

}

}

Config temp;

bool swapped;

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

{

swapped = false;

for (int j = 0; j < soldConfigs.size() - 1; j++)

{

if (soldConfigs[j].processor.model.compare(soldConfigs[j + 1].processor.model) > 0)

{

temp = soldConfigs[j];

soldConfigs[j] = soldConfigs[j + 1];

soldConfigs[j + 1] = temp;

swapped = true;

}

}

if (!swapped)

break;

}

showConfigs(soldConfigs);

}

audit.h

#include <vector>

#include "config.h"

#include "show.h"

using namespace std;

void auditMenu(vector<Config> &);

void allSelling(vector<Config> &);

void allByProccessorAndRam(vector<Config> &);

void allSold(vector<Config> &);

files.cpp

#include <fstream>

#include <vector>

#include "files.h"

using namespace std;

void saveInFile(vector<Config> &configs)

{

fstream file;

int size = configs.size();

file.open("configs.bin", ios::out | ios::binary);

file.write((char \*)&size, sizeof(int));

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

{

int len;

file.write((char \*)&configs[i].serial\_number, sizeof(int));

file.write((char \*)&configs[i].ram, sizeof(int));

len = configs[i].model.size();

file.write((char \*)&len, sizeof(int));

file.write((char \*)&configs[i].model[0], len);

len = configs[i].brand.size();

file.write((char \*)&len, sizeof(int));

file.write((char \*)&configs[i].brand[0], len);

file.write((char \*)&configs[i].price, sizeof(float));

len = configs[i].processor.manufacturer.size();

file.write((char \*)&len, sizeof(int));

file.write((char \*)&configs[i].processor.manufacturer[0], len);

len = configs[i].processor.model.size();

file.write((char \*)&len, sizeof(int));

file.write((char \*)&configs[i].processor.model[0], len);

file.write((char \*)&configs[i].processor.frequency, sizeof(float));

file.write((char \*)&configs[i].processor.core\_count, sizeof(int));

file.write((char \*)&configs[i].status, sizeof(Config::Status));

}

file.close();

}

void getFromFile(vector<Config> &configs)

{

fstream file;

int size = configs.size();

file.open("configs.bin", ios::in | ios::binary);

file.read((char \*)&size, sizeof(int));

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

{

int len;

Config config;

file.read((char \*)&config.serial\_number, sizeof(int));

file.read((char \*)&config.ram, sizeof(int));

file.read((char \*)&len, sizeof(int));

config.model.resize(len);

file.read((char \*)&config.model[0], len);

file.read((char \*)&len, sizeof(int));

config.brand.resize(len);

file.read((char \*)&config.brand[0], len);

file.read((char \*)&config.price, sizeof(float));

file.read((char \*)&len, sizeof(int));

config.processor.manufacturer.resize(len);

file.read((char \*)&config.processor.manufacturer[0], len);

file.read((char \*)&len, sizeof(int));

config.processor.model.resize(len);

file.read((char \*)&config.processor.model[0], len);

file.read((char \*)&config.processor.frequency, sizeof(float));

file.read((char \*)&config.processor.core\_count, sizeof(int));

file.read((char \*)&config.status, sizeof(Config::Status));

configs.push\_back(config);

}

file.close();

}

files.h

#include <vector>

#include "config.h"

using namespace std;

void saveInFile(vector<Config> &);

void getFromFile(vector<Config> &);