

ТЕХНИЧЕСКИ УНИВЕРСИТЕТ ВАРНА

**Факултет по изчислителна техника и
автоматизация**

Софтуерни и интернет технологии

**Семестриална домашна работа: Продажба
на компютри**

Изготвил: Явор Йорданов Чамов

№: 21621577

21621577 ЯВОР ЙОРДАНОВ ЧАМОВ

Продажба на компютри

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.

Базова задача – сложност ниска

- A. Меню с избор на функциите в програмата (7 седмица)
- B. Добавяне на нови компютърни конфигурации (7-8 седмица)
 - a. Добавяне на една нова конфигурация
 - b. Добавяне на списък от конфигурации. Въвежда се цяло число **n** и след него **n** на брой нови конфигурации
- C. Извеждане на всички конфигурации на екрана (8-9 седмица)
 - a. Извеждане на конфигурациите с най-голяма тактова честота на процесора
 - b. Извеждане на конфигурациите от дадена марка

Допълнение първо – сложност средна (+ базова задача)

- D. Корекция на данни за конфигурация
 - a. Въвежда се сериен номер и данни за корекция
 - b. Ако конфигурацията е продадена, не може да се прави корекция.

Допълнение второ – сложност висока (+ базова задача + допълнение първо)

- E. Продажба на компютърна конфигурация
 - a. Въвежда се сериен номер и цената на конфигурацията.
 - b. Въвеждат се характеристики и след това се избира конфигурацията за продажба

Допълнение трето – сложност висока (+ базова задача + допълнение първо + допълнение второ)

- F. Одит на конфигурациите в под меню(11-12 седмица)
 - a. Извеждане на всички конфигурации, които са в продажба, сортирани по сериен номер.
 - b. Извеждане на всички конфигурации с даден модел процесор и RAM памет, сортирани по цена от най-скъпия към най-евтиния.
 - c. Извеждане на продадените конфигурации, сортирани по модел на процесора.
- G. Данните в програмата да могат да се запазват във файл между две стартирания на програмата.

Допълнение четвърто – (за допълнителни точки)

- H. Допълнителни условия:
 - a. за точка A: при добавянето на нови компютри, ако в системата има вече въведена информация, извежда подходящо съобщение
 - b. За точка B: да се реализира добавяне със запитване за нов запис и прекъсване на въвеждането.

Анализ на решението

Структури

Компютърните конфигурации в програмата са представени чрез структурите Processor и Computer.

```
struct Processor
{
    string manufacturer;
    string model;
    double frequency;
    int cores;
};

struct Computer
{
    string id;
    string brand;
    string model;
    Processor processor;
    double ram;
    double price;
    string available_status;
};
```

Съхранение на информация

По време на изпълнение на програмата данните за съществуващите конфигурации се запазват в масив от тип структура Computer. Масивът **configurations** е деклариран в main методът с максимален брой от 100 елемента. Заедно с него е декларирана и променливата **present_configurations_count** с начална стойност 0. Тази променлива пази актуалният брой на запазените конфигурации.

След приключването на изпълнение на програмата, всички съществуващи конфигурации се записват във файл **configurations.dat**. При повторно стартиране на програмата конфигурациите **автоматично** се прочитат от файла и запазват в масива **configurations**.

Начин на реализиране на всяка част от условието

A. Меню с избор на функциите в програмата

```
void read_valid_integer_value(int& value);
int main();
```

Функцията *read_valid_integer_value* приема като входен параметър адреса на променлива от тип int и гарантира, че потребителят е въвел валидна числена стойност. Реализирана е с безкраен while цикъл, който прекратява своето изпълнение при въведена валидна числена стойност. В случай на невалидни входни данни на екрана се извежда съобщение, което приканва потребителят да въведе валидна числена стойност. Стойността се запазва в адреса на подаденият параметър value.

Менюто с избор на функциите в програмата е реализирано в *main* функцията. За имплементацията му е използван do-while цикъл. На всяка итерация на екрана се извежда съобщение с всички възможни действия на програмта и клавишът, който трябва да се натисне, за да се изпълнят. След това се извежда съобщение, което подканва потребителят да въведе своя избор. Въвеждането на избор е реализирано с функцията *read_valid_integer_value*.

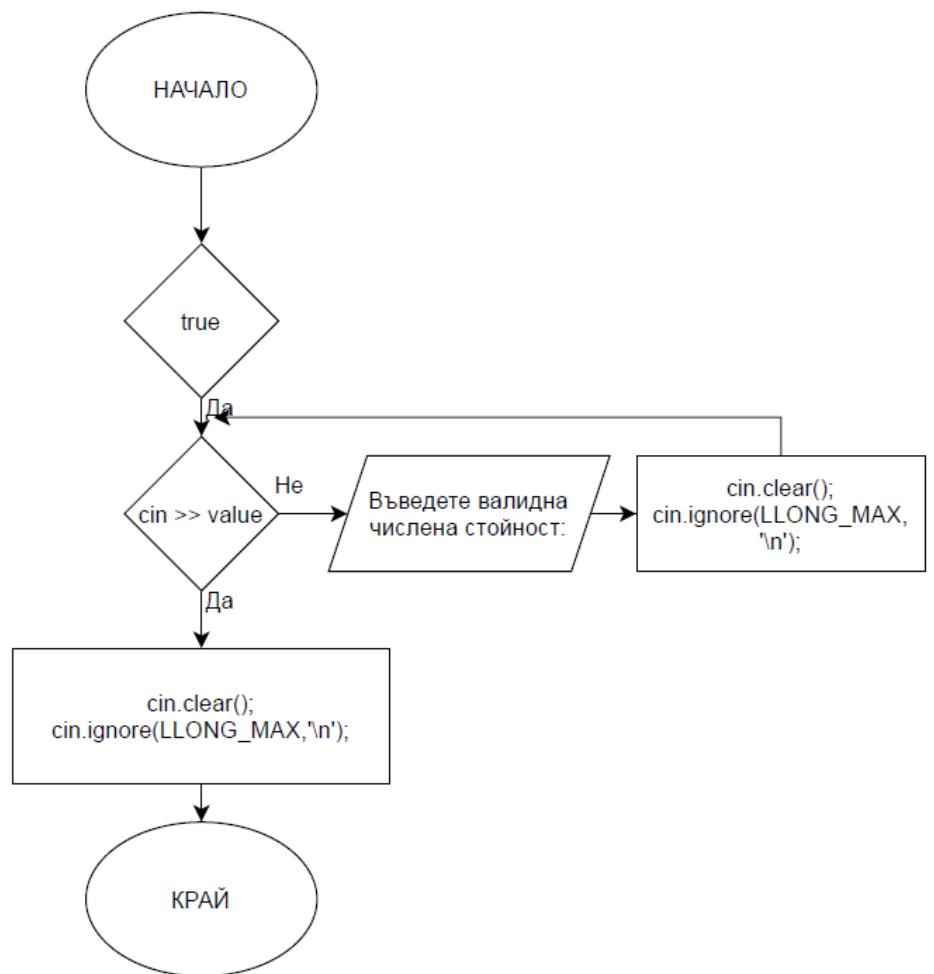
Валидирането на меню избор е реализирано с do-while цикъл, който прекратява изпълнението си при въведен избор в посочения интервал [(първа опция в менюто) – (последна опция в менюто)].

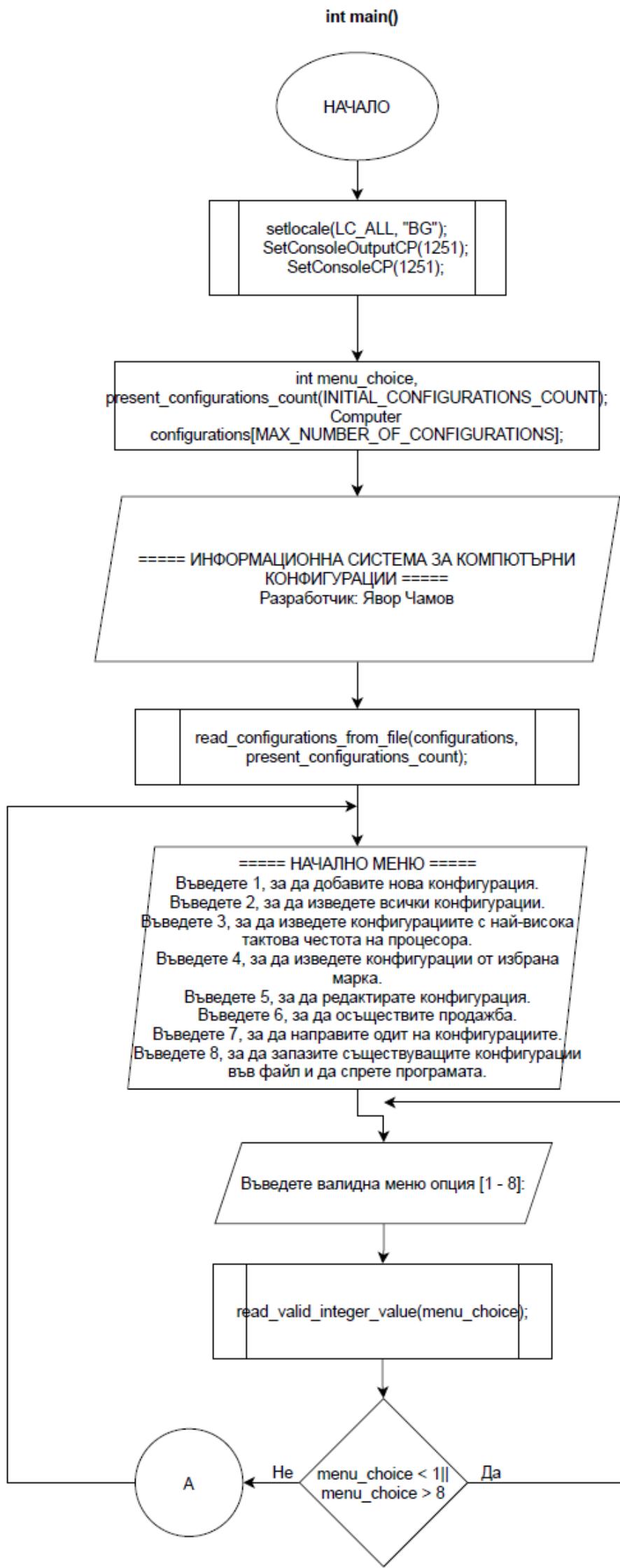
След правилно въведена и валидна меню опция програмата зачиства текста от екрана и чрез конструкцията switch се задейства функционалността, отговаряща на посочената меню опция.

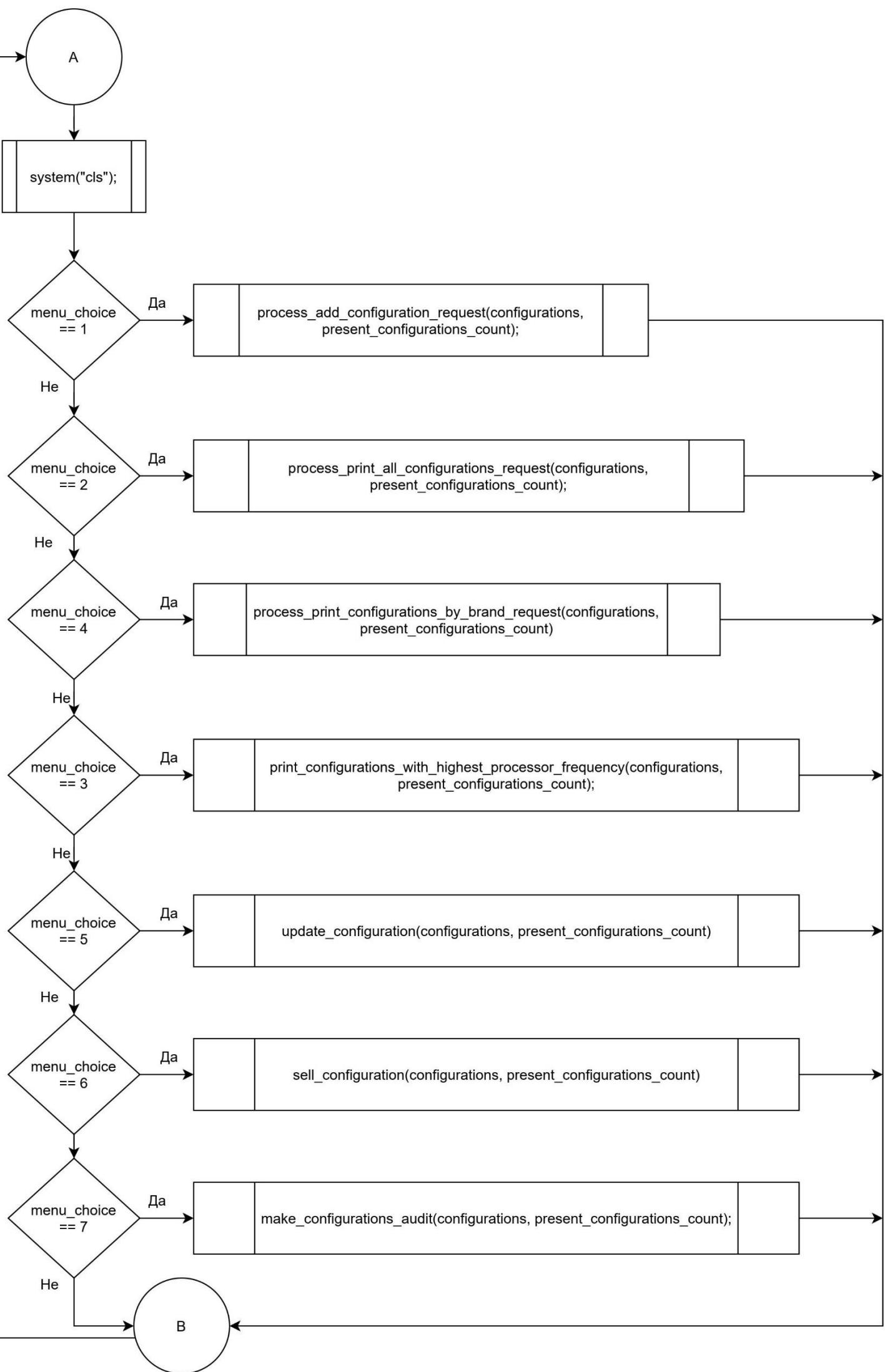
След като приключи изпълнението на избраната функционалност се извежда празен ред в конзолата и отново се визуализира началното меню.

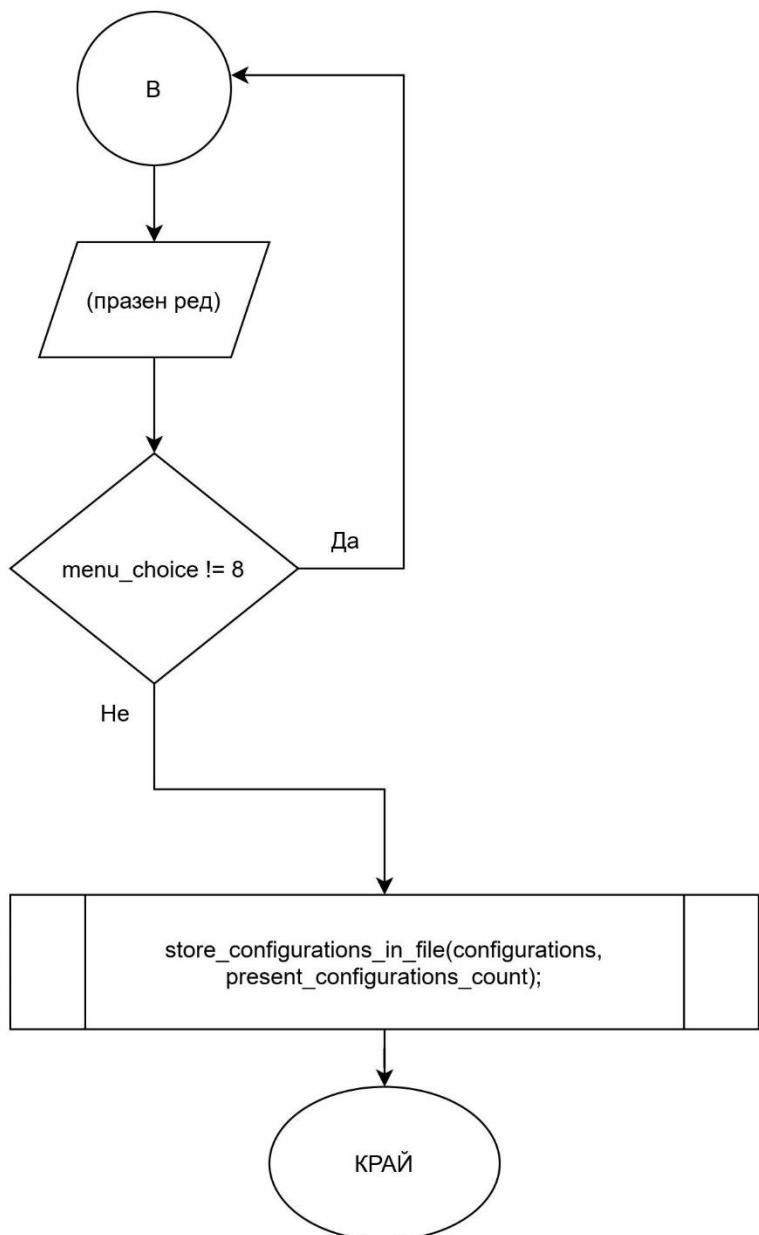
Ако потребителят е избрал последната меню опция, програмата прекратява своето изпълнение.

```
void read_valid_integer_value(int& value);
```









Екранни снимки с примерни входни и изходни данни

a. Първоначален изглед

```
===== ИНФОРМАЦИОННА СИСТЕМА ЗА ПРОДАЖБА НА КОМПЮТЪРНИ КОНФИГУРАЦИИ =====
Разработчик: Явор Чамов

16 конфигурации бяха прочетени от файл.

===== НАЧАЛНО МЕНЮ =====

Въведете 1, за да добавите нова конфигурация.
Въведете 2, за да изведете всички конфигурации.
Въведете 3, за да изведете конфигурациите с най-висока тактова честота на процесора.
Въведете 4, за да изведете конфигурации от избрана марка.
Въведете 5, за да редактирате конфигурация.
Въведете 6, за да осъществите продажба.
Въведете 7, за да направите одит на конфигурациите.
Въведете 8, за да запазите съществуващите конфигурации във файл и да спрете програмата.
Въведете валидна меню опция [1 - 8]:
```

b. Примерен изглед след валидна меню опция

```
===== ДОБАВЯНЕ НА НОВИ КОНФИГУРАЦИИ =====
```

```
Въведете брой конфигурации за добавяне:
```

c. Изглед след невалидна меню опция

```
===== НАЧАЛНО МЕНЮ =====

Въведете 1, за да добавите нова конфигурация.
Въведете 2, за да изведете всички конфигурации.
Въведете 3, за да изведете конфигурациите с най-висока тактова честота на процесора.
Въведете 4, за да изведете конфигурации от избрана марка.
Въведете 5, за да редактирате конфигурация.
Въведете 6, за да осъществите продажба.
Въведете 7, за да направите одит на конфигурациите.
Въведете 8, за да запазите съществуващите конфигурации във файл и да спрете програмата.
Въведете валидна меню опция [1 - 8]: -1
Въведете валидна меню опция [1 - 8]: 9
Въведете валидна меню опция [1 - 8]: а
Въведете валидна числена стойност: b
Въведете валидна числена стойност:
```

В. Добавяне на нови компютърни конфигурации

- a. добавяне на една нова конфигураци
- b. добавяне на списък от конфигурации. Въвежда се цяло число n и след него n на брой нови конфигурации

Н. Допълнителни условия:

- a. при добавянето на нови компютри, ако в системата има вече въведена информация, извежда подходящо съобщение
- b. да се реализира добавяне със запитване за нов запис и прекъсване на въвеждането

```
void read_valid_integer_value(int& value);

bool is_configurations_count_valid(int configurations_count, int
present_configurations_count);

bool is_string_empty(string& input);

void read_valid_double_value(double& value);

void read_processor_data(string& manufacturer, string& model, double& frequency, int&
cores);

bool configuration_exists_by_id(string& id, Computer configurations[], int
present_configurations_count);

void read_computer_data(string& id, string& brand, string& model, double& ram, double&
price, string& available_status, Computer configurations[], int
present_configurations_count);

void add_configuration(Computer configurations[], int& present_configurations_count,
string& processor_manufacturer,
string& processor_model, string& computer_id, string& computer_brand, string&
computer_model, string& available_status,
double& processor_frequency, double& computer_ram, double& computer_price, int&
processor_cores);

void process_add_configurations_request(Computer configurations[], int&
present_configurations_count);
```

Функцията *is_configurations_count_valid* връща булев резултат. Приема като входни параметри желаният брой конфигурации за добавяне и броят на текущите конфигурации. Ако броят на желаните конфигурации за добавяне е по-голям от 0 и има място за тяхното съхранение, резултатът от функцията е *true*. В противен случай – *false*.

Функцията *is_string_empty* връща булев резултат. Приема като параметът променлива от тип *string*. Ако резултатът от функцията *empty* на класа *string* е *true*, функцията връща *true*. В противен случай се *string*-ът се обхожда и проверява дали съществува символ различен от интервал и таб. Ако такъв символ съществува функцията връща *false*. Ако не е намерен символ различен от споменатите, функцията връща *true*.

Функцията *read_valid_double_value* приема като входен параметър адреса на променлива от тип *double* и гарантира, че потребителят е въвел валидна числена стойност. Реализирана е с безкраен while цикъл, който прекратява своето изпълнение при въведена валидна числена

стойност. В случай на невалидни входни данни на екрана се извежда съобщение, което приканва потребителят да въведе валидна числена стойност. Стойността се запазва в адреса на подаденият параметър value.

Функцията *read_processor_data* приема като параметри променливи (подадени по адрес), в които ще бъдат записани данните за всеки елемент на процесора. Посредством do-while цикъл и функцията *is_string_empty* на потребителят не му се позволява да въведе празна стойност за променливите от тип string. Прочитането на стойност за честотата на процесора се извършва чрез функцията *read_valid_double_value*. Стойностите за честота на процесора и броят ядра не могат да бъдат по-малки от 0. Валидацията е направена чрез while цикъл.

Функцията *configuration_exists_by_id* връща булев резултат. Приема като входни параметри серийния номер на конфигурацията, масив с всички запазени конфигурации и броят на записаните конфигурации. Итерира през масива и за всяка конфигурация проверява дали нейният сериен номер е равен на подадения като параметър. Ако това условие е вярно, функцията връща *true*, в противен случай – *false*.

Функцията *read_computer_data* приема като параметри променливи (подадени по адрес), в които ще бъдат записани данните за всеки елемент на компютъра. Посредством do-while цикъл и функцията *is_string_empty* на потребителят не му се позволява да въведе празна стойност за променливите от тип string. Допълнително при въвеждането на сериен номер (*id*) се прави проверка дали вече съществува конфигурация с този сериен номер чрез *configuration_exists_by_id*. Ако съществува, потребителят трябва да въведе нов сериен номер на конфигурацията. Прочитането на стойност за RAM и цена се извършва чрез функцията *read_valid_double_value*. Стойностите за RAM и цена не могат да бъдат по-малки от 0. Валидацията е направена чрез while цикъл. Наличният статус на конфигурацията не може да бъде различен от „в продажба/продадена“. Валидацията е направена чрез do-while цикъл.

Функцията *add_configuration* има за цел да добави една конфигурация към масива, който съдържа всички запазени конфигурации.

Като входни параметри приема:

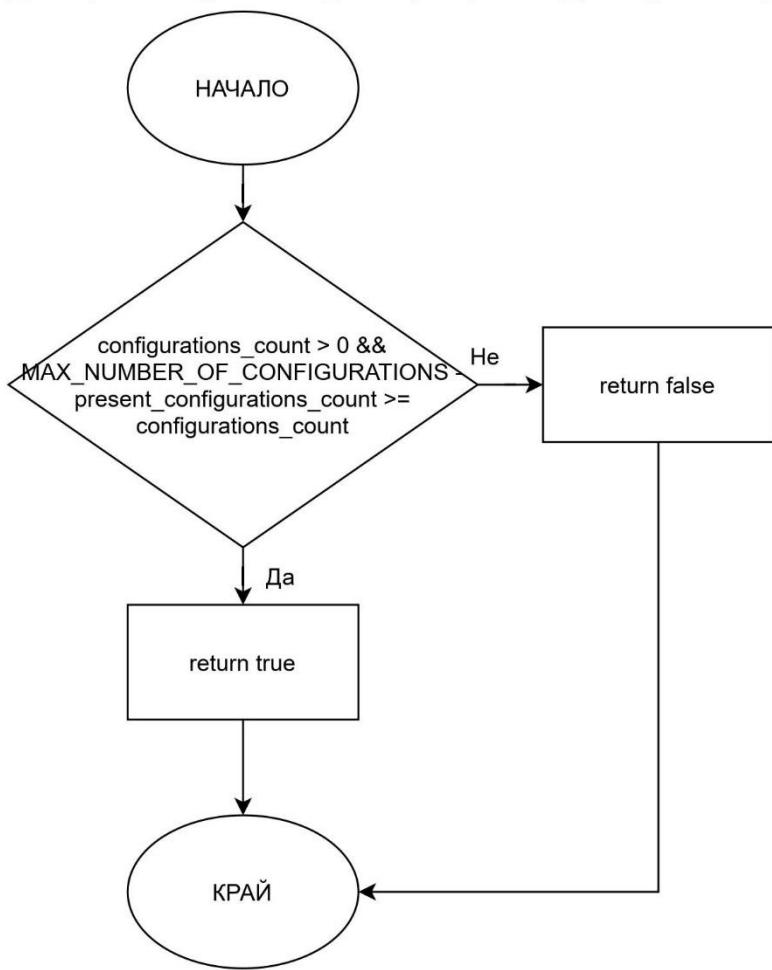
- масив с всички запазени конфигурации
- брой запазени конфигурации
- променливи (подадени по адрес), в които ще бъдат записани данните за всеки елемент на конфигурацията

Чрез функциите *read_processor_data* и *read_computer_data* се присвояват стойности на подадените по адрес променливи за данните на конфигурацията. След това се създават променливи от тип Processor и Computer със съответните стойности. Променливата от тип Computer се добавя към масива с конфигурации. Броят на конфигурациите в масива се увеличава с 1. След успешно добавяне се извежда съобщение на екрана.

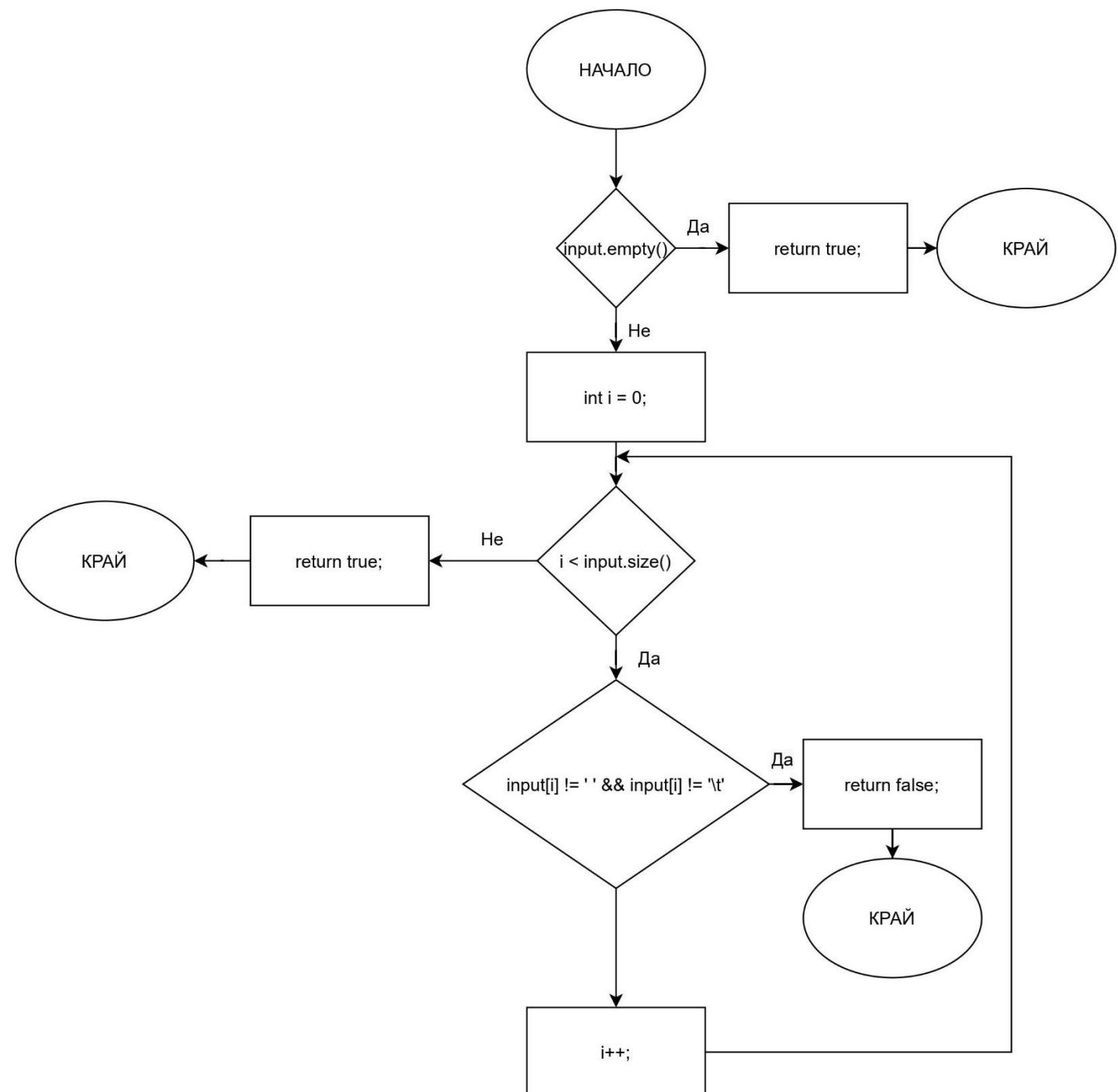
Функционалността за добавяне на нови компютърни конфигурации е реализирана във функцията *process_add_configurations_request*. Функцията *process_add_configurations_request* приема като входни параметри масив с всички запазени конфигурации и брой запазени конфигурации. При нейното извикване се извежда текст показващ името на избраната функционалност, както и текст приканващ потребителя да въведе брой конфигурации за добавяне. Прочитането на броя конфигурации е осъществено чрез функцията *read_valid_integer_value*. След това се извършва валидация на въведенния брой конфигурации чрез функцията *is_configurations_count_valid*. В случай, че броят на желаните конфигурации за добавяне не е валиден, на екрана се извежда подходящо съобщение. Ако броят на свободните места за конфигурации е по-малък от желаният за добавяне, на екрана се извежда съобщение указващо максималният брой конфигурации, които могат да бъдат добавени към момента. Ако

броят на свободните места за конфигурации не е по-малък от желаният за добавяне, на екрана се извежда съобщение показващо интервала, към който трябва да принадлеци броят на конфигурациите за добавяне. При невалиден брой конфигурации за добавяне, функцията приключва изпълнението си. При валиден брой конфигурации за добавяне се декларират променливи, в които ще се запише информация за всяка част от конфигурацията. Посредством цикъл се итерира през броя конфигурации за добавяне. Във всяка итерация се извика функцията *add_configuration*. Ако цикълът не е достигнал до последната конфигурация на екрана се извежда съобщение, което пита потребителят дали иска да продължи с добавянето на конфигурации. За да продължи с добавянето на конфигурации трябва да въведе *y/Y*, а за да прекрати добавянето на конфигурации *n/N*. Входът на избора е валидиран чрез while цикъл. При въвеждане на */N* се извежда съобщение и функцията спира своето изпълнение.

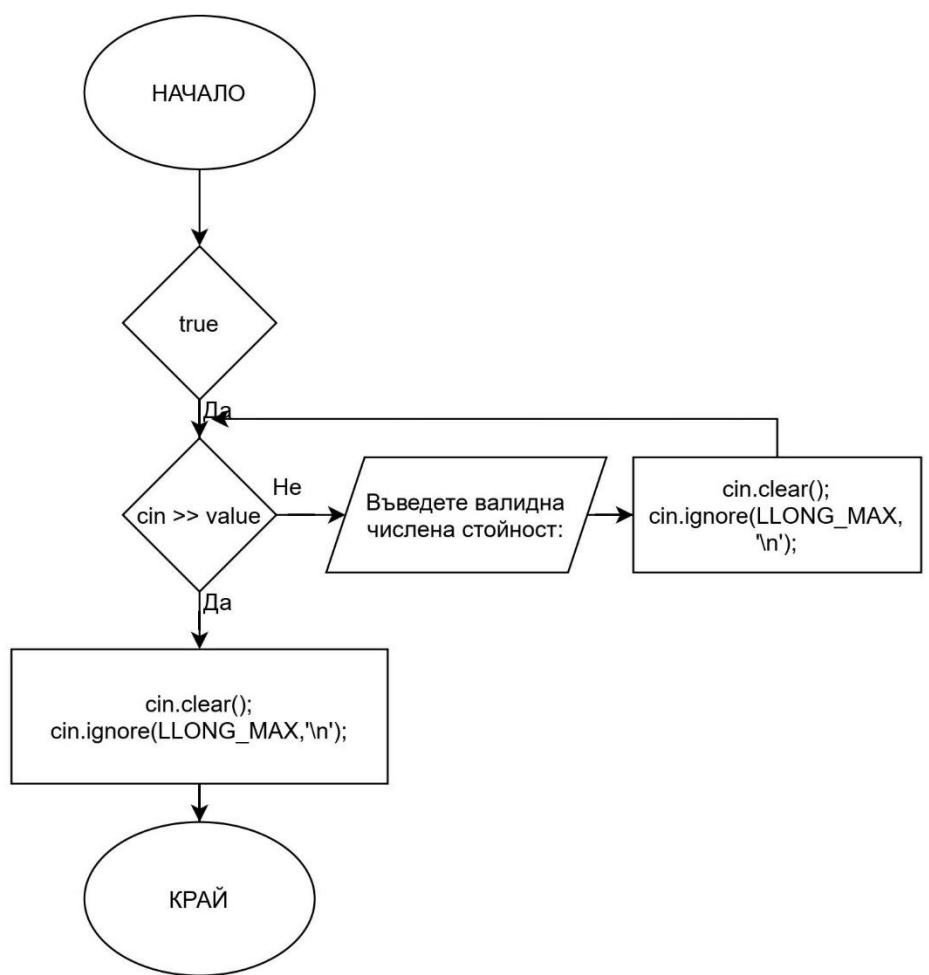
```
bool is_configurations_count_valid(int configurations_count, int present_configurations_count);
```



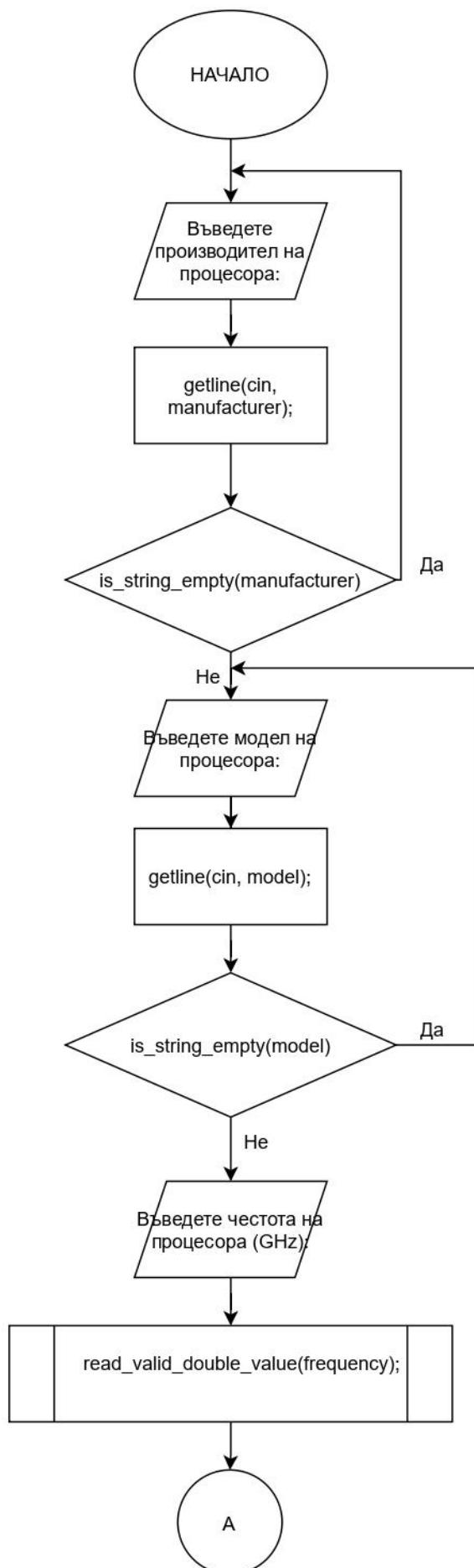
```
bool is_string_empty(string& input);
```

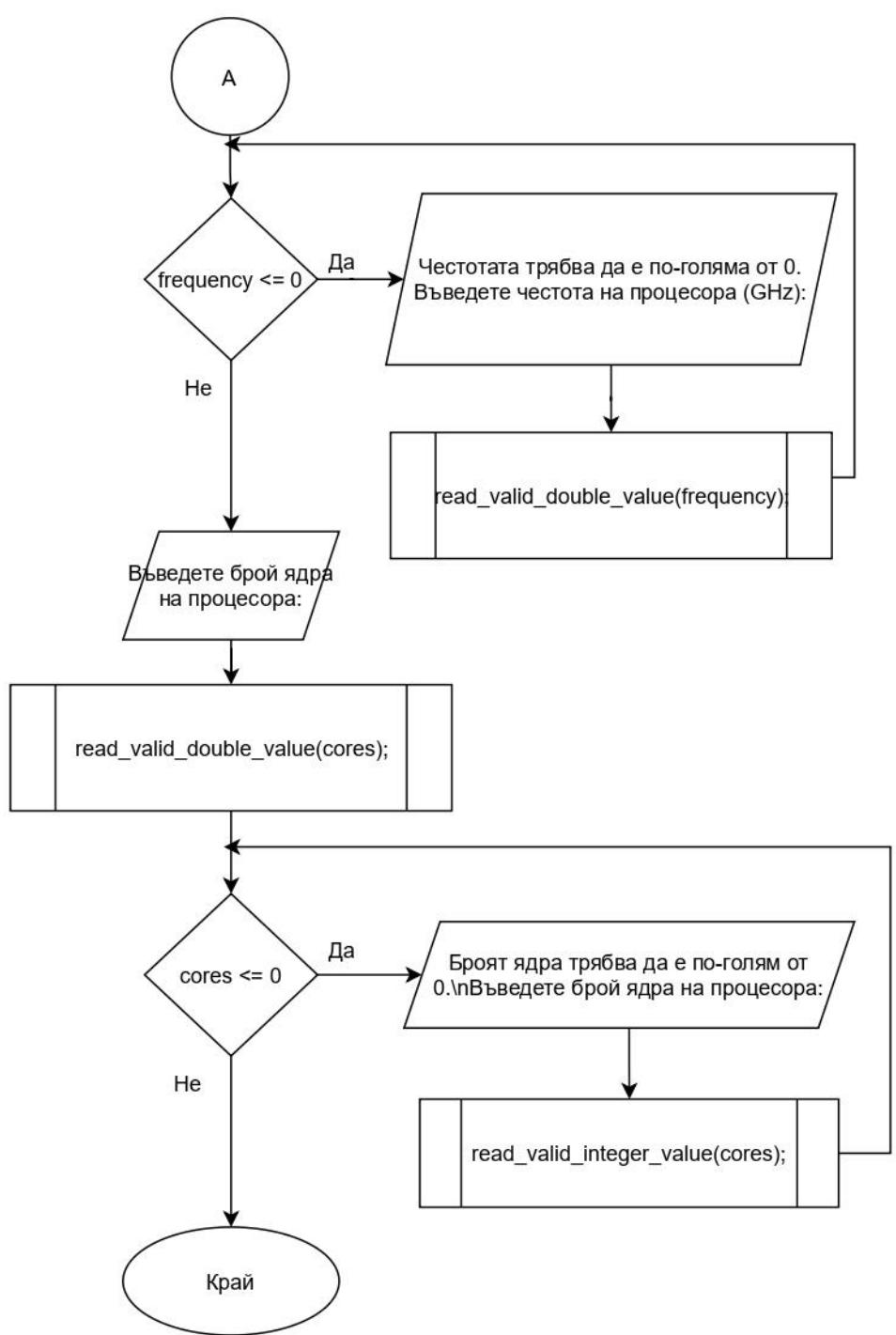


```
void read_valid_double_value(double& value);
```

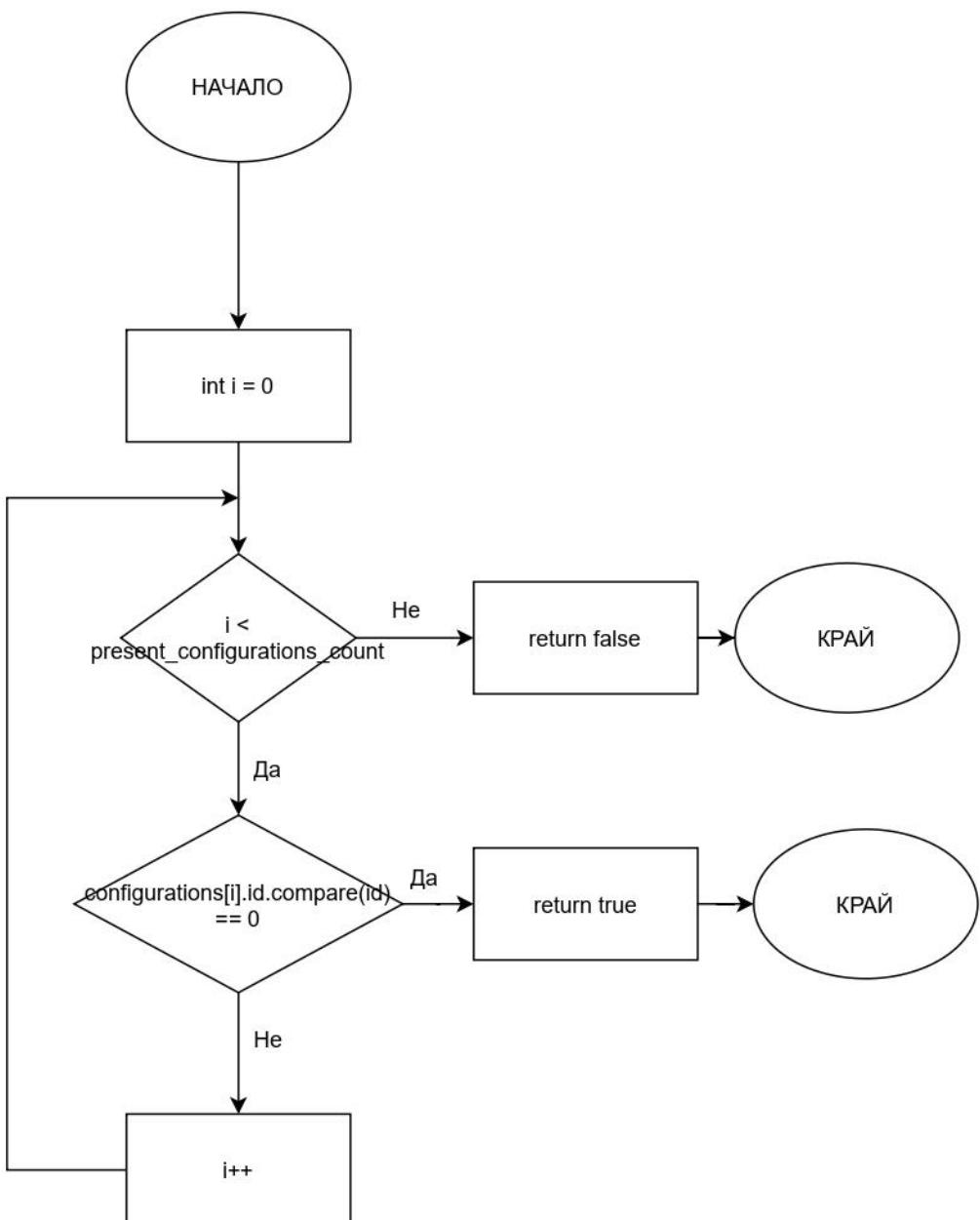


```
void read_processor_data(string& manufacturer, string& model, double& frequency, int& cores);
```

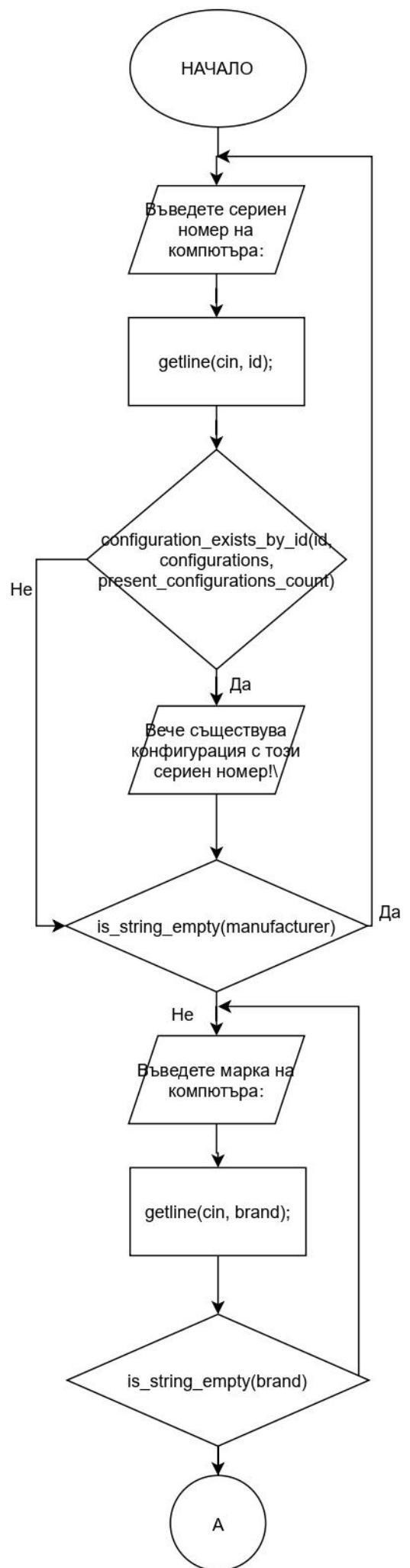


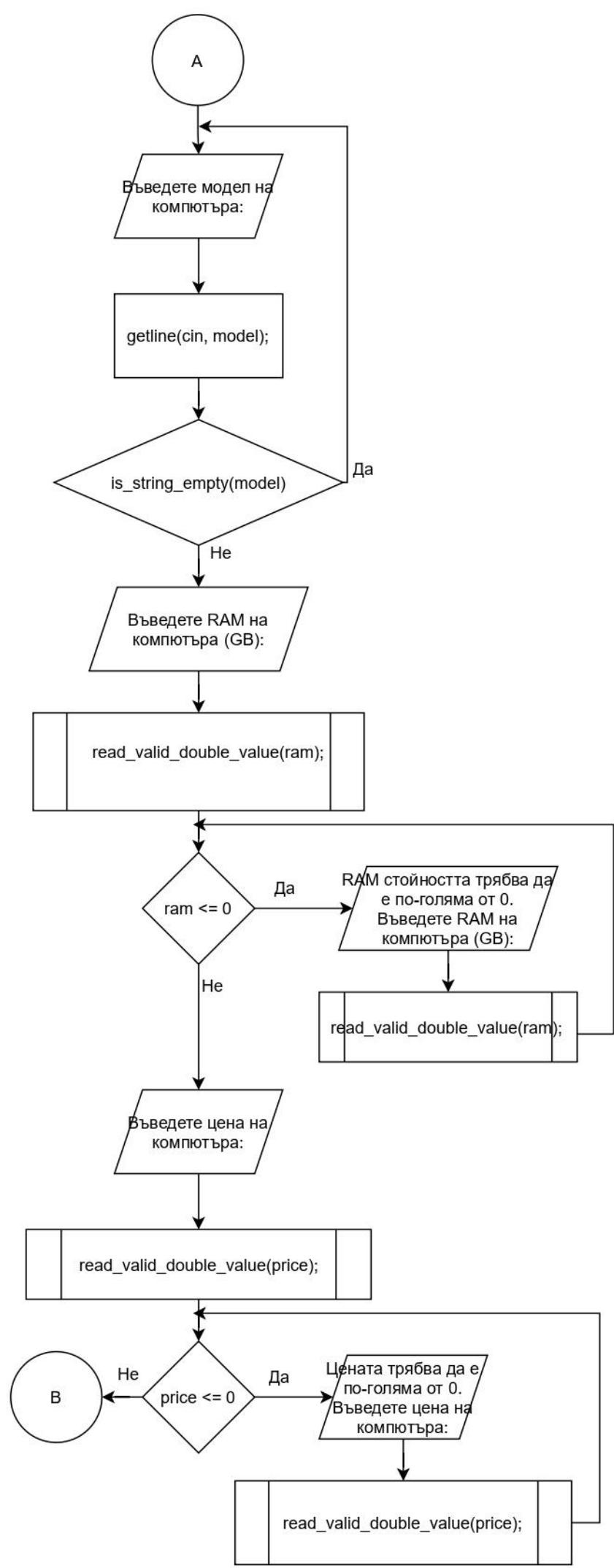


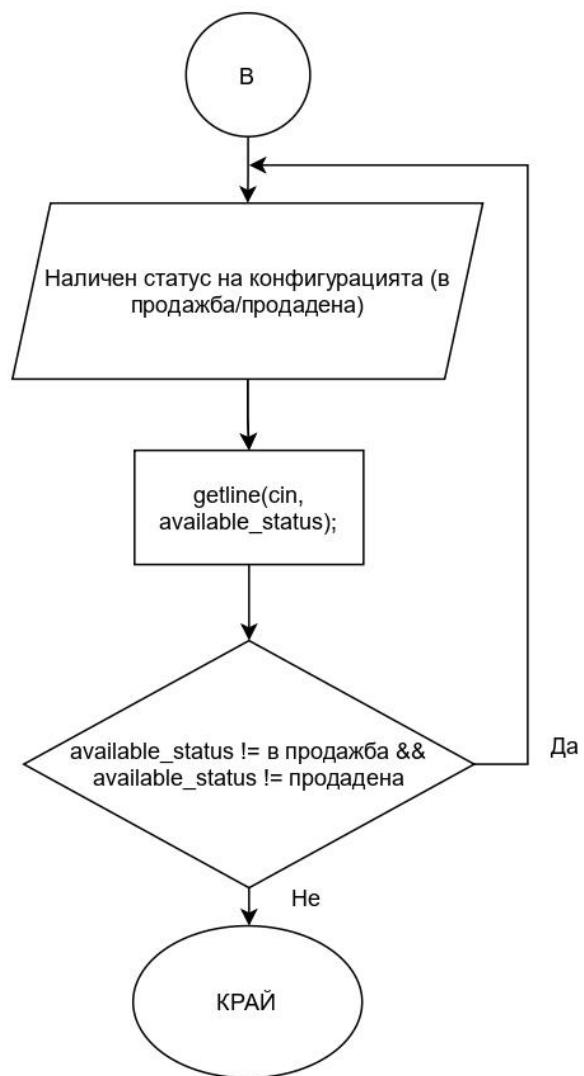
```
bool configuration_exists_by_id(string& id, Computer configurations[], int present_configurations_count);
```



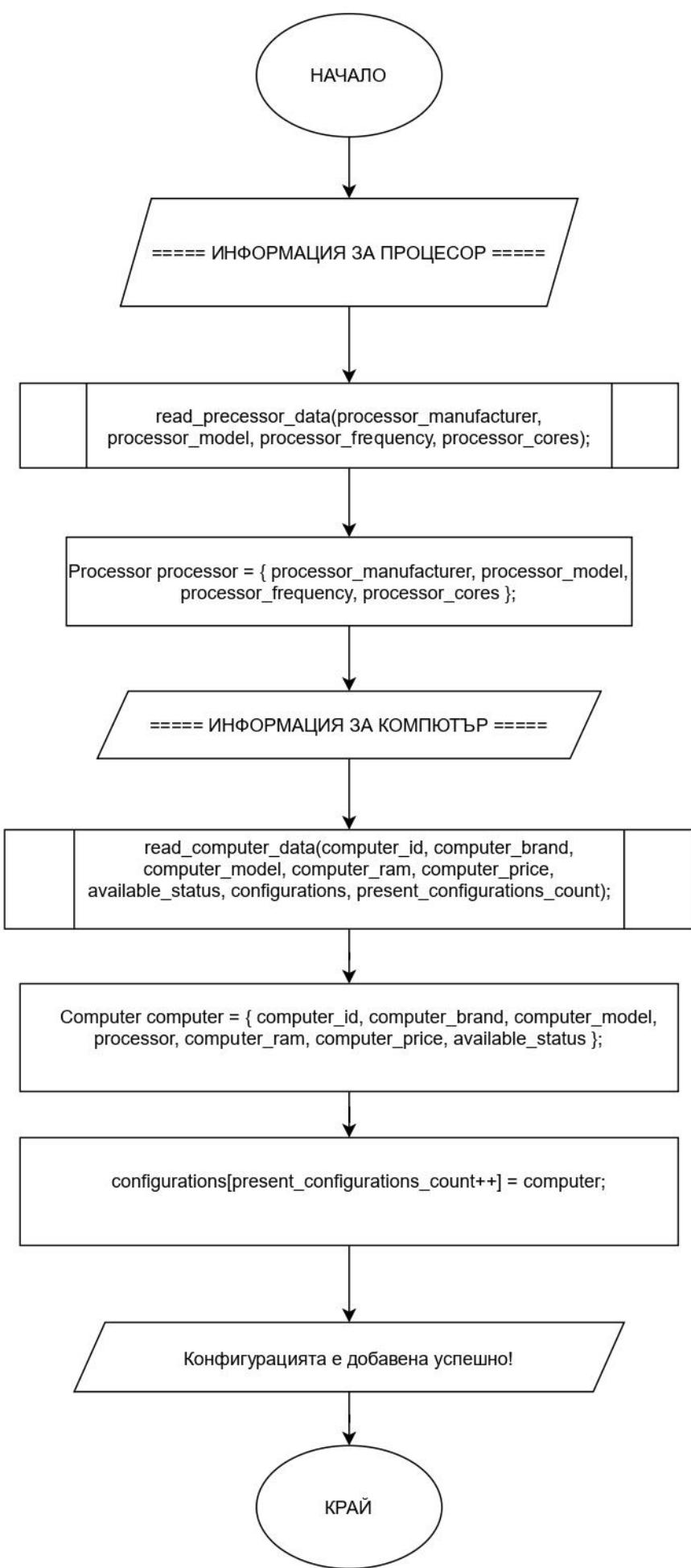
```
void read_computer_data(string& id, string& brand, string& model, double& ram, double& price, string& available_status, Computer configurations[], int present_configurations_count);
```



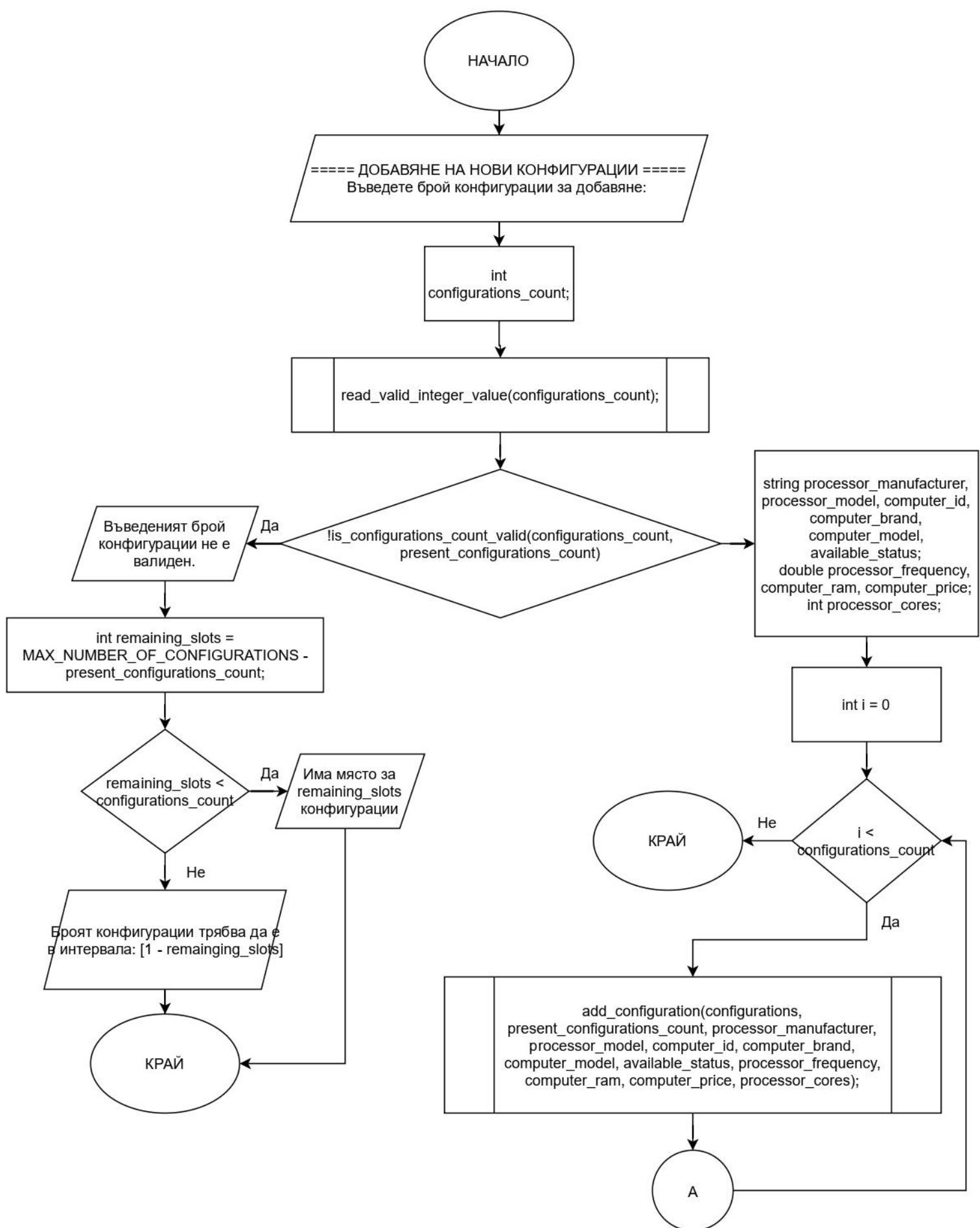


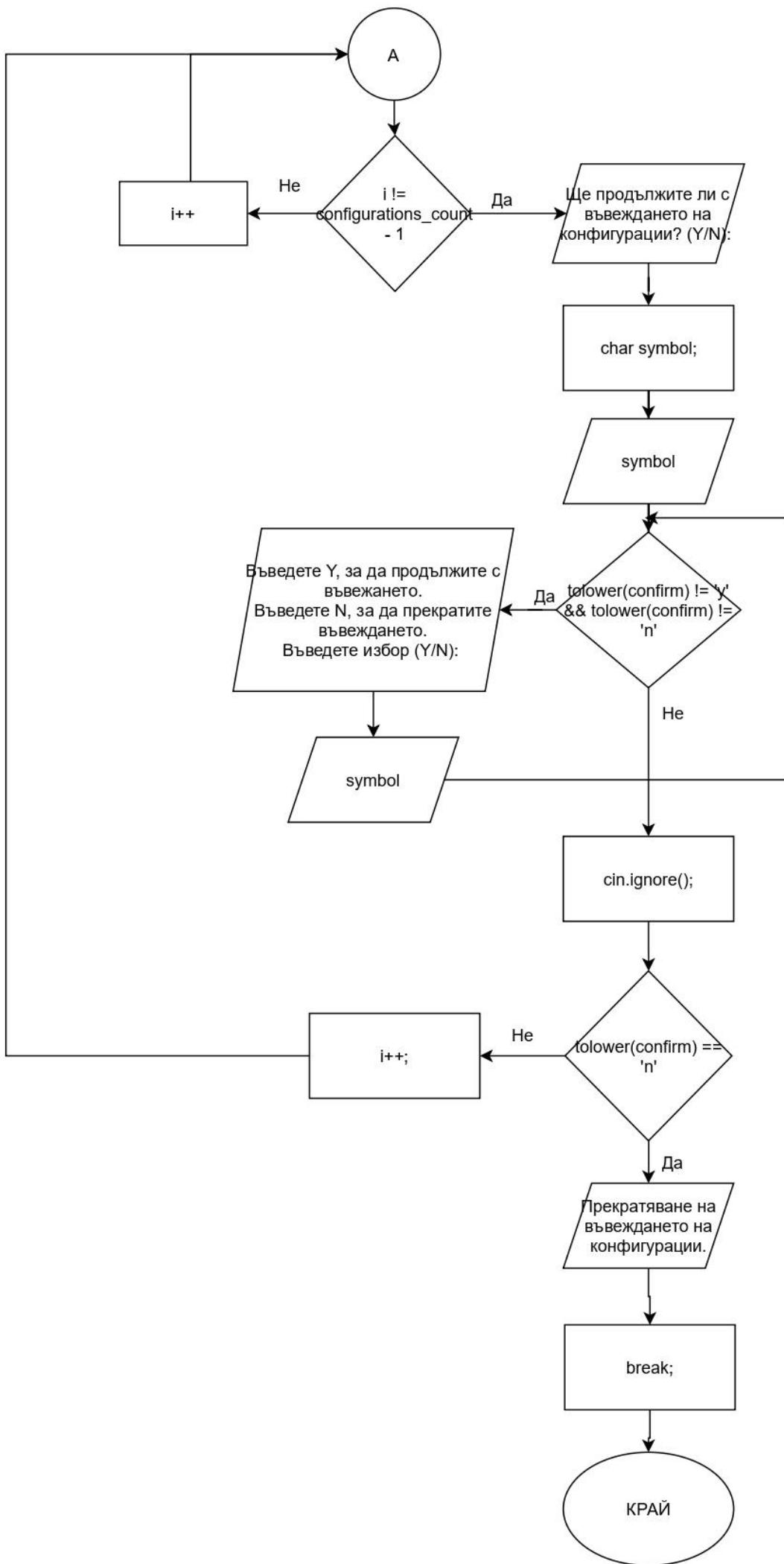


```
void add_configuration(Computer configurations[], int& present_configurations_count, string& processor_manufacturer,  
string& processor_model, string& computer_id, string& computer_brand, string& computer_model, string& available_status,  
double& processor_frequency, double& computer_ram, double& computer_price, int& processor_cores);
```



```
void process_add_configuration_request(Computer configurations[], int& present_configurations_count);
```





Екранни снимки с примерни входни и изходни данни

a. Първоначале изглед

```
===== ДОБАВЯНЕ НА НОВИ КОНФИГУРАЦИИ =====
```

```
Въведете брой конфигурации за добавяне:
```

b. Изглед след невалидна меню опция (1)

```
===== ДОБАВЯНЕ НА НОВИ КОНФИГУРАЦИИ =====
```

```
Въведете брой конфигурации за добавяне: -1
```

```
Въведеният брой конфигурации не е валиден.
```

```
Броят конфигурации трябва да е в интервала: [1 - 100]
```

c. Изглед след невалидна меню опция (2)

```
===== ДОБАВЯНЕ НА НОВИ КОНФИГУРАЦИИ =====
```

```
Въведете брой конфигурации за добавяне: 101
```

```
Въведеният брой конфигурации не е валиден.
```

```
Има място за 100 конфигурации.
```

d. Изглед след добавена конфигурация

```
===== ДОБАВЯНЕ НА НОВИ КОНФИГУРАЦИИ =====
```

```
Въведете брой конфигурации за добавяне: 2
```

```
===== ИНФОРМАЦИЯ ЗА ПРОЦЕСОР =====
```

```
Въведете производител на процесора: Intel
```

```
Въведете модел на процесора: i9
```

```
Въведете честота на процесора (GHz): 4.22
```

```
Въведете брой ядра на процесора: 6
```

```
===== ИНФОРМАЦИЯ ЗА КОМПЮТЪР =====
```

```
Въведете сериен номер на компютъра: 0C-5F-D9-ED-0A-42
```

```
Въведете марка на компютъра: Acer
```

```
Въведете модел на компютъра: Nitro 5
```

```
Въведете RAM на компютъра (GB): 16
```

```
Въведете цена на компютъра: 2000.99
```

```
Наличен статус на конфигурацията (в продажба/продадена): в продажба
```

```
Конфигурацията е добавена успешно!
```

```
Ще продължите ли с въвеждането на конфигурации? (Y/N):
```

e. Изглед след добавена конфигурация и отказ от повторно въвеждане

Ще продължите ли с въвеждането на конфигурации? (Y/N): N

Прекратяване на въвеждането на конфигурации.

f. Изглед след добавяне на конфигурация с вече съществуващ сериен номер

===== ДОБАВЯНЕ НА НОВИ КОНФИГУРАЦИИ =====

Въведете брой конфигурации за добавяне: 1

===== ИНФОРМАЦИЯ ЗА ПРОЦЕСОР =====

Въведете производител на процесора: Intel

Въведете модел на процесора: i9

Въведете честота на процесора (GHz): 4.22

Въведете брой ядра на процесора: 6

===== ИНФОРМАЦИЯ ЗА КОМПЮТЪР =====

Въведете сериен номер на компютъра: 0C-5F-D9-ED-0A-42

Вече съществува конфигурация с този сериен номер!

Въведете сериен номер на компютъра:

g. Изглед след оставяне на празно поле

===== ИНФОРМАЦИЯ ЗА КОМПЮТЪР =====

Въведете сериен номер на компютъра: 0C-5F-D9-ED-0A-42

Вече съществува конфигурация с този сериен номер!

Въведете сериен номер на компютъра:

Въведете сериен номер на компютъра:

Въведете сериен номер на компютъра: Валидация, която не разрешава

Въведете сериен номер на компютъра: оставянето на поле празно.

Въведете сериен номер на компютъра:

Въведете сериен номер на компютъра:

Въведете сериен номер на компютъра:

h. Излгед след негативна RAM стойност

Въведете RAM на компютъра (GB): -1
RAM стойността трябва да е по-голяма от 0.
Въведете RAM на компютъра (GB):

i. Излгед след невалиден наличен статус

Наличен статус на конфигурацията (в продажба/продадена): невалиден вход
Наличен статус на конфигурацията (в продажба/продадена):
Наличен статус на конфигурацията (в продажба/продадена): asokdoaskdoasdkop
Наличен статус на конфигурацията (в продажба/продадена):

C. Извеждане на всички конфигурации на екрана

- a. извеждане на конфигурациите с най-голяма тактова честота на процесора
- b. извеждане на конфигурациите от дадена марка

```
void print_configuration(Computer& computer);

void print_all_configurations(Computer configurations[], int
    present_configurations_count);

void process_print_all_configurations_request(Computer configurations[], int
    present_configurations_count);

bool compare_strings_case_insensitive(string first, string second);

void print_configurations_by_brand(Computer configurations[], int
    present_configurations_count, string& brand);

void process_print_configurations_by_brand_request(Computer configurations[], int
    present_configurations_count);

double find_max_processor_frequency(Computer configurations[], int
    present_configurations_count);

void print_configurations_by_processor_frequency(Computer configurations[], int
    present_configurations_count, double processor_frequency);

void print_configurations_with_highest_processor_frequency(Computer configurations[], int
    present_configurations_count);
```

Функцията *print_configuration* извежда информация за една конфигурация. Приема като входен параметър конфигурация, чиято информация трябва да бъде изведена на монитора.

Функцията *print_all_configurations* извежда информация за всички записани конфигурации. Приема като входен параметър масив с всички запазени конфигурации и брой запазени конфигурации. Итерира през масива с конфигурации и при всяка итерация извиква *print_configuration* с текущата конфигурация.

Функционалността за извеждане на всички конфигурации е реализирана във функцията *process_print_all_configurations_request*. Приема като входен параметър масив с всички запазени конфигурации и брой запазени конфигурации. Извежда информативно съобщение на екрана. Ако броят на запазените конфигурации е 0, извежда съобщени на екрана и приключва изпълнение. В противен случай, извиква функцията *print_all_configurations*.

Функцията *compare_strings_case_insensitive* сравнява две string стойности без значение дали буквата е малка или главна. Приема като входни параметри две string стойности и връща булев резултат. Ако дължината на string-овете е различна връща *false* и прекратява изпълнение. В противен случай обхожда първият string и сравнява символите му със символите от втория низ чрез функцията *tolower*. Ако има различен символ връща *false*, в противен случай – *true*.

Функцията *print_configurations_by_brand* извежда всички конфигурации с дадена марка. Приема като входни параметри масив с всички запазени конфигурации, брой запазени конфигурации и името на марката. Дефинира булева променлива, която е флаг за това дали са

намерени конфигурации от търсената марка. Функцията итерира през всички запазени конфигурации и проверява дали марката на текущата конфигурацията е същата като въведената. Сравняването става чрез *compare_strings_case_insensitive*. Ако марките са равни конфигурацията се извежда чрез *print_configuration* и булевата променлива присвоява стойност *true*. Ако не са намерени конфигурации с марката на екрана се извежда съобщение.

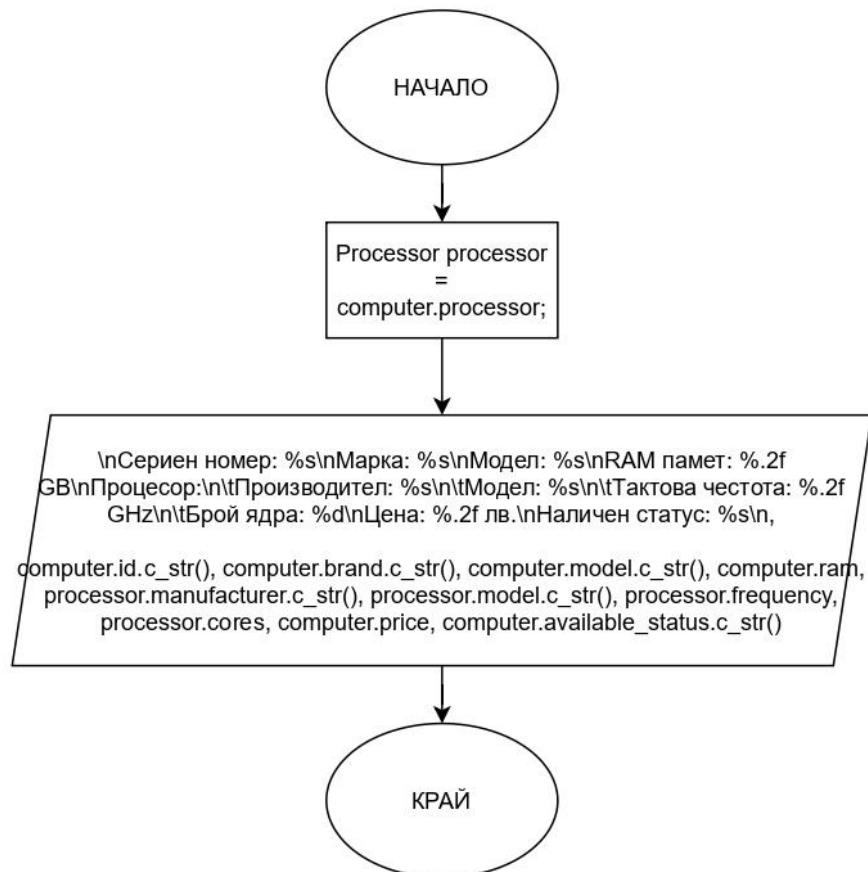
Функционалността за извеждане на всички конфигурации от дадена марка е реализирана във функцията *process_print_configurations_by_brand_request*. Приема като входен параметър масив с всички запазени конфигурации и брой запазени конфигурации. Извежда информативно съобщение на екрана. Ако броят на запазените конфигурации е 0, извежда съобщени на екрана и приключва изпълнение. В противен случай се дефинира string променлива за името на марката. Потребителят въвежда марка като входът е валидиран чрез do-while цикъл и *is_string_empty*. Накрая се извиква функцията *print_configurations_by_brand* с въведената марка.

Функцията *find_max_processor_frequency* намира най-голямата честота на процесор от запазените конфигурации. Приема като входни параметри масив с всички запазени конфигурации, брой запазени конфигурации и името на марката. Връща като резултат *double* стойност. Дефинира double променлива и ѝ присвоява честота на първият процесор. След това итерира през останалите конфигурации и проверява дали съществува процесор с по-голяма честота. Ако такъв съществува, стойността на променливата се актуализира. Накрая се връща стойността на променливата.

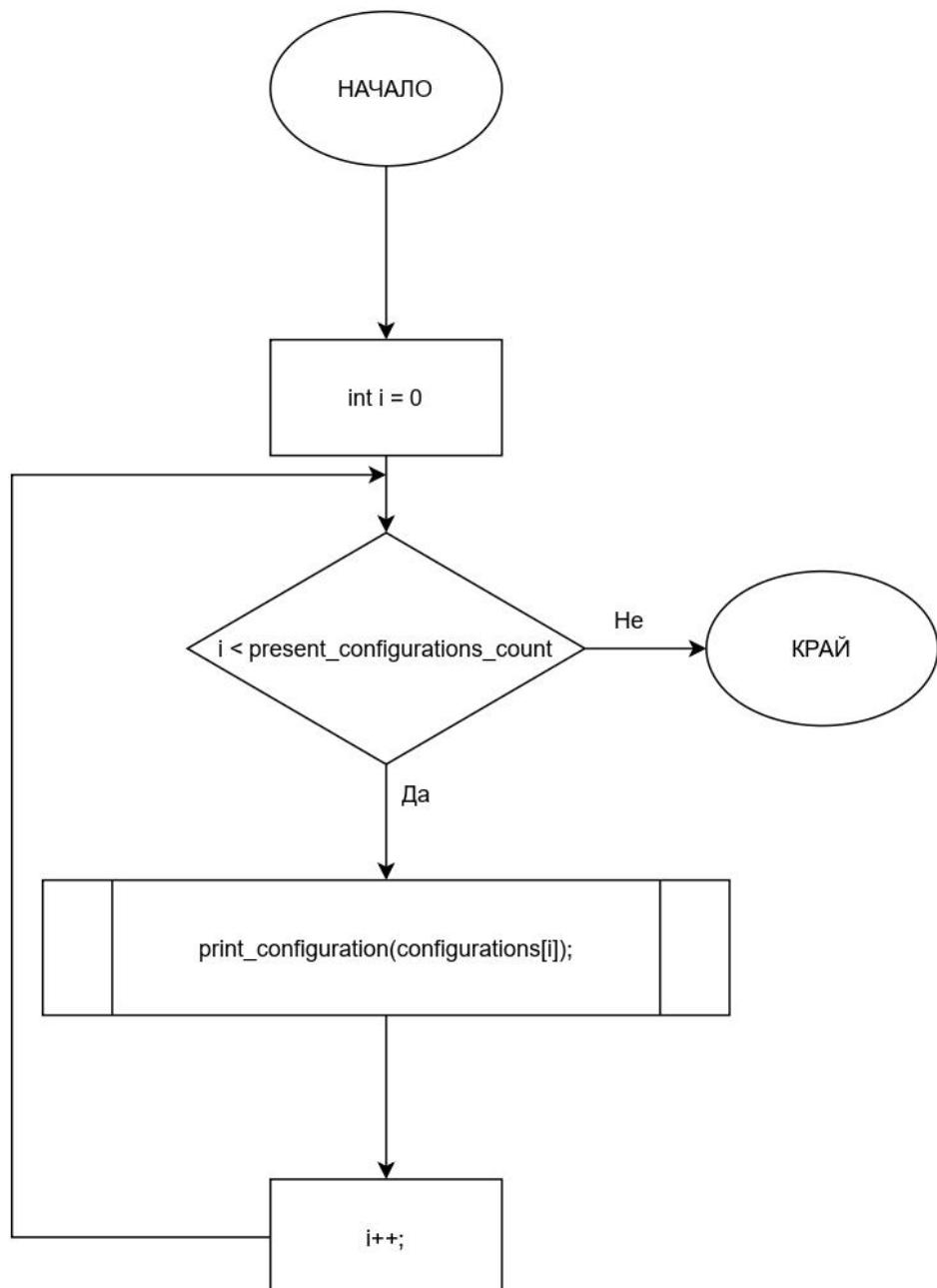
Функцията *print_configurations_by_processor_frequency* извежда всички конфигурации с дадена честота на процесора. Приема като входни параметри масив с всички запазени конфигурации, брой запазени конфигурации и желаната честота на процесора. Функцията итерира през всички запазени конфигурации и проверява дали честота на текущата конфигурация е равна на подадената като параметър. Ако честотите са равни конфигурацията се извежда чрез *print_configuration*.

Функционалността за извеждане на всички конфигурации с дадена честота на процесора е реализирана във функцията *print_configurations_with_highest_processor_frequency*. Приема като входен параметър масив с всички запазени конфигурации и брой запазени конфигурации. Извежда информативно съобщение на екрана. Ако броят на запазените конфигурации е 0, извежда съобщени на екрана и приключва изпълнение. В противен случай се дефинира double променлива и ѝ се присвоява най-високата честота на процесор чрез *find_max_processor_frequency*. След това се извеждат конфигурациите чрез функцията *print_configurations_by_processor_frequency*.

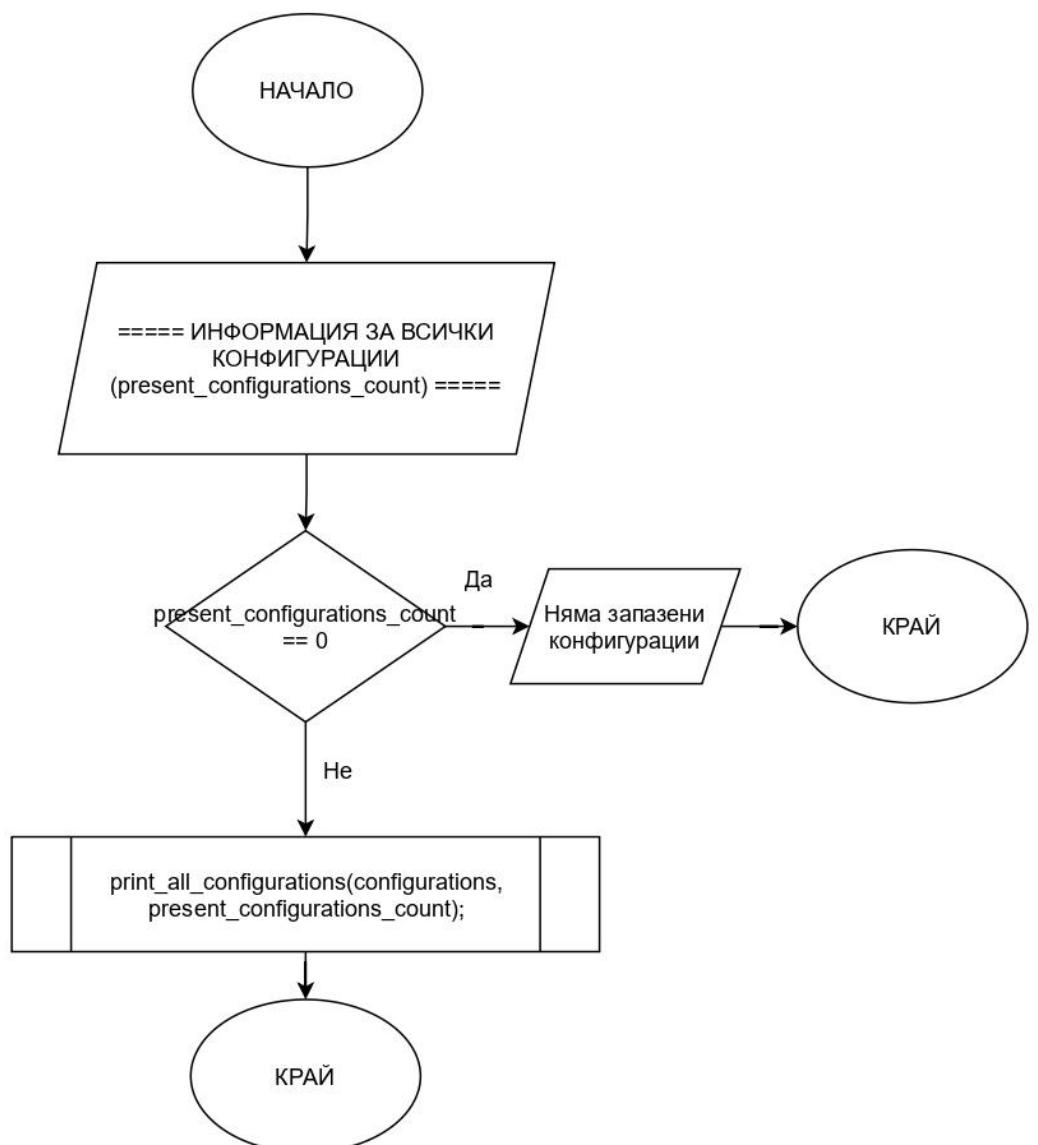
```
void print_configuration(Computer& computer);
```



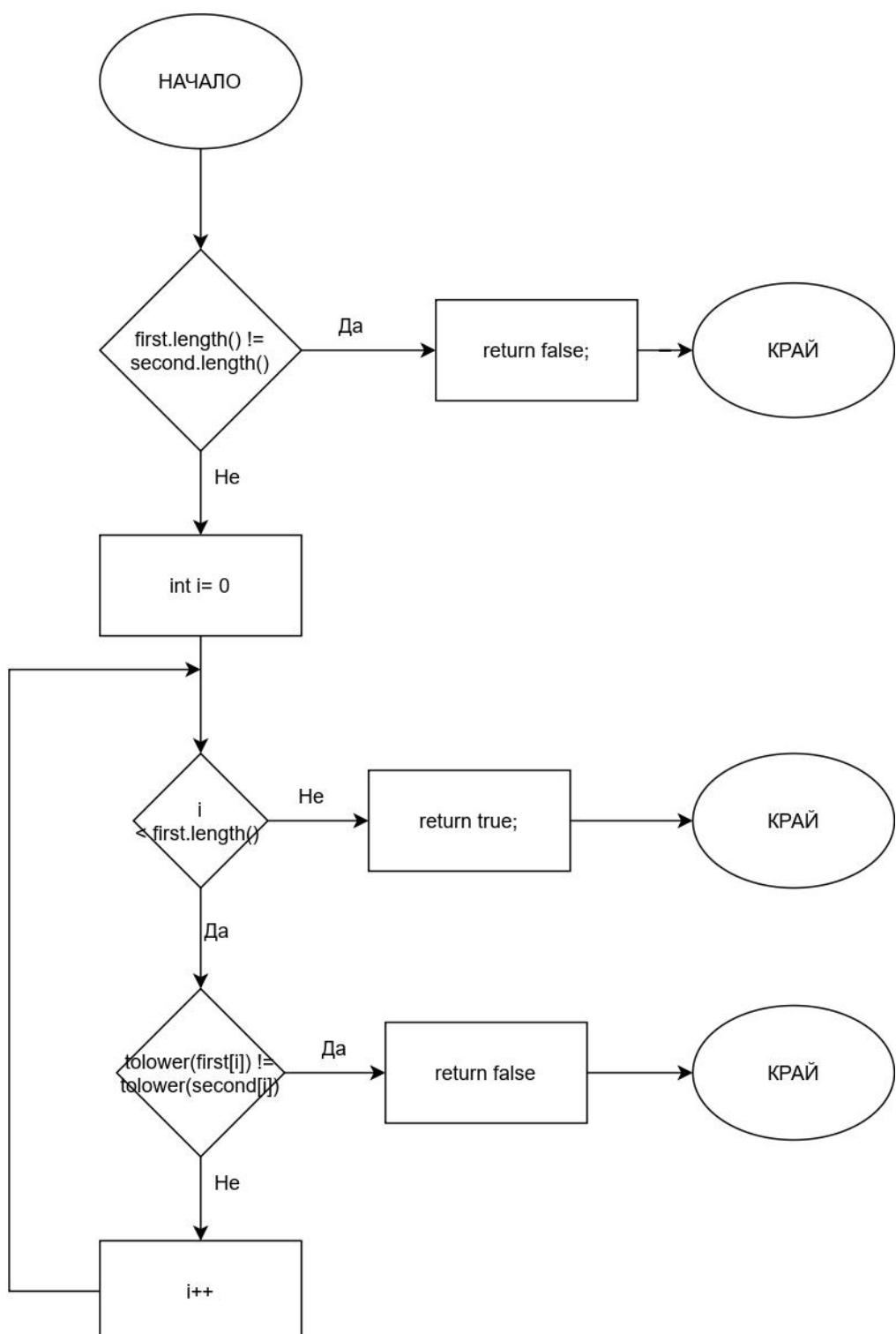
```
void print_all_configurations(Computer configurations[], int present_configurations_count);
```



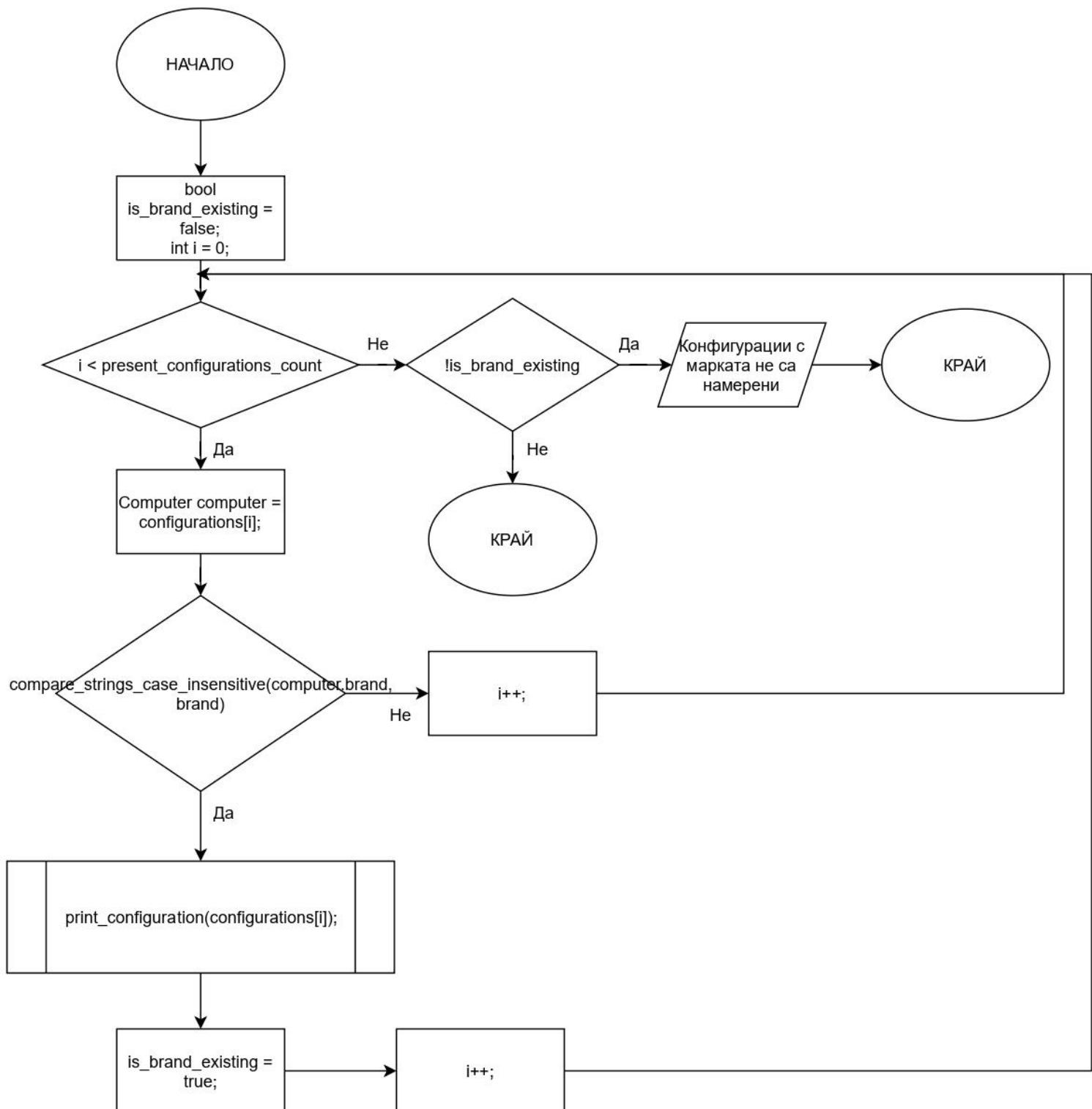
```
void process_print_all_configurations_request(Computer configurations[], int present_configurations_count);
```



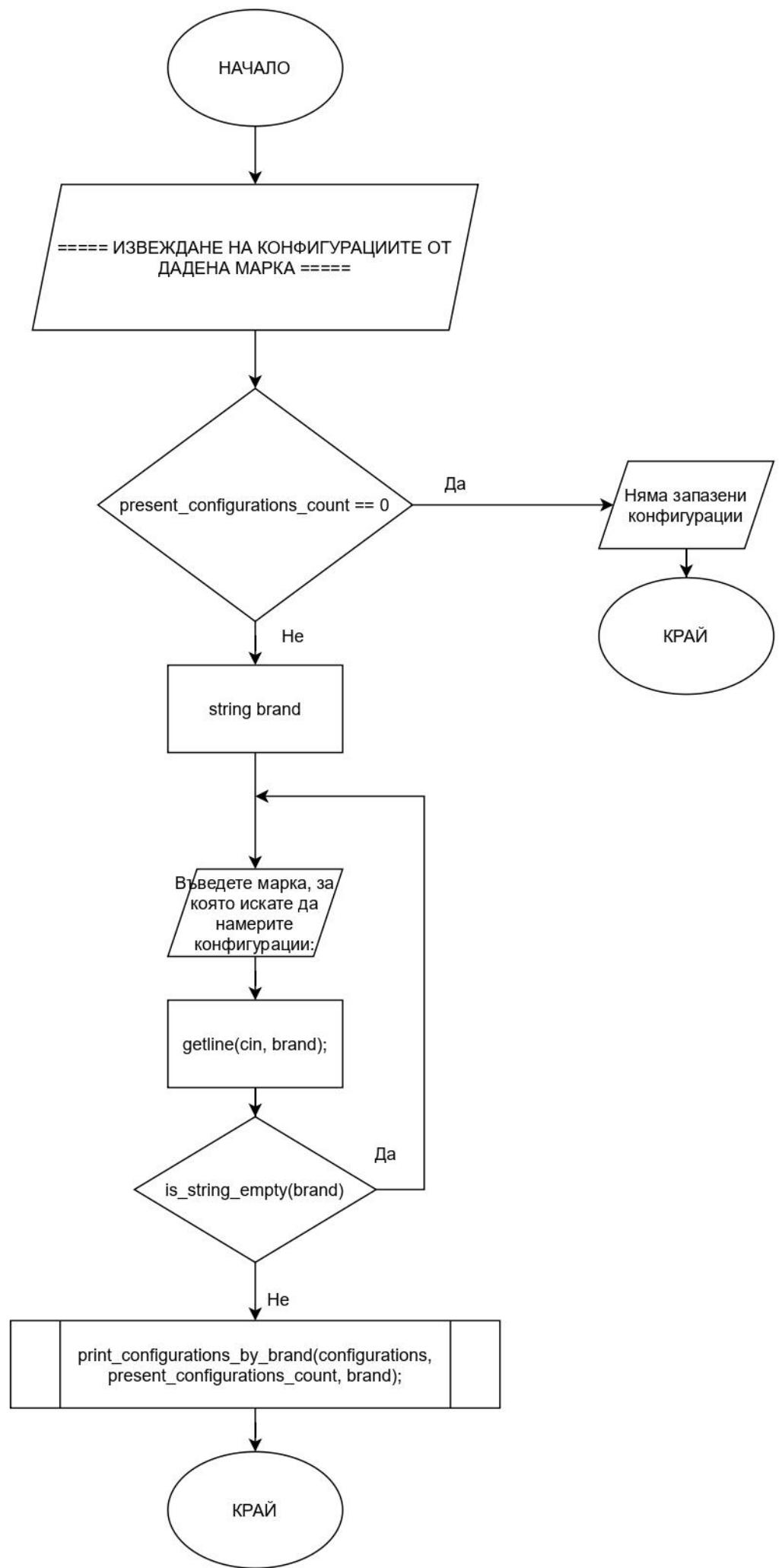
```
bool compare_strings_case_insensitive(string first, string second);
```



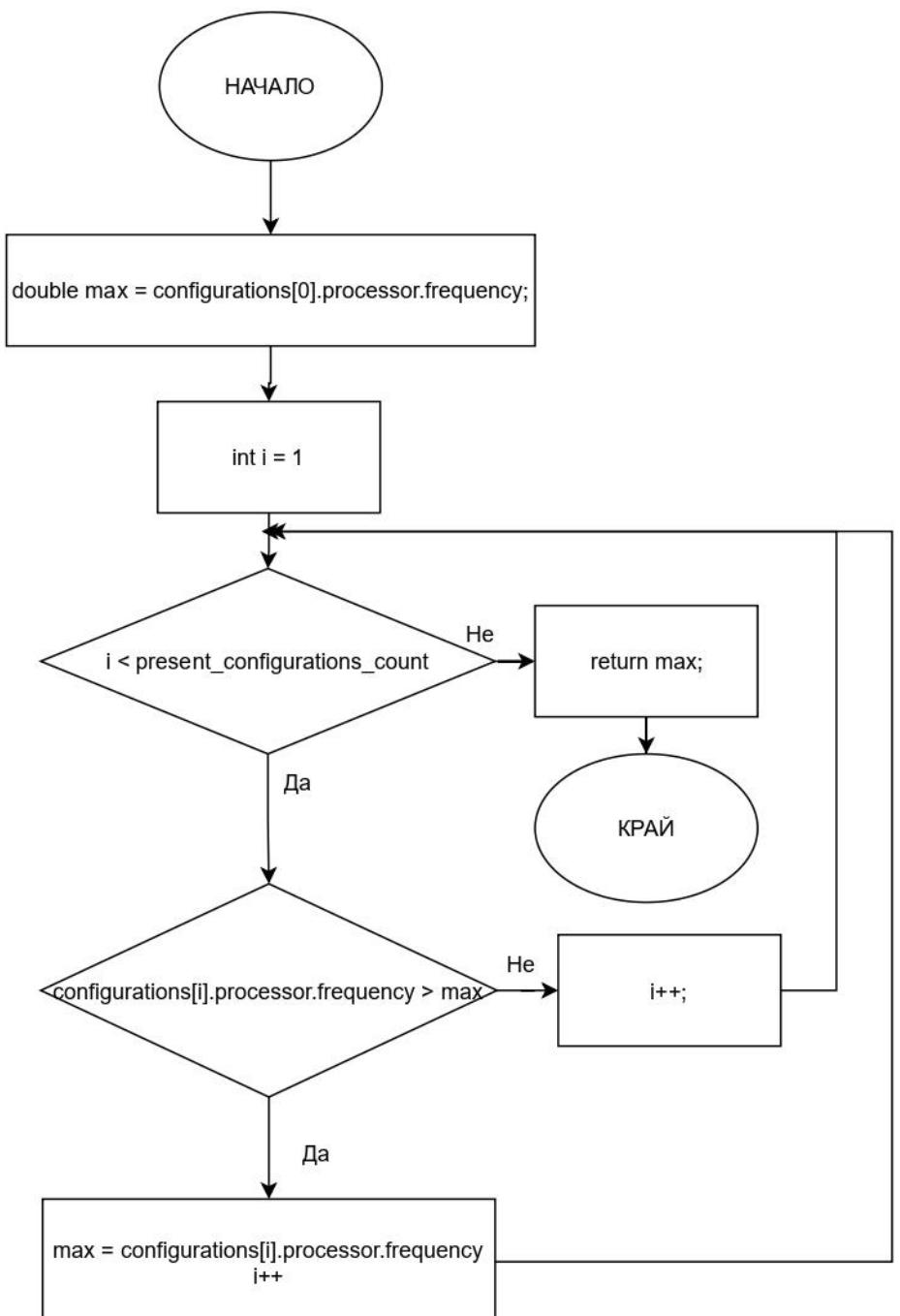
```
void print_configurations_by_brand(Computer configurations[], int present_configurations_count, string& brand);
```



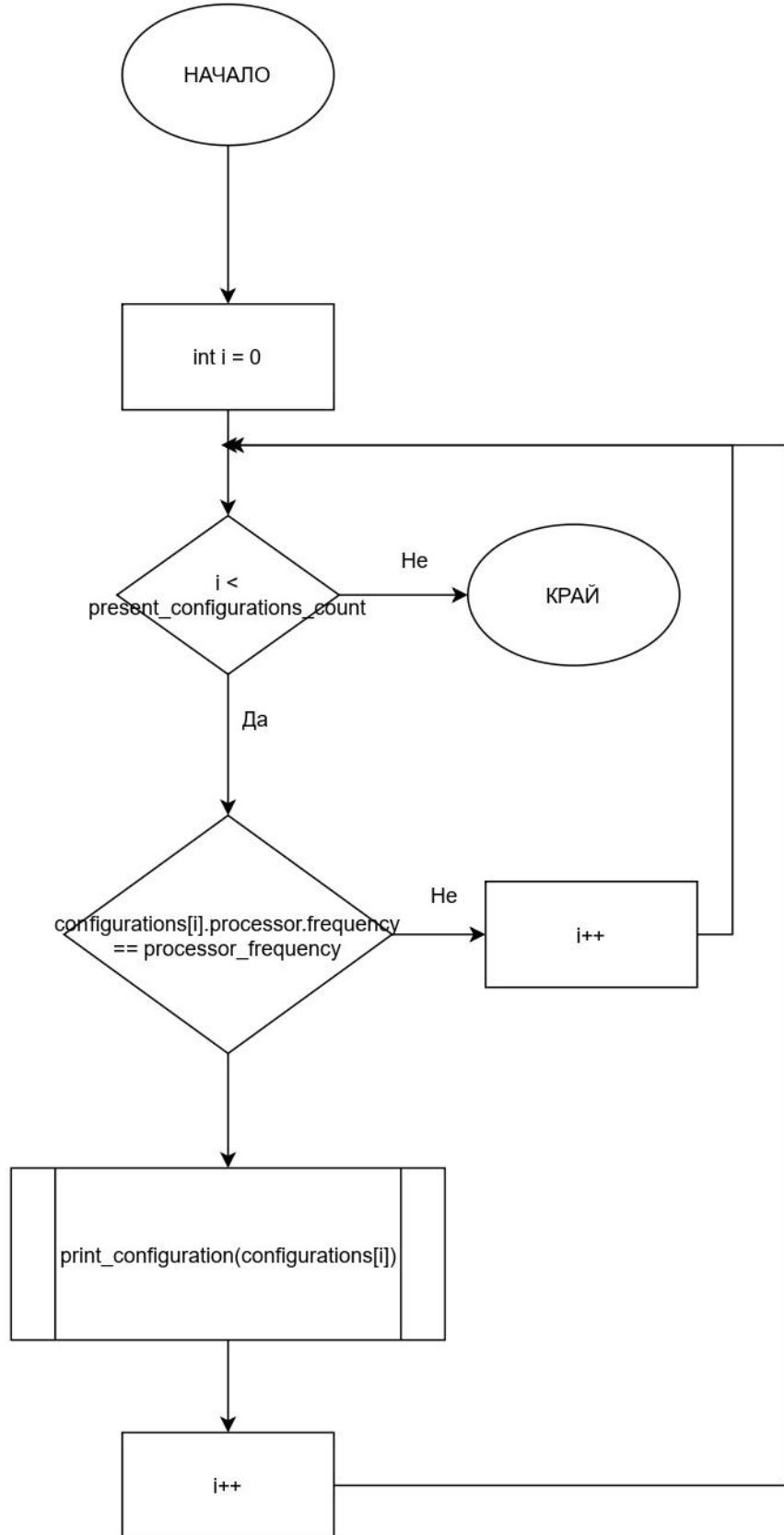
```
void process_print_configurations_by_brand_request(Computer configurations[], int present_configurations_count);
```



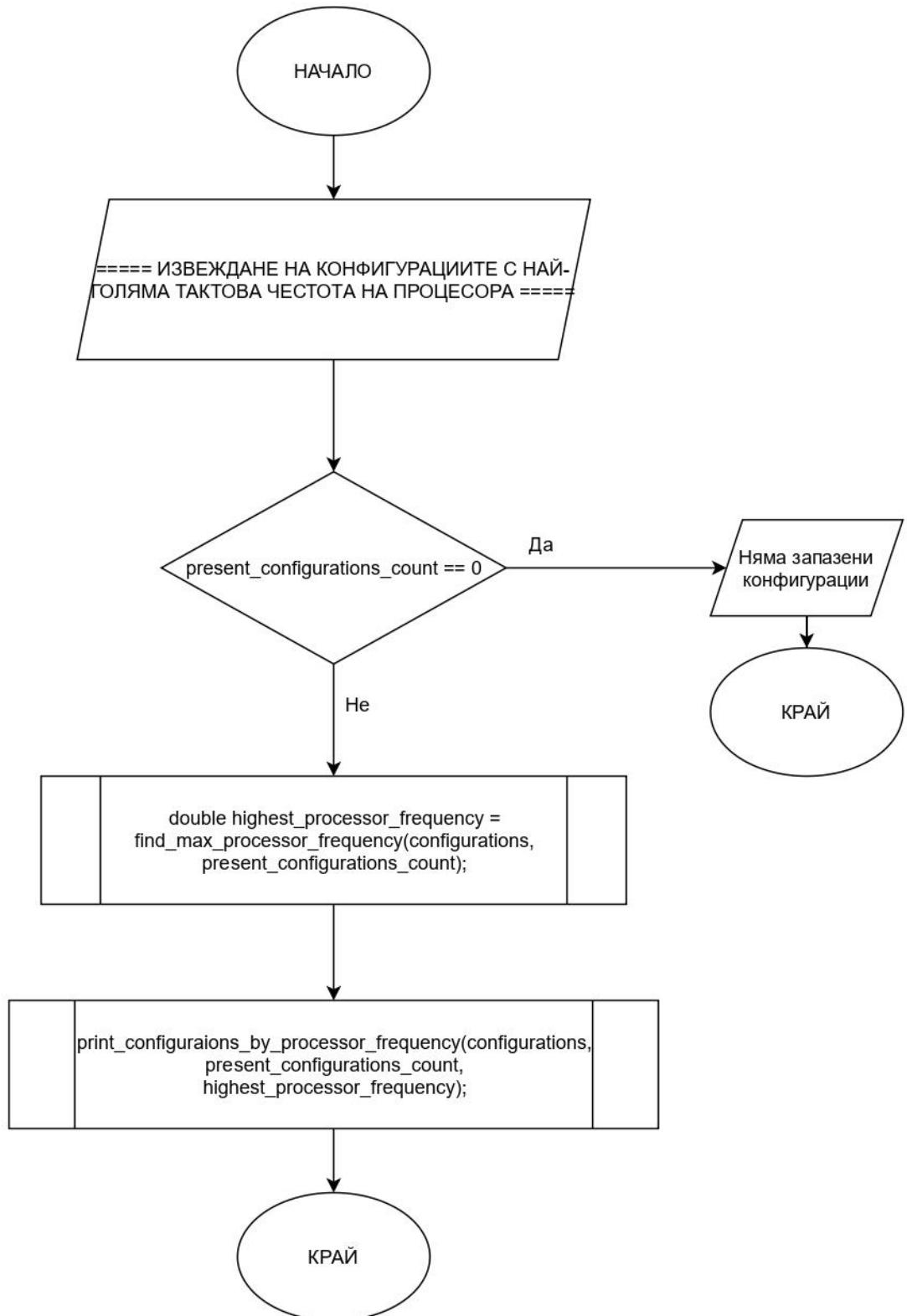
```
double find_max_processor_frequency(Computer configurations[], int present_configurations_count);
```



```
void print_configurations_by_processor_frequency(Computer configurations[], int present_configurations_count, double processor_frequency);
```



```
void print_configurations_with_highest_processor_frequency(Computer configurations[], int present_configurations_count);
```



Екранни снимки с примерни входни и изходни данни

- a. Изглед при извеждане на всички конфигурации, когато няма записани такива

```
===== ИНФОРМАЦИЯ ЗА ВСИЧКИ КОНФИГУРАЦИИ (0) =====
```

Няма запазени конфигурации.

```
===== НАЧАЛНО МЕНЮ =====
```

- b. Изглед при изведени всички конфигурации

```
===== ИНФОРМАЦИЯ ЗА ВСИЧКИ КОНФИГУРАЦИИ (16) =====
```

Сериен номер: 15-58-64-54-29-C1

Марка: Acer

Модел: Nitro

RAM памет: 16,00 GB

Процесор:

Производител: Intel

Модел: i7

Тактова честота: 3,00 GHz

Брой ядра: 6

Цена: 2200,00 лв.

Наличен статус: в продажба

Сериен номер: 17-48-54-64-29-C1

Марка: Nokia

Модел: Nitro

RAM памет: 16,00 GB

Процесор:

Производител: Intel

Модел: i7

Тактова честота: 3,00 GHz

Брой ядра: 6

Цена: 2200,00 лв.

Наличен статус: в продажба

с. Изглед при извеждане на конфигурациите с най-висока честота на процесора

===== ИЗВЕЖДАНЕ НА КОНФИГУРАЦИИТЕ С НАЙ-ГОЛЯМА ТАКТОВА ЧЕСТОТА НА ПРОЦЕСОРА =====

Сериен номер: C3-B3-83-92-55-C2

Марка: HP

Модел: Nitro

RAM памет: 16,00 GB

Процесор:

Производител: Intel

Модел: i7

Тактовна честота: 5,00 GHz

Брой ядра: 6

Цена: 2200,00 лв.

Наличен статус: в продажба

Сериен номер: D1-B6-83-92-55-C2

Марка: Apple

Модел: A1

RAM памет: 16,00 GB

Процесор:

Производител: Intel

Модел: i9

Тактовна честота: 5,00 GHz

Брой ядра: 6

Цена: 2200,00 лв.

Наличен статус: в продажба

d. Изглед при извеждане на всички конфигурации от дадена марка, когато марката не е намерена

===== ИЗВЕЖДАНЕ НА КОНФИГУРАЦИИТЕ ОТ ДАДЕНА МАРКА =====

Въведете марка, за която искате да намерите конфигурации: No Existing Brand

Конфигурации с марката No Existing Brand не са намерени.

e. Изглед при извеждане на всички конфигурации от дадена марка, когато марката е намерена

===== ИЗВЕЖДАНЕ НА КОНФИГУРАЦИИТЕ ОТ ДАДЕНА МАРКА =====

Въведете марка, за която искате да намерите конфигурации: Apple

Сериен номер: D1-B6-83-92-55-C2

Марка: Apple

Модел: A1

RAM памет: 16,00 GB

Процесор:

Производител: Intel

Модел: i9

Тактовна честота: 5,00 GHz

Брой ядра: 6

Цена: 2200,00 лв.

Наличен статус: в продажба

D. Корекция на данни за конфигурация

- a. Въвежда се сериен номер и данни за корекция
- b. Ако конфигурацията е продадена, не може да се прави корекция

```
bool is_configuration_available(string& id, Computer configurations[], int
    present_configurations_count);

Computer get_configuration_by_id(string& id, Computer configurations[], int
    present_configurations_count);

void update_configuration_id(string& old_id, Computer configurations[], int
    present_configurations_count);

void update_configuration_brand(string& id, Computer configurations[], int
    present_configurations_count);

void update_configuration_model(string& id, Computer configurations[], int
    present_configurations_count);

void update_configuration_ram(string& id, Computer configurations[], int
    present_configurations_count);

void update_configuration_price(string& id, Computer configurations[], int
    present_configurations_count);

void update_configuration_status(string& id, Computer configurations[], int
    present_configurations_count);

void update_processor_manufacturer(string& id, Computer configurations[], int
    present_configurations_count);

void update_processor_model(string& id, Computer configurations[], int
    present_configurations_count);

void update_processor_frequency(string& id, Computer configurations[], int
    present_configurations_count);

void update_processor_cores(string& id, Computer configurations[], int
    present_configurations_count);

void update_processor(string& configuration_id, Computer configurations[], int
    present_configurations_count);

void update_configuration(Computer configurations[], int
    present_configurations_count);
```

Функцията *is_configuration_available* проверява дали дадена конфигурация е налична за продажба. Връща булев резултат. Приема като параметри серииният номер на конфигурацията, масив с всички запазени конфигурации, брой запазени конфигурации. Обхожда всички конфигурации и проверява дали серииният номер на текущата конфигурация е равен на подаденият като параметър. Ако условието е вярно, се прави проверка дали наличният статус е равен на „в продажба“. Връща *true* ако конфигурацията е налична и *false* ако конфигурацията не е налична.

Функцията *get_configuration_by_id* се използва за взимане на конфигурация по сериен номер. Приема като параметри серииният номер на конфигурацията, масив с всички запазени конфигурации, брой запазени конфигурации. Обхожда всички конфигурации и проверява дали серииният номер на текущата конфигурация е равен на подаденият като параметър. Ако условието е вярно, се връща текущата конфигурация.

Функцията *update_configuration_id* се използва за актуализиране на серииният номер на дадена конфигурация. Приема като параметри серииният номер на конфигурацията, масив с всички запазени конфигурации, брой запазени конфигурации. Извежда информативно съобщение и извежда детайли за конфигурацията, която предстои да бъде актуализирана чрез *get_configuration_by_id* и *print_configuration*. След това се дефинира променлива от тип string за новият сериен номер. Входът за серииният номер е валидиран чрез do-while цикъл и функцията *is_string_empty*. След въвеждане на нов сериен номер се прави проверка дали съществува конфигурация с този номер. Ако съществува такава конфигурация се извежда съобщение и функцията приключва изпълнението си. Ако конфигурацията не съществува се итерира през всички конфигурации и се проверява за съвпадение в серииният номер. След като се намери конфигурацията, серииният й номер се заменя с нововъведения. Променливата, която е съхранявала старият сериен номер присвоява новият такъв. Извежда се съобщение на екрана и обновената конфигурация.

Функцията *update_configuration_brand* се използва за актуализиране на марката на дадена конфигурация. Приема като параметри серииният номер на конфигурацията, масив с всички запазени конфигурации, брой запазени конфигурации. Извежда информативно съобщение и извежда детайли за конфигурацията, която предстои да бъде актуализирана чрез *get_configuration_by_id* и *print_configuration*. След това се дефинира променлива от тип string за новата марка. Входът за марката е валидиран чрез do-while цикъл и функцията *is_string_empty*. Обхождат се всички конфигурации и се проверява за съвпадение в серииният номер. След като се намери конфигурацията, марката ѝ се заменя с нововъведената. Извежда се съобщение на екрана и обновената конфигурация.

Функцията *update_configuration_model* се използва за актуализиране на модела на дадена конфигурация. Приема като параметри серииният номер на конфигурацията, масив с всички запазени конфигурации, брой запазени конфигурации. Извежда информативно съобщение и извежда детайли за конфигурацията, която предстои да бъде актуализирана чрез *get_configuration_by_id* и *print_configuration*. След това се дефинира променлива от тип string за новият модел. Входът за модела е валидиран чрез do-while цикъл и функцията *is_string_empty*. Обхождат се всички конфигурации и се проверява за съвпадение в серииният номер. След като се намери конфигурацията, моделът ѝ се заменя с нововъведената. Извежда се съобщение на екрана и обновената конфигурация.

Функцията *update_configuration_ram* се използва за актуализиране на RAM стойността на дадена конфигурация. Приема като параметри серииният номер на конфигурацията, масив с всички запазени конфигурации, брой запазени конфигурации. Извежда информативно съобщение и извежда детайли за конфигурацията, която предстои да бъде актуализирана чрез *get_configuration_by_id* и *print_configuration*. След това се дефинира променлива от тип double за RAM стойност. Входът за RAM стойността е валидиран чрез while цикъл (не може да е по-малка от 0). Обхождат се всички конфигурации и се проверява за съвпадение в серииният номер. След като се намери конфигурацията, RAM стойността ѝ се заменя с нововъведената. Извежда се съобщение на екрана и обновената конфигурация.

Функцията *update_configuration_price* се използва за актуализиране на цена на дадена конфигурация. Приема като параметри серийният номер на конфигурацията, масив с всички запазени конфигурации, брой запазени конфигурации. Извежда информативно съобщение и извежда детайли за конфигурацията, която предстои да бъде актуализирана чрез *get_configuration_by_id* и *print_configuration*. След това се дефинира променлива от тип double за новата цена. Входът за цената е валидиран чрез while цикъл (не може да е по-малка от 0). Обхождат се всички конфигурации и се проверява за съвпадение в серийния номер. След като се намери конфигурацията, цената ѝ се заменя с нововъведената. Извежда се съобщение на екрана и обновената конфигурация.

Функцията *update_configuration_status* се използва за актуализиране на наличният статус на дадена конфигурация. Приема като параметри серийният номер на конфигурацията, масив с всички запазени конфигурации, брой запазени конфигурации. Извежда информативно съобщение и извежда детайли за конфигурацията, която предстои да бъде актуализирана чрез *get_configuration_by_id* и *print_configuration*. След това се дефинира променлива от тип string за новият статус. Входът за статуса е валидиран чрез do-while цикъл. Обхождат се всички конфигурации и се проверява за съвпадение в серийния номер. След като се намери конфигурацията, статусът ѝ се заменя с нововъведения. Извежда се съобщение на екрана и обновената конфигурация.

Функцията *update_processor_manufacturer* се използва за актуализиране на производителя на процесора на дадена конфигурация. Приема като параметри серийният номер на конфигурацията, масив с всички запазени конфигурации, брой запазени конфигурации. Извежда информативно съобщение и извежда детайли за конфигурацията, която предстои да бъде актуализирана чрез *get_configuration_by_id* и *print_configuration*. След това се дефинира променлива от тип string за новият производител. Входът за производителя е валидиран чрез do-while цикъл и *is_string_empty*. Обхождат се всички конфигурации и се проверява за съвпадение в серийния номер. След като се намери конфигурацията, производителят на процесора ѝ се заменя с нововъведения. Извежда се съобщение на екрана и обновената конфигурация.

Функцията *update_processor_model* се използва за актуализиране на модела на процесора на дадена конфигурация. Приема като параметри серийният номер на конфигурацията, масив с всички запазени конфигурации, брой запазени конфигурации. Извежда информативно съобщение и извежда детайли за конфигурацията, която предстои да бъде актуализирана чрез *get_configuration_by_id* и *print_configuration*. След това се дефинира променлива от тип string за новият модел. Входът за производителя е валидиран чрез do-while цикъл и *is_string_empty*. Обхождат се всички конфигурации и се проверява за съвпадение в серийния номер. След като се намери конфигурацията, моделът на процесора ѝ се заменя с нововъведения. Извежда се съобщение на екрана и обновената конфигурация.

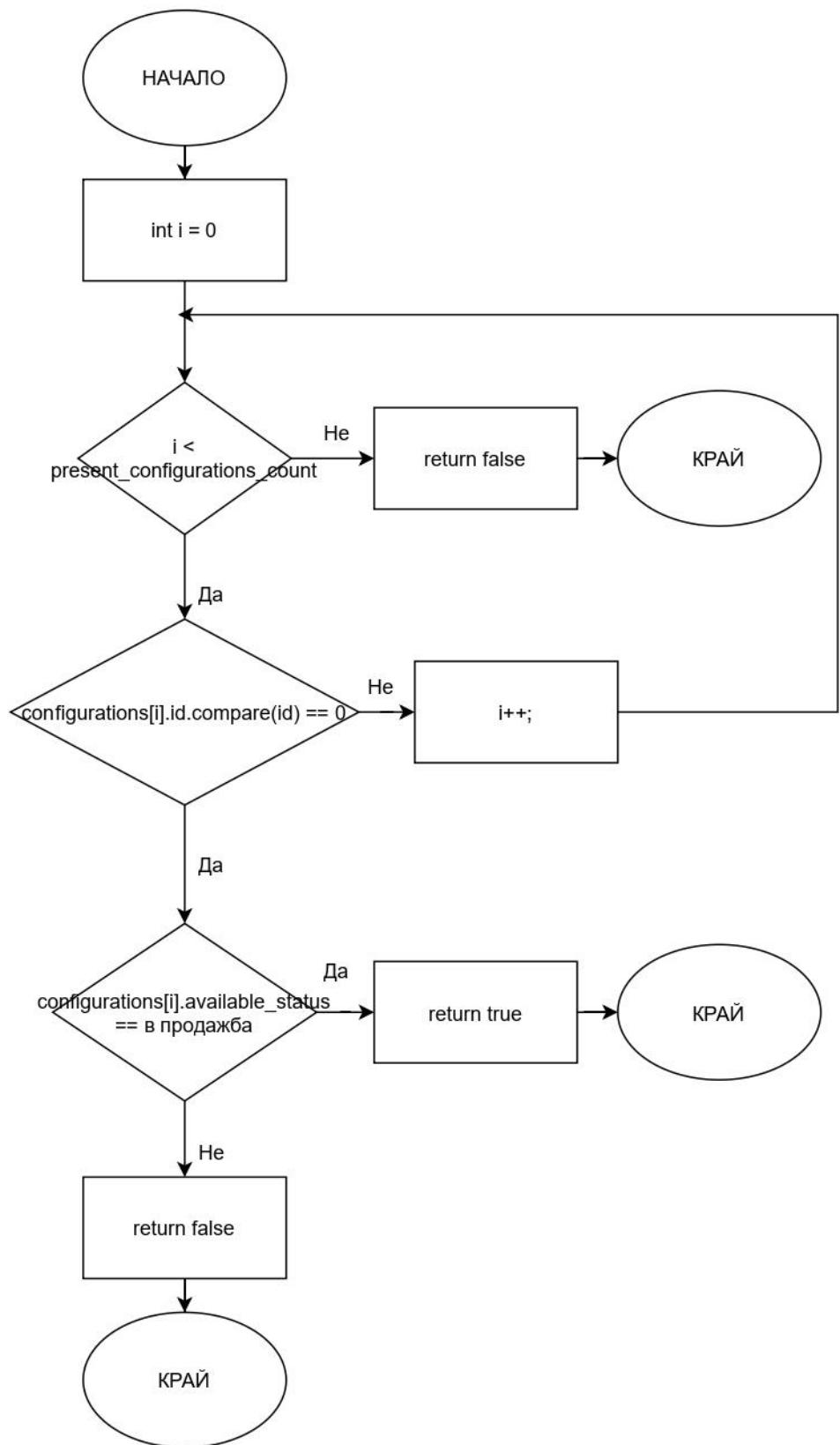
Функцията *update_processor_frequency* се използва за актуализиране на честотата на процесора на дадена конфигурация. Приема като параметри серийният номер на конфигурацията, масив с всички запазени конфигурации, брой запазени конфигурации. Извежда информативно съобщение и извежда детайли за конфигурацията, която предстои да бъде актуализирана чрез *get_configuration_by_id* и *print_configuration*. След това се дефинира променлива от тип double за новата честота. Входът за честотата е валидиран чрез while цикъл (не може да е по-малка от 0). Обхождат се всички конфигурации и се проверява за съвпадение в серийния номер. След като се намери конфигурацията, честотата на процесора ѝ се заменя с нововъведената. Извежда се съобщение на екрана и обновената конфигурация.

Функцията *update_processor_cores* се използва за актуализиране на броят ядра на процесора на дадена конфигурация. Приема като параметри серийният номер на конфигурацията, масив с всички запазени конфигурации, брой запазени конфигурации. Извежда информативно съобщение и извежда детайли за конфигурацията, която предстои да бъде актуализирана чрез *get_configuration_by_id* и *print_configuration*. След това се дефинира променлива от тип int за новият брой ядра. Входът за броят ядра е валидиран чрез while цикъл (не може да е по-малък от 0). Обхождат се всички конфигурации и се проверява за съвпадение в серийният номер. След като се намери конфигурацията, броят ядра на процесора ѝ се заменят с нововъведените. Извежда се съобщение на екрана и обновената конфигурация.

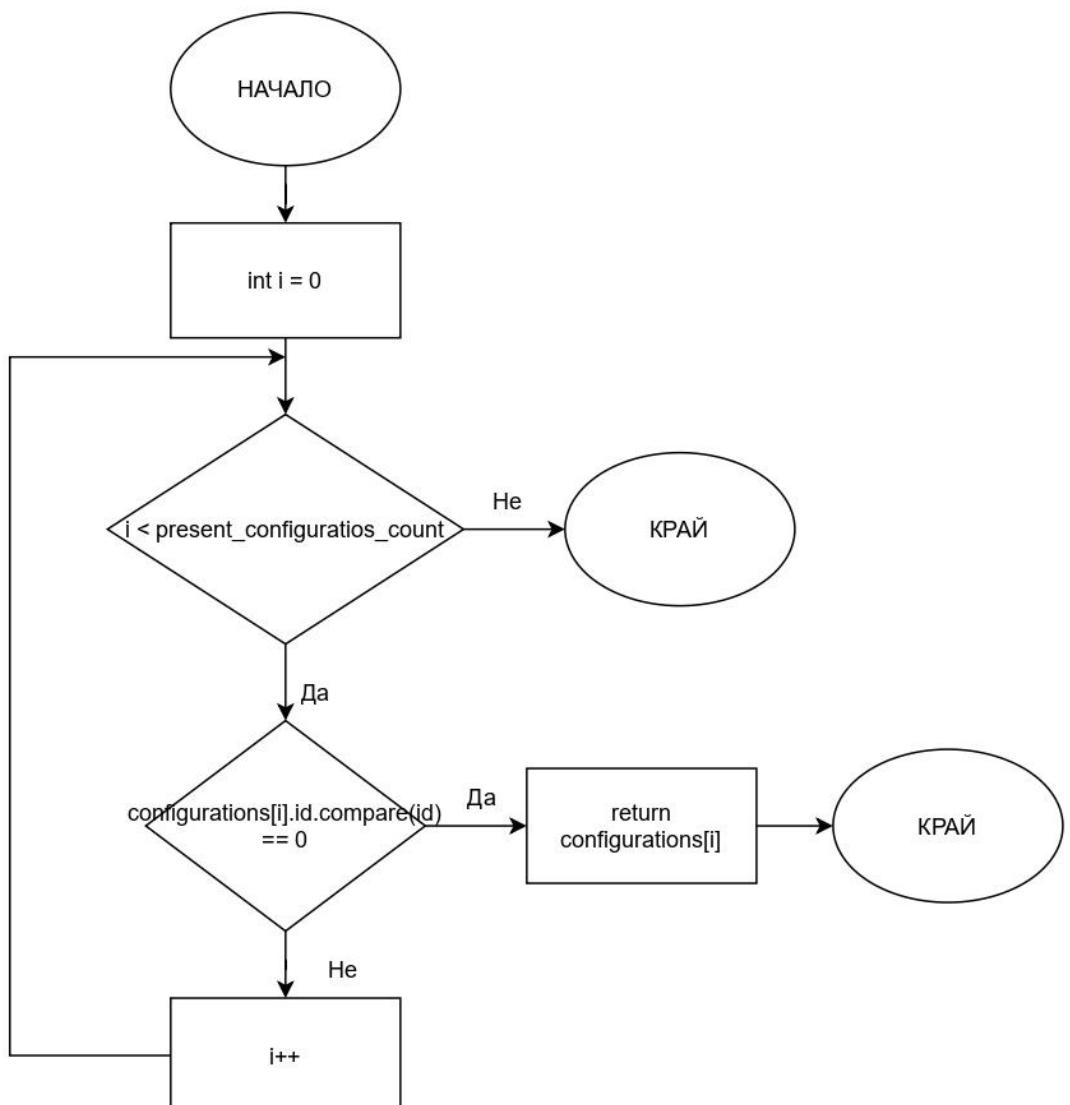
Функционалността за актуализиране на елемент на процесора е реализирана във функцията *update_processor*. Приема като параметри серийният номер на конфигурацията, масив с всички запазени конфигурации, брой запазени конфигурации. Извежда информативно съобщение на екрана и визуализира подменю за редакция на елементи по процесора. Дефинира променлива от тип int, която служи за определяне на меню опцията. Меню опцията е валидирана чрез do-while цикъл. Чрез конструкцията *switch* се определя коя функция за редакция да се извика.

Функционалността за актуализиране на елемент на конфигурация е реализирана във функцията *update_configuration*. Приема като параметри масив с всички запазени конфигурации, брой запазени конфигурации. Извежда информативно съобщение на екрана и приканва потребителят да въведе сериен номер на конфигурация за редактиране. Входът е валидиран чрез do-while цикъл и *is_string_empty*. Ако конфигурация със въведения сериен номер не съществува се извежда съобщение и функцията прекратява своето изпълнение. Ако конфигурацията е намерана, но не е налична за продажба се извежда съобщение и функцията прекратява изпълнението си. При успешно намерена и налична конфигурация се извежда съобщение и се визуализира подменю за редакция на елементи по конфигурацията. Дефинира се променлива от тип int, която служи за определяне на меню опцията. Меню опцията е валидирана чрез do-while цикъл. Чрез конструкцията *switch* се определя коя функция за редакция да се извика.

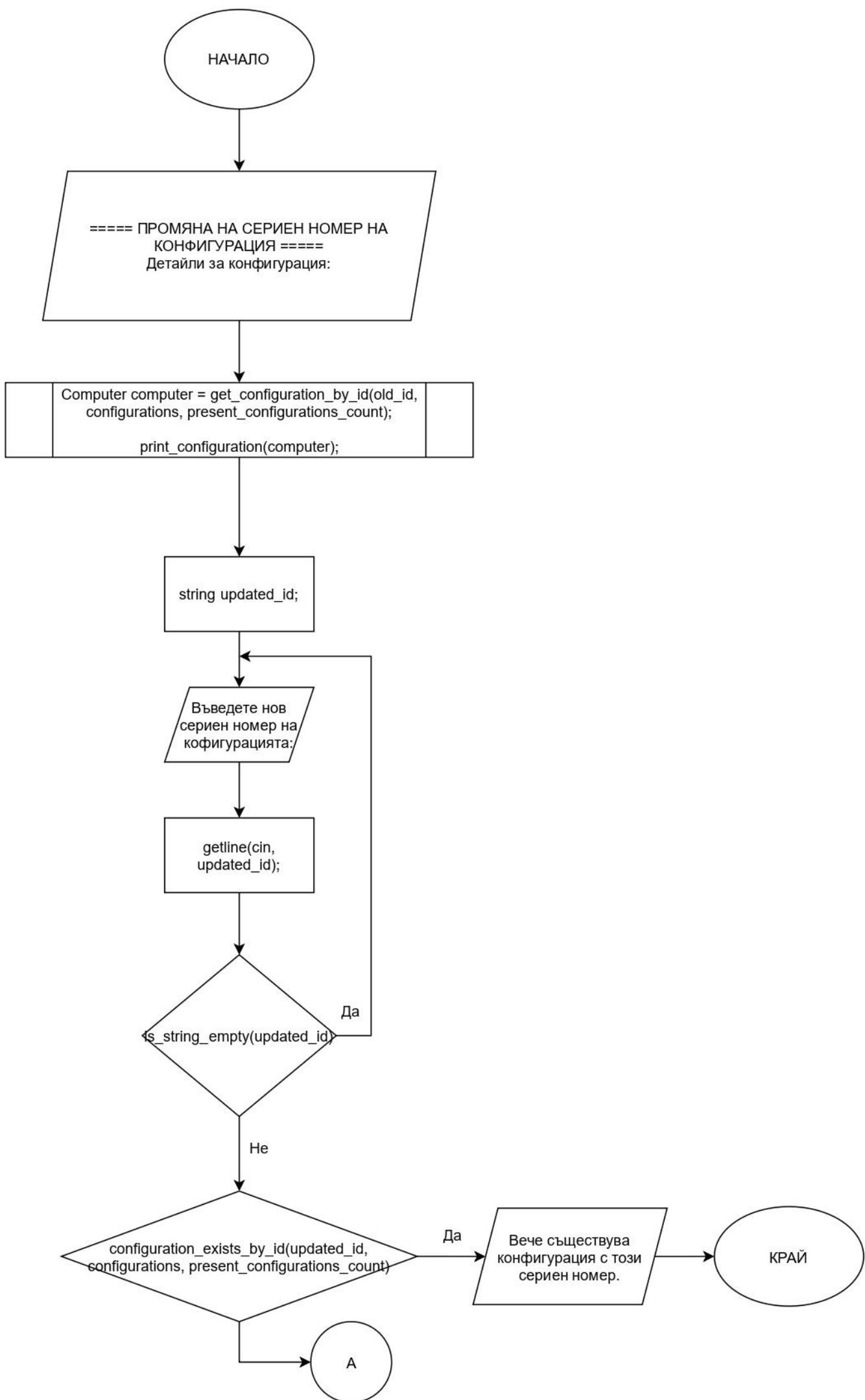
```
bool is_configuration_available(string& id, Computer configurations[], int present_configurations_count);
```

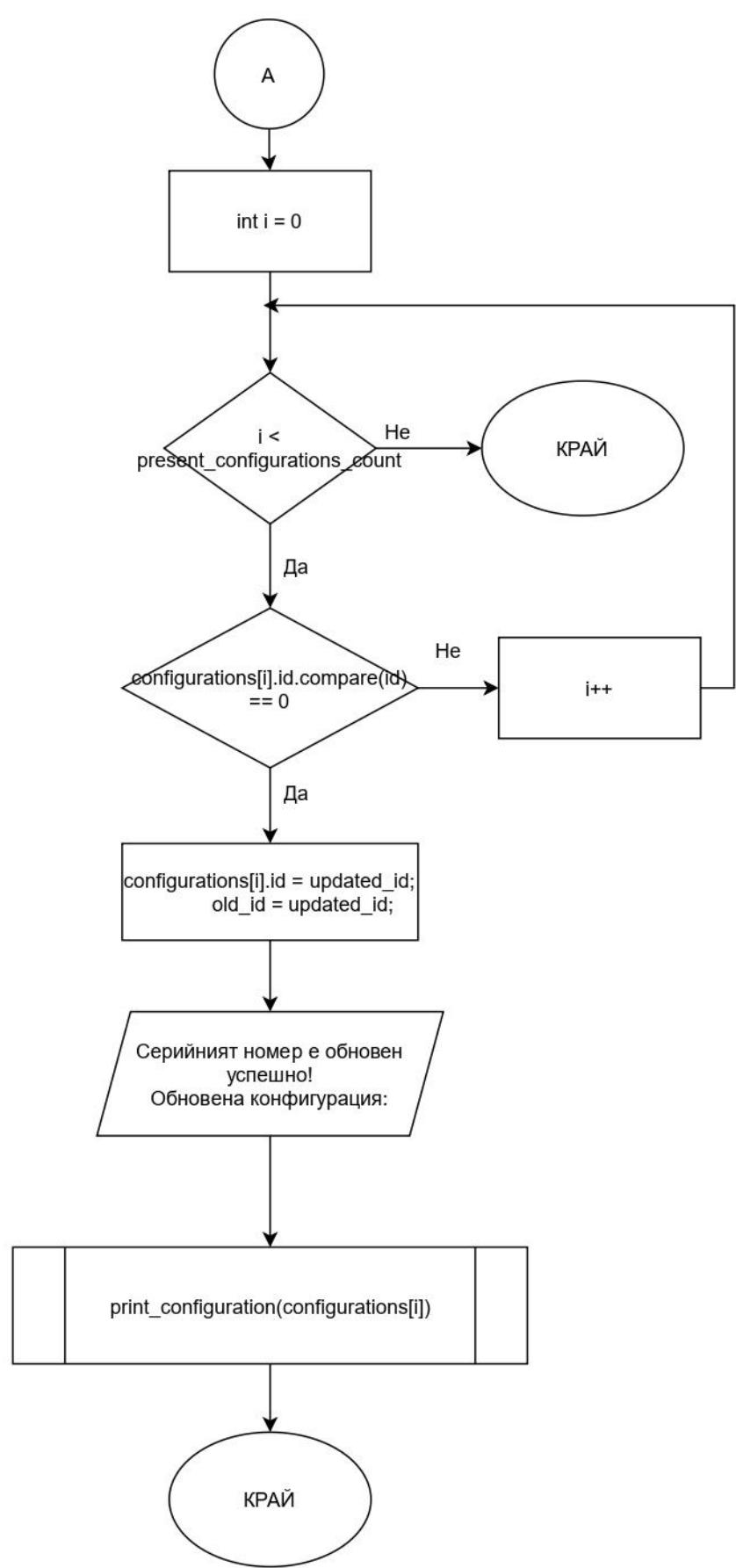


```
Computer get_configuration_by_id(string& id, Computer configurations[], int present_configurations_count);
```

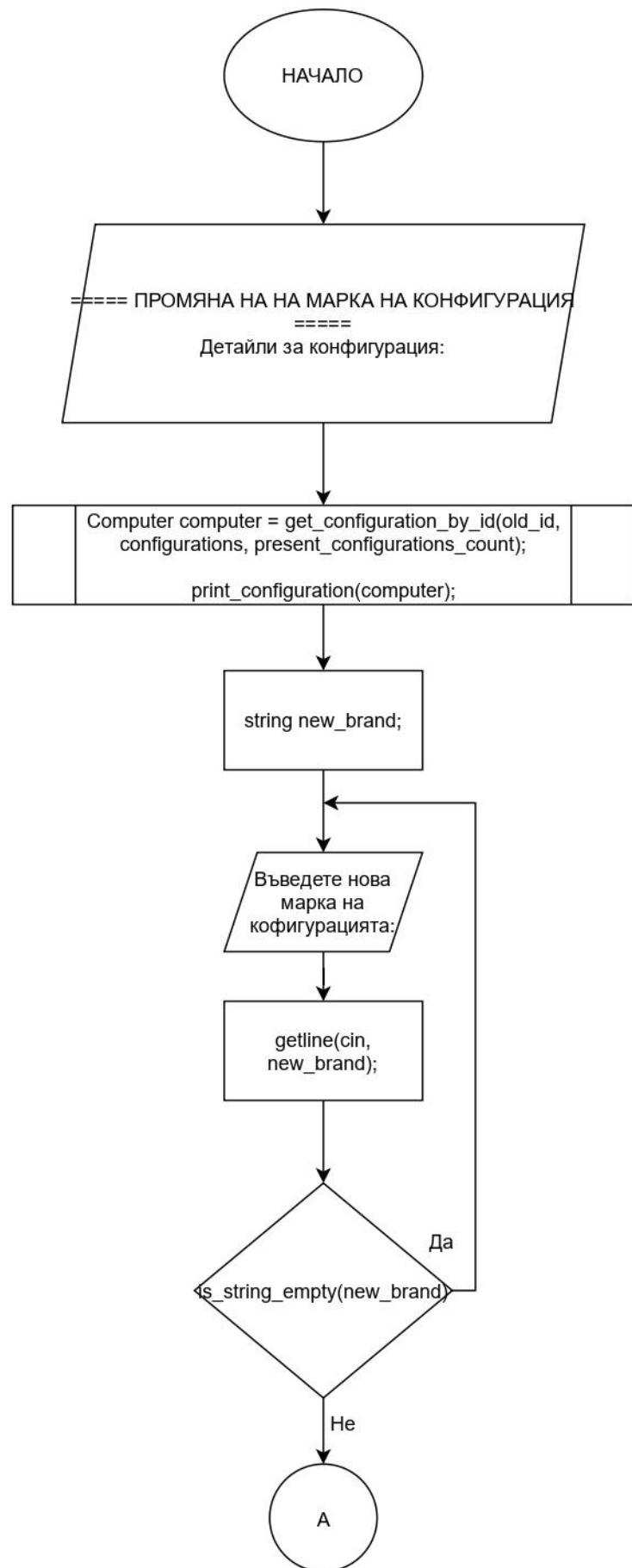


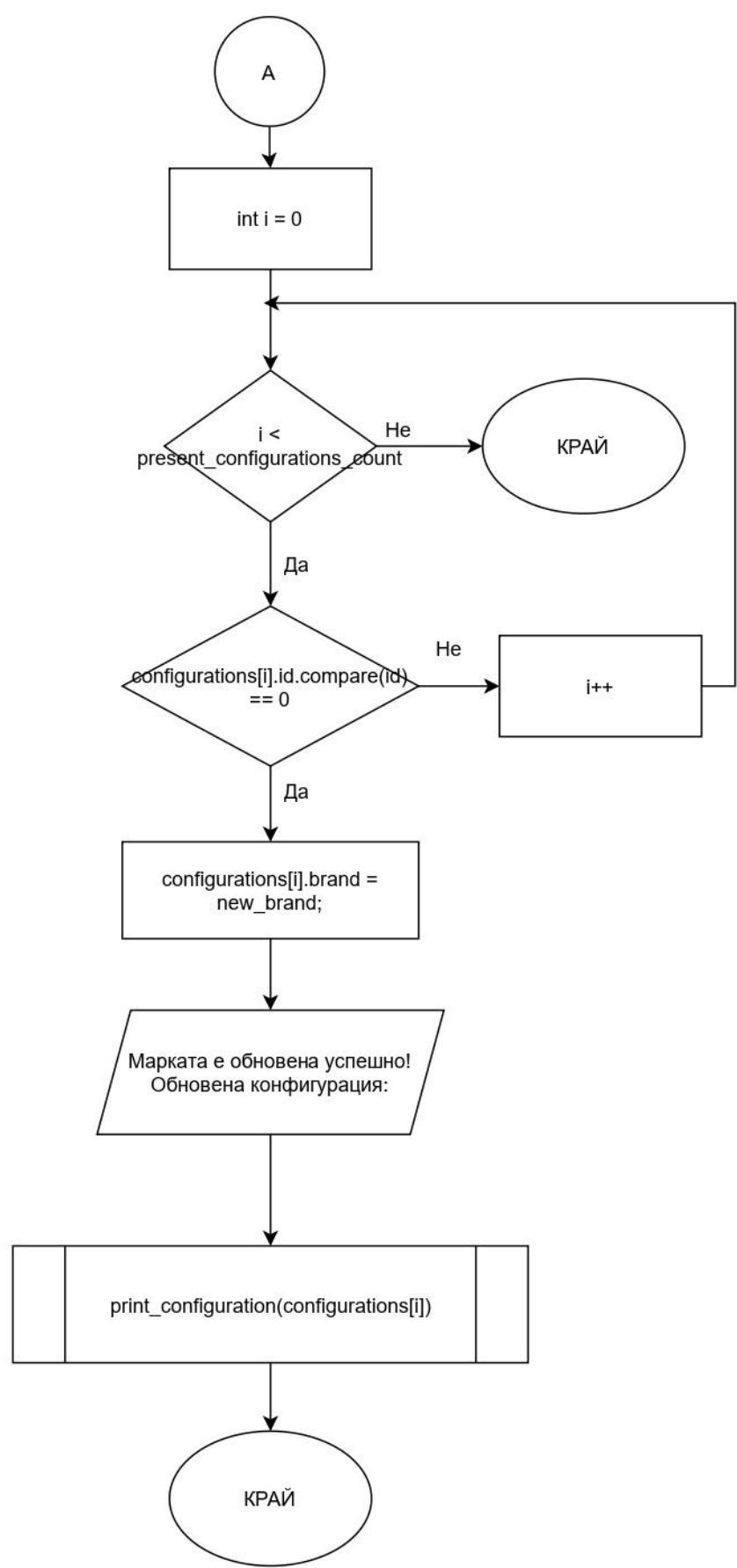
```
void update_configuration_id(string& old_id, Computer configurations[], int present_configurations_count);
```



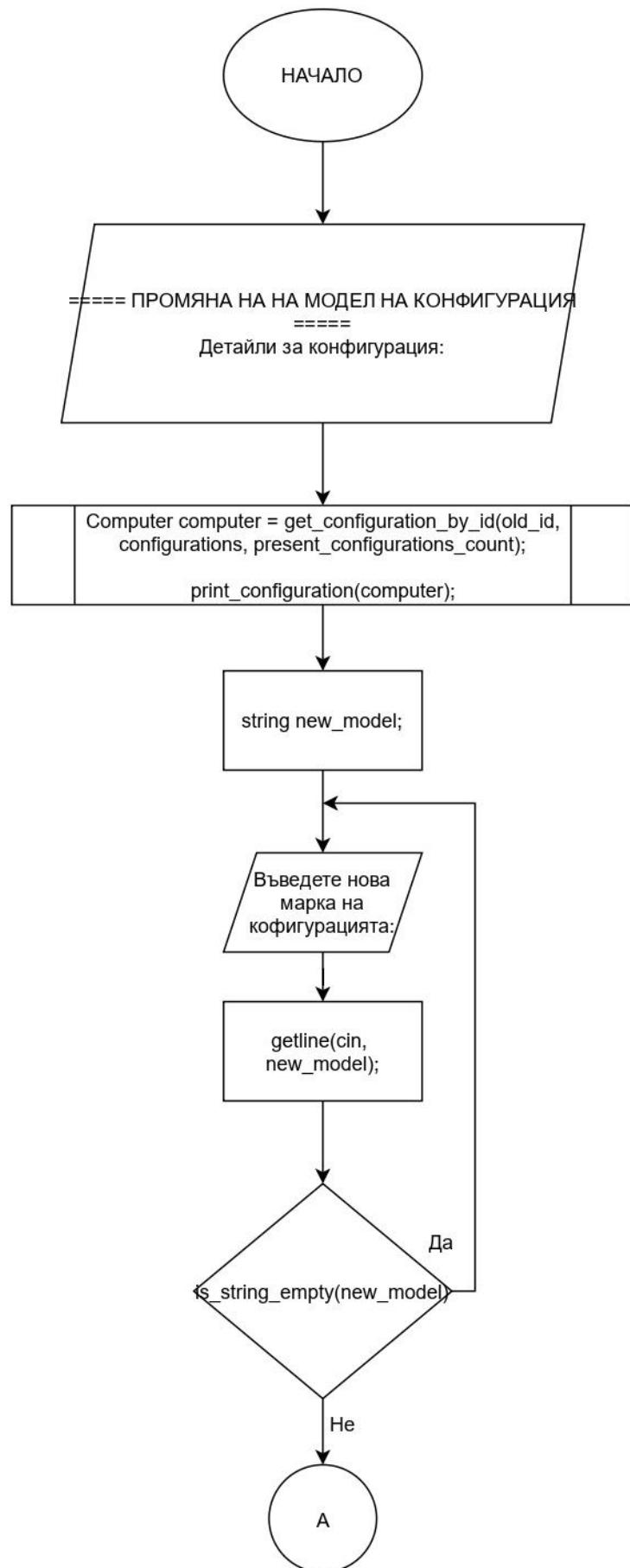


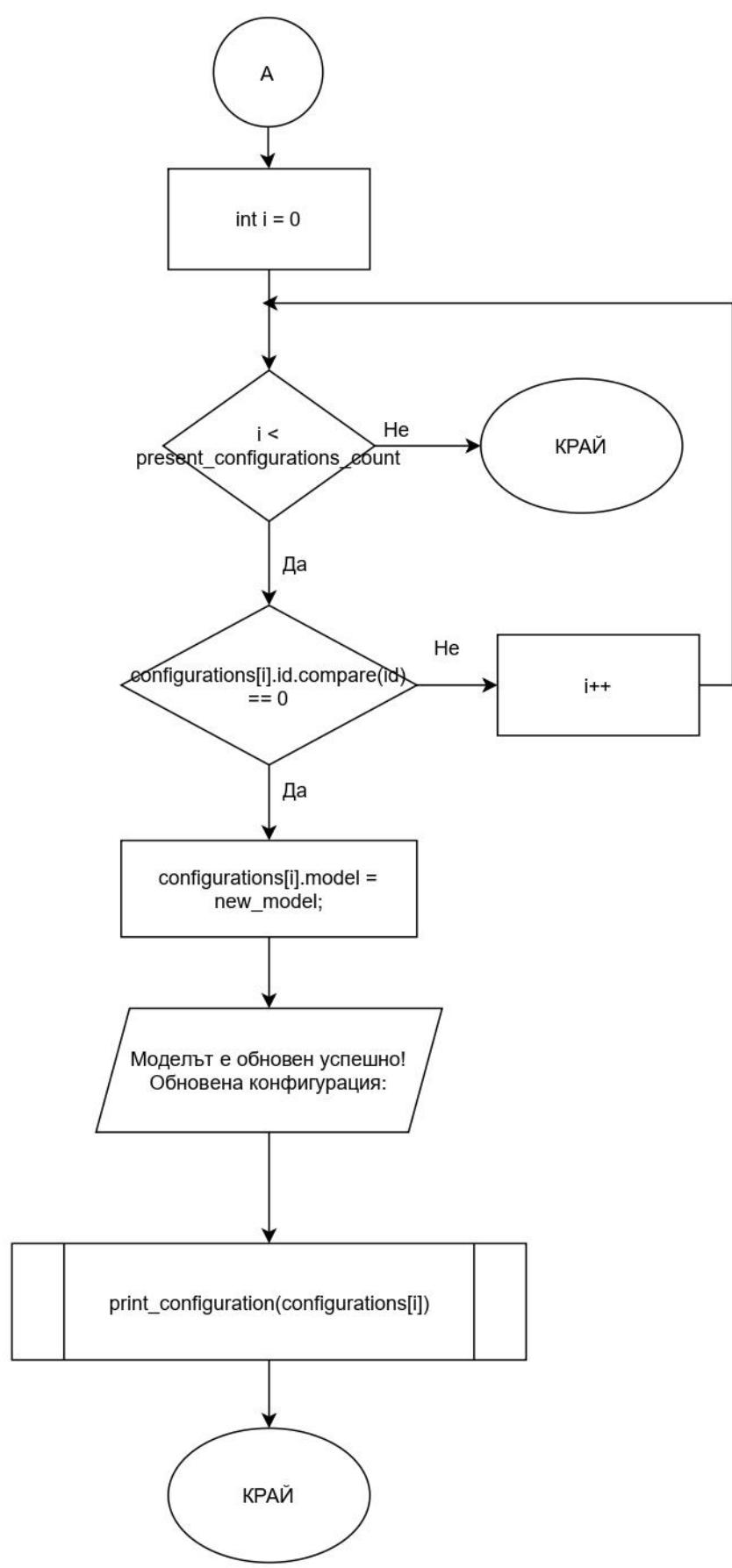
```
void update_configuration_brand(string& id, Computer configurations[], int present_configurations_count);
```



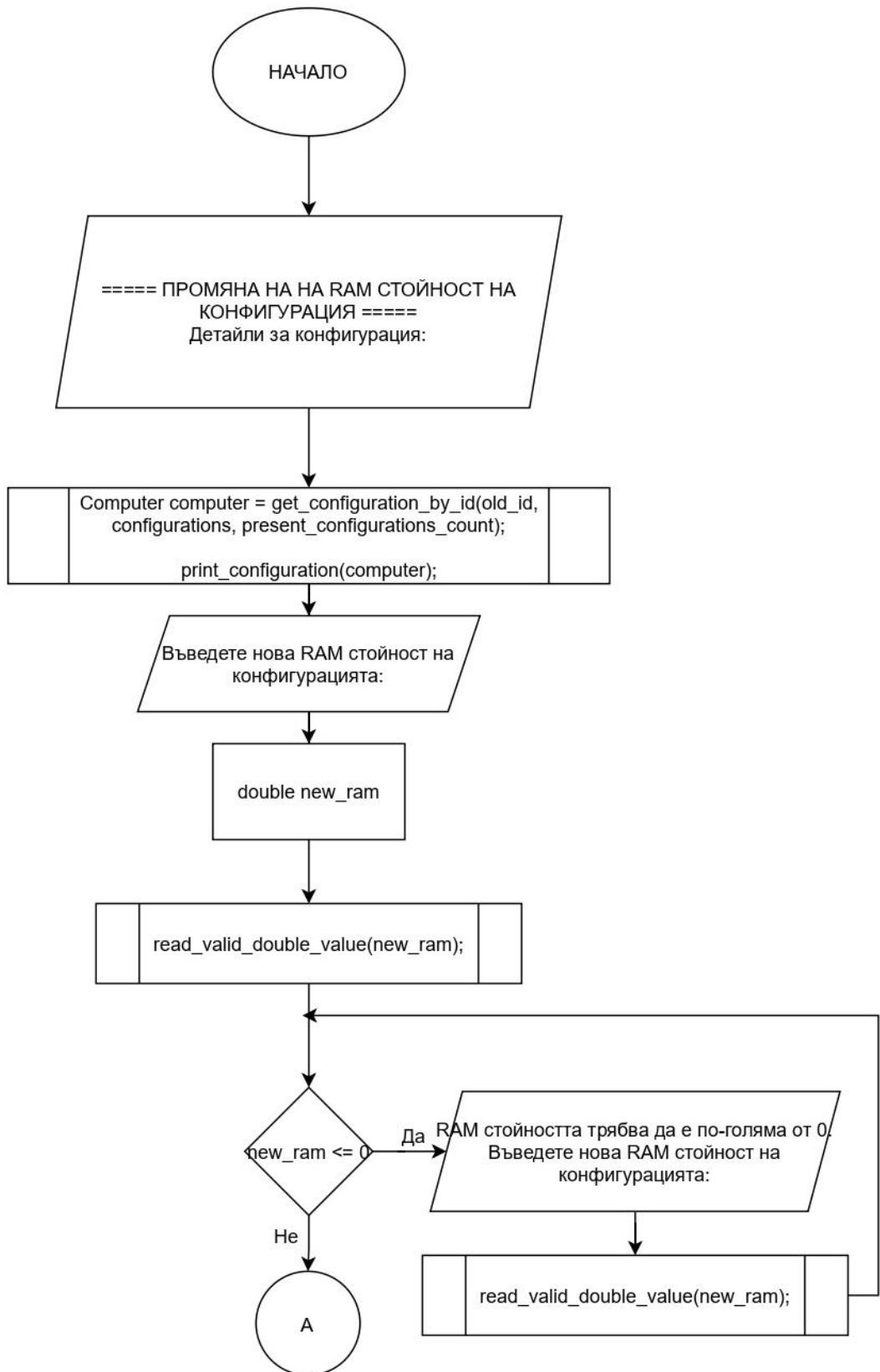


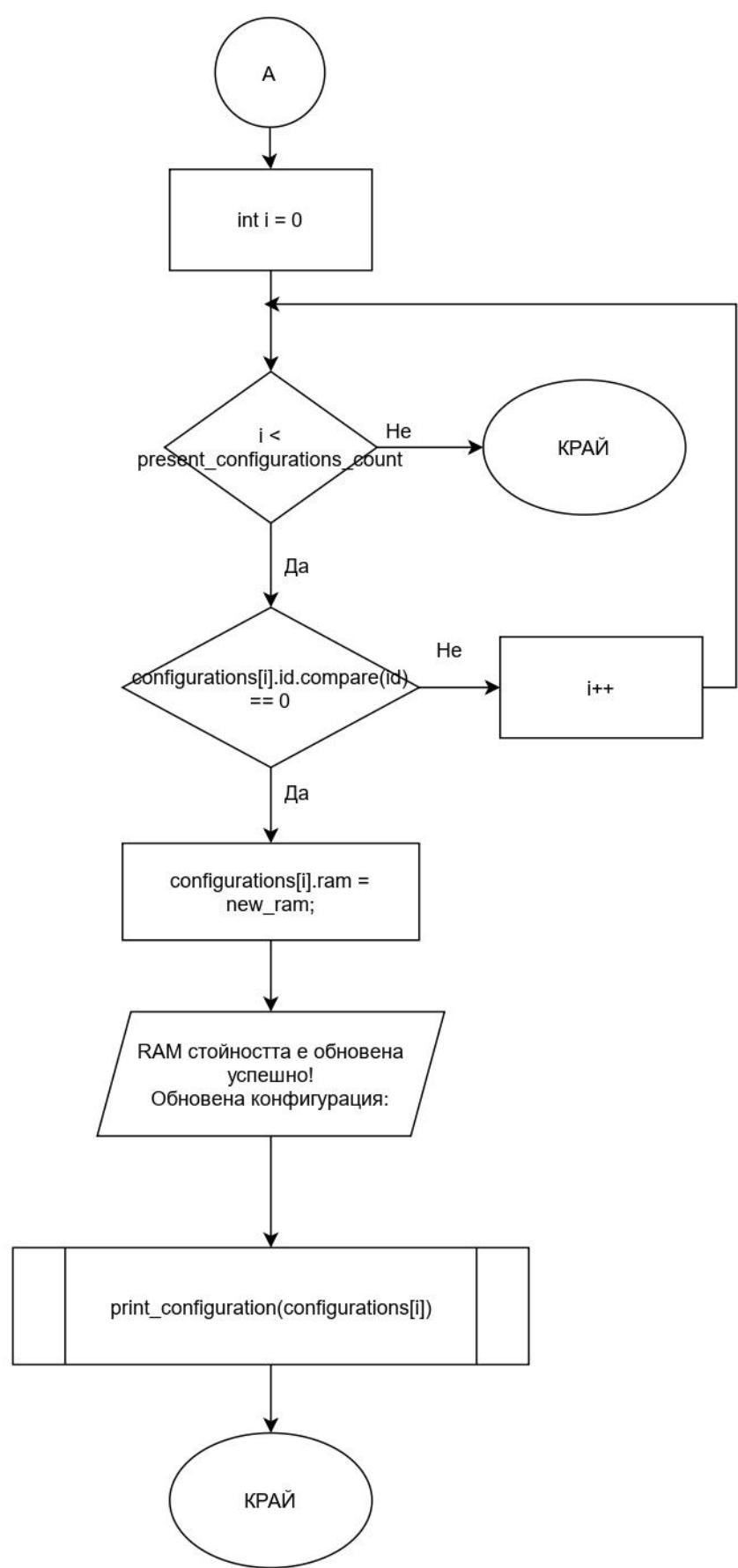
```
void update_configuration_model(string& id, Computer configurations[], int present_configurations_count);
```



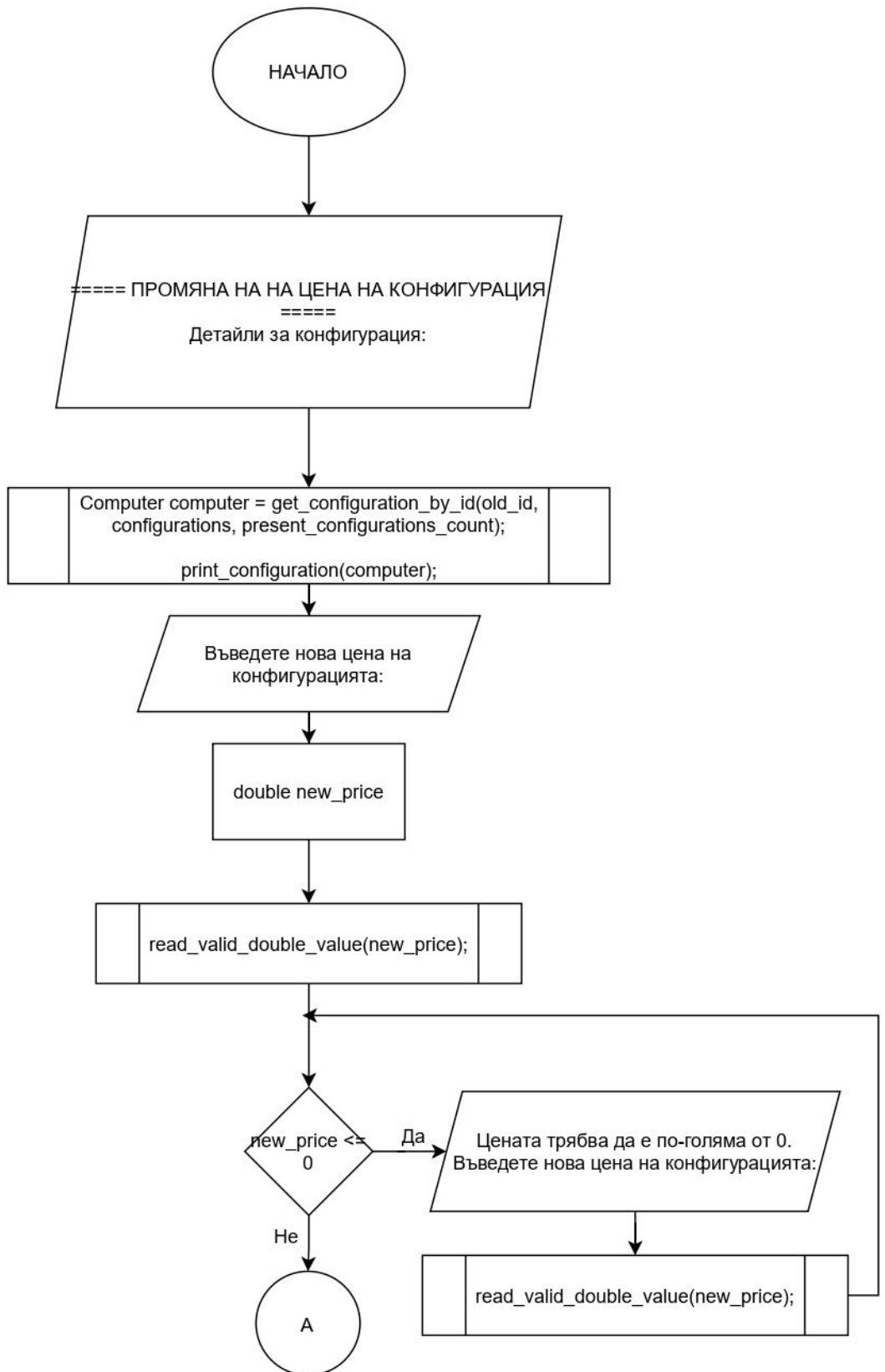


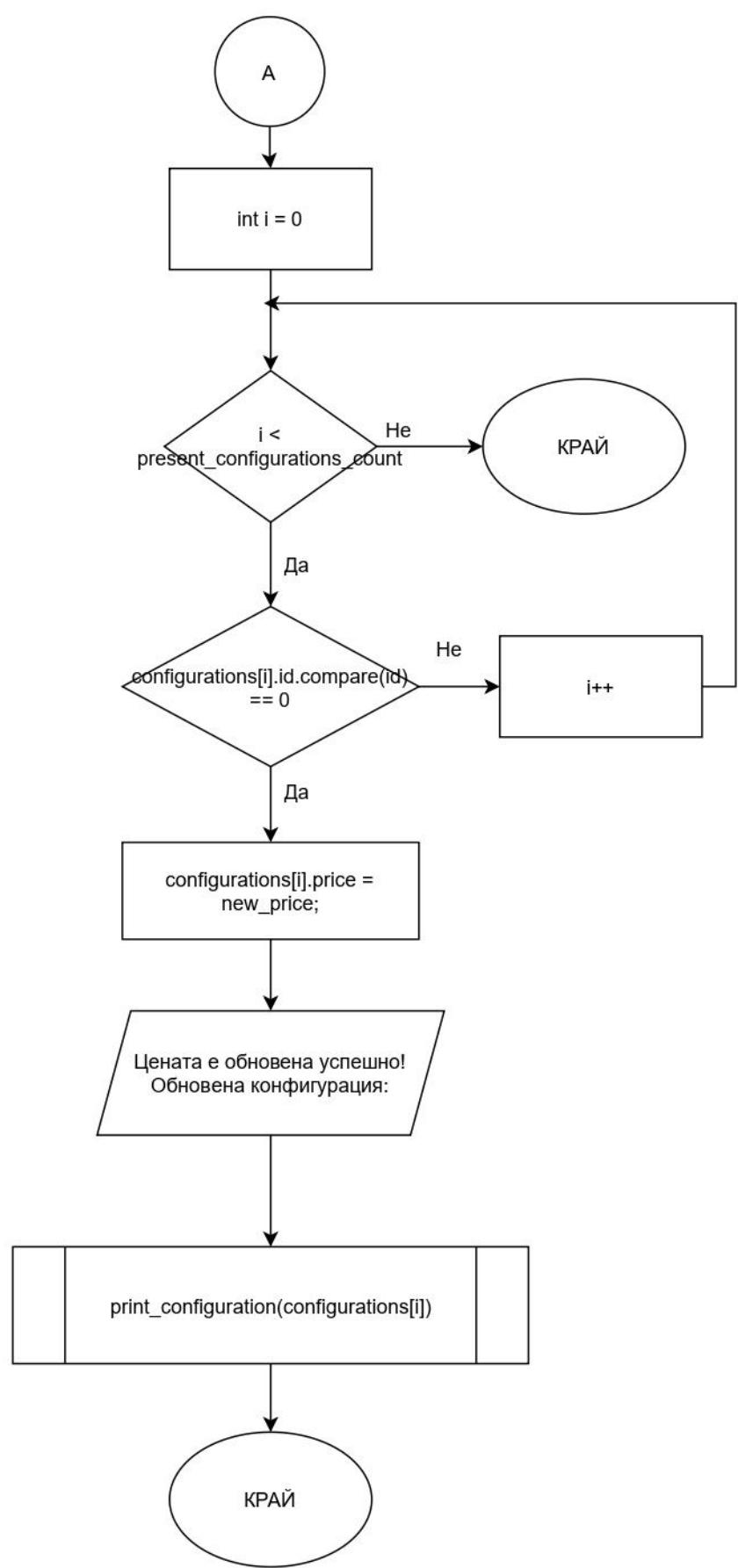
```
void update_configuration_ram(string& id, Computer configurations[], int present_configurations_count);
```



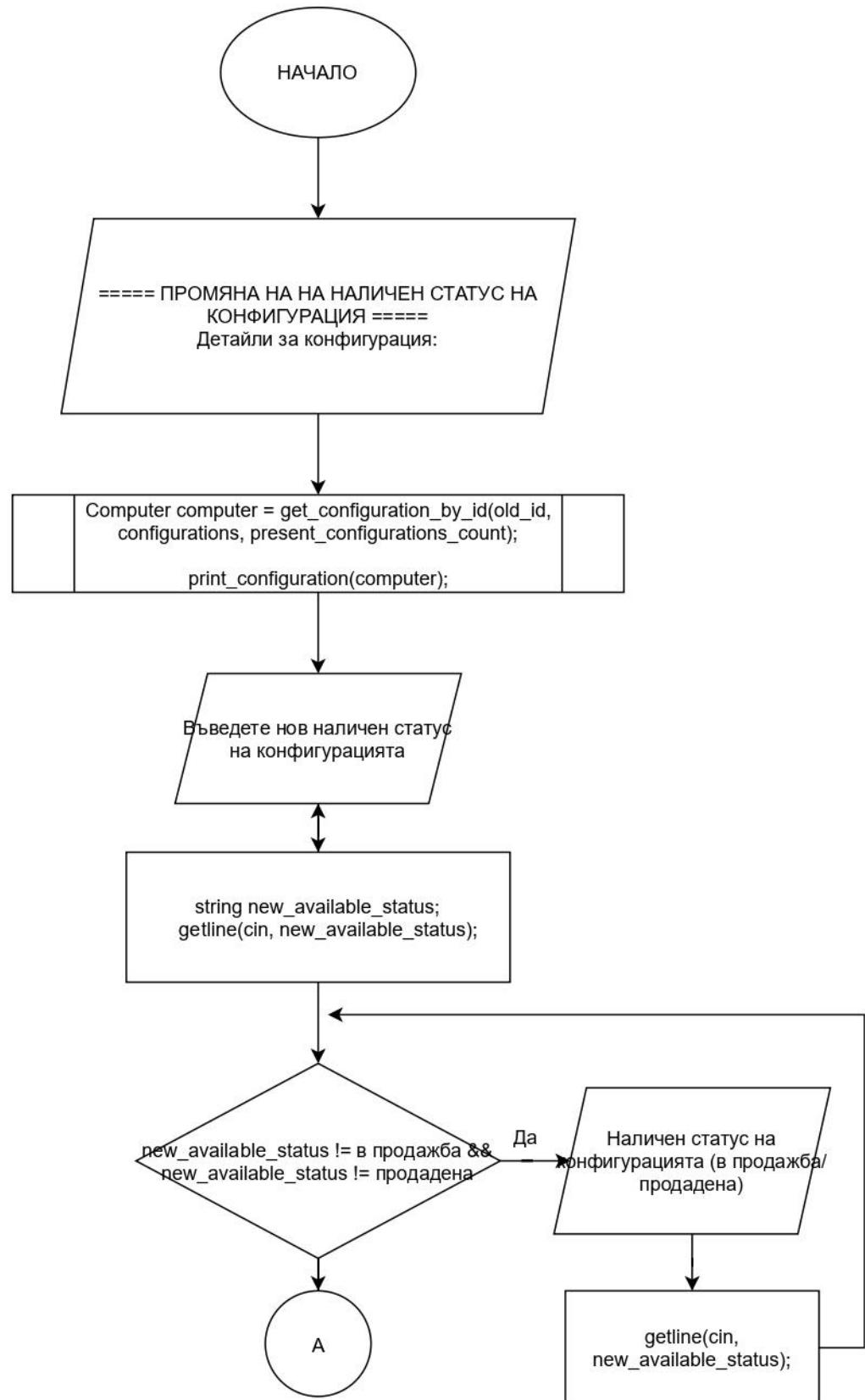


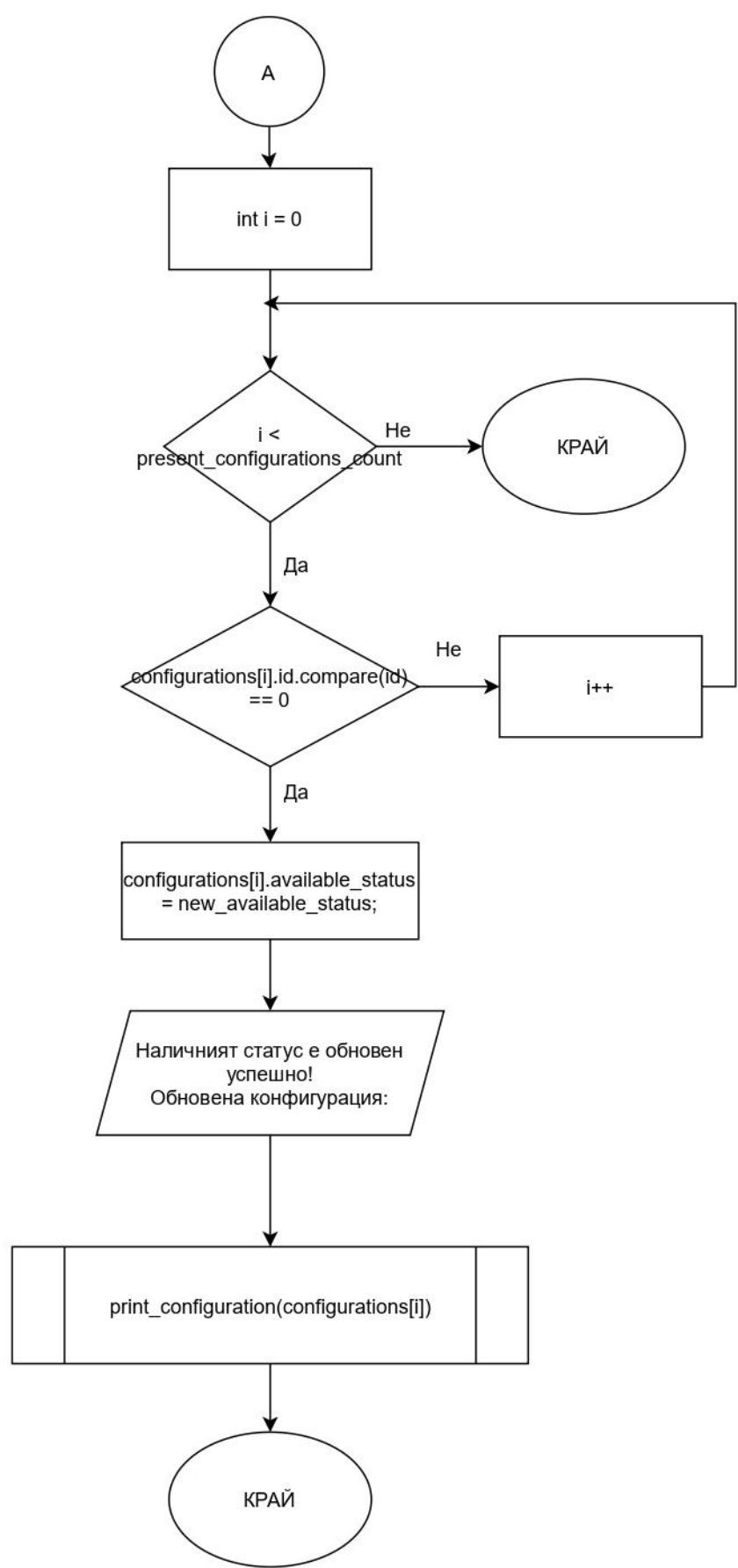
```
void update_configuration_price(string& id, Computer configurations[], int present_configurations_count);
```



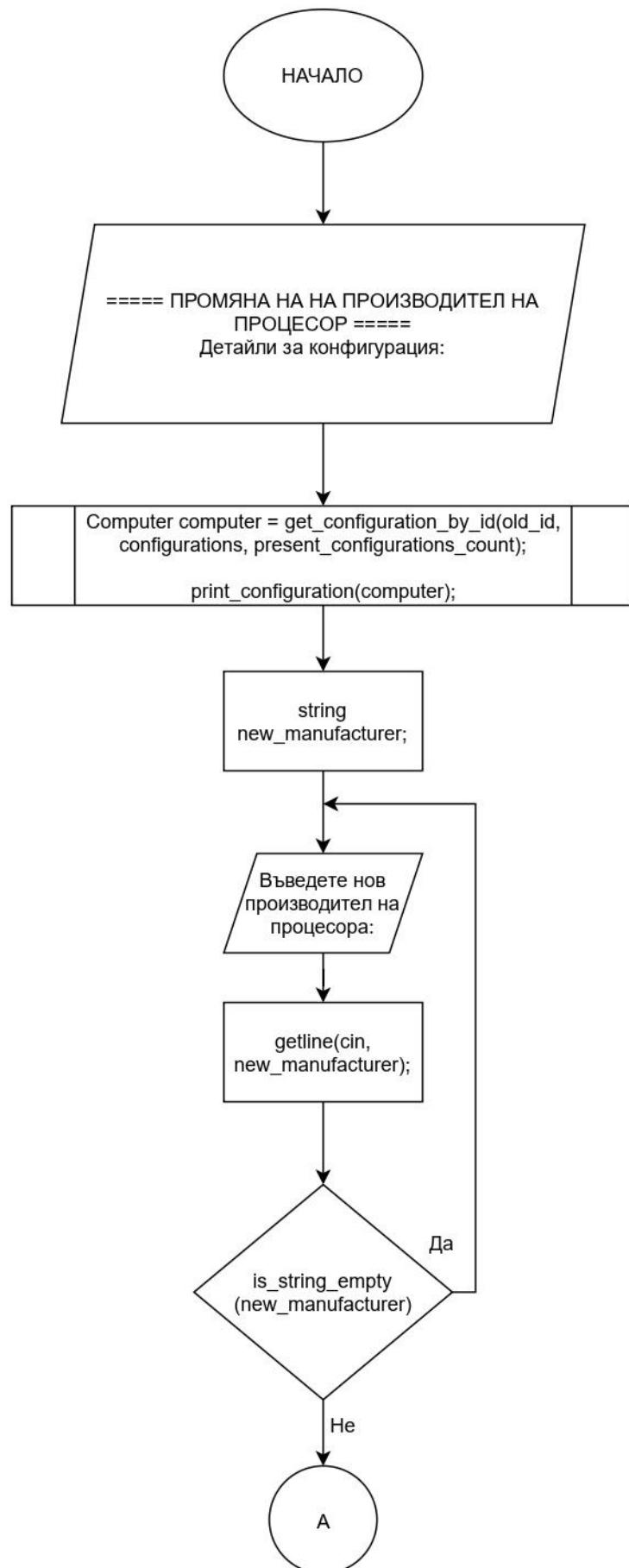


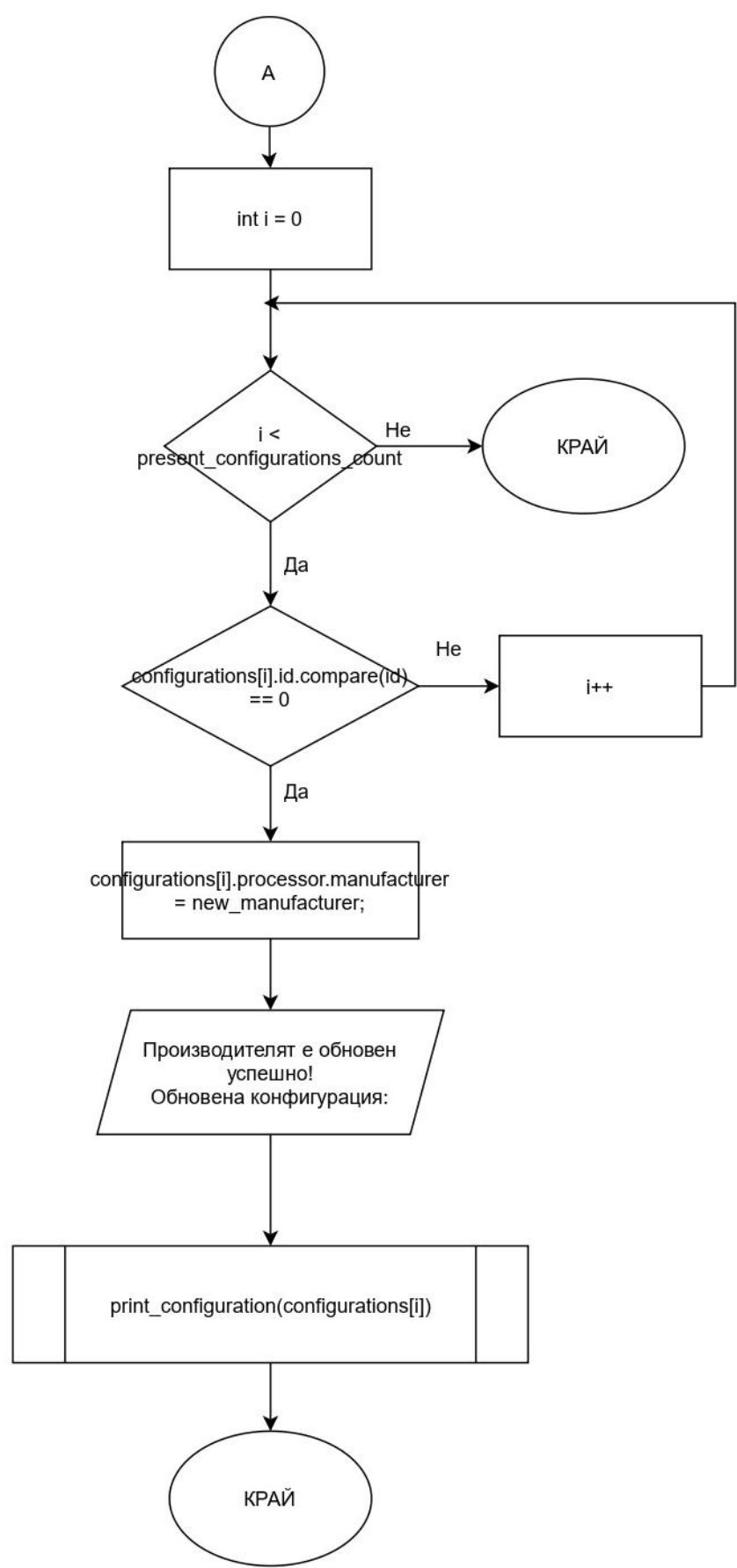
```
void update_configuration_status(string& id, Computer configurations[], int present_configurations_count);
```



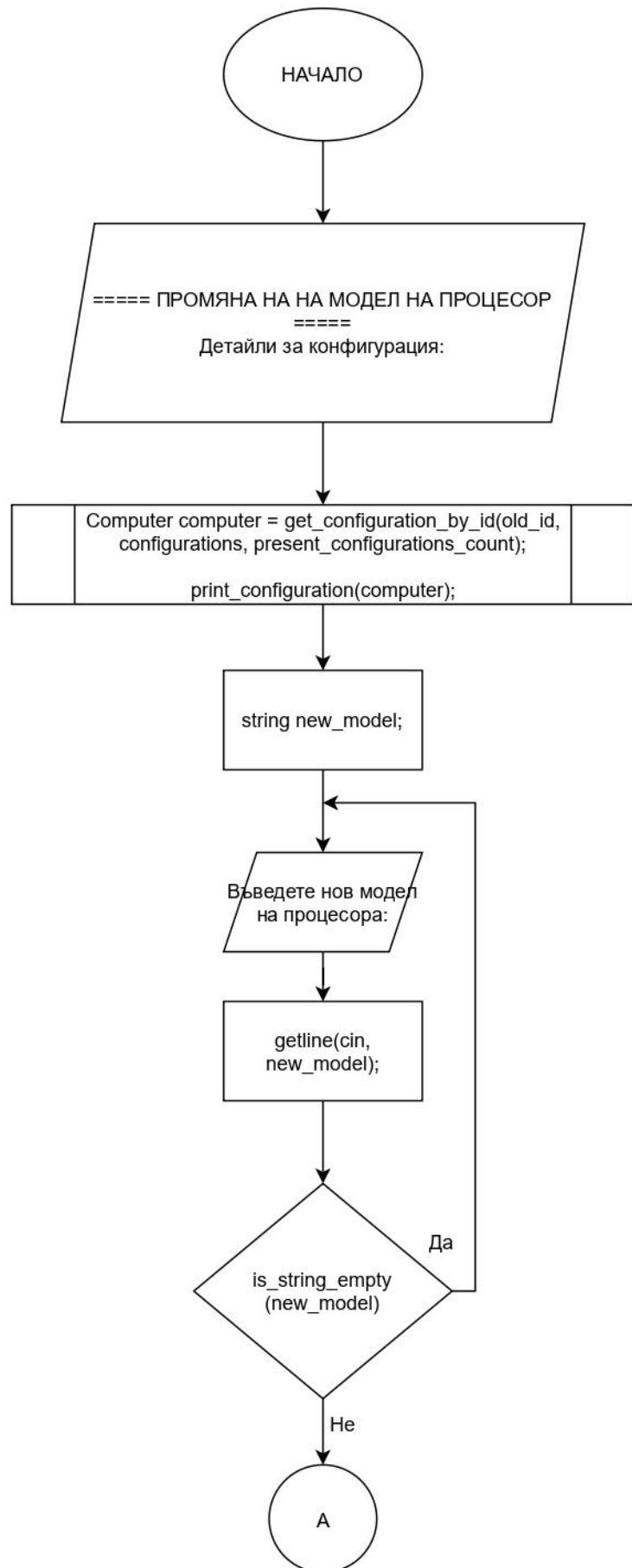


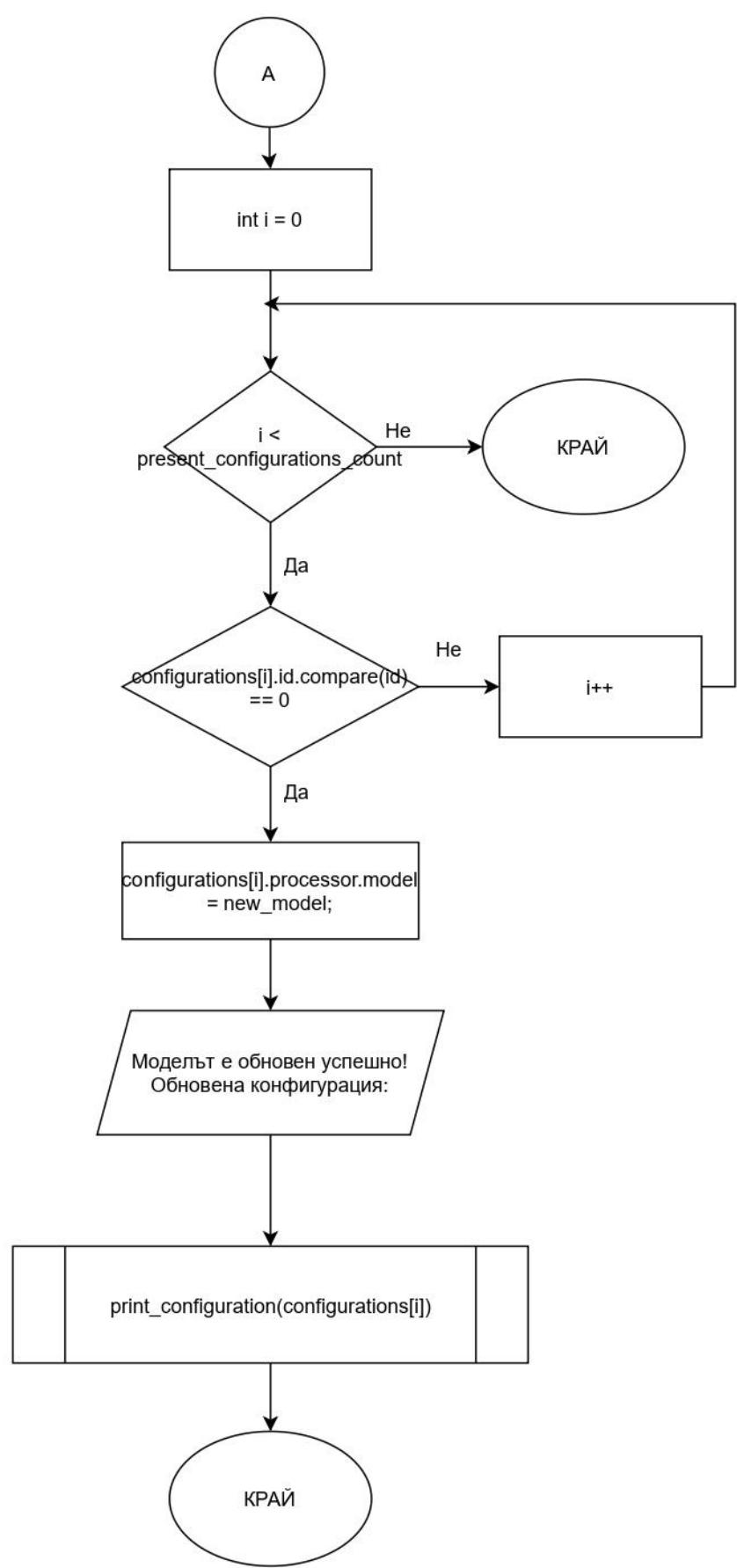
```
void update_processor_manufacturer(string& id, Computer configurations[], int present_configurations_count);
```



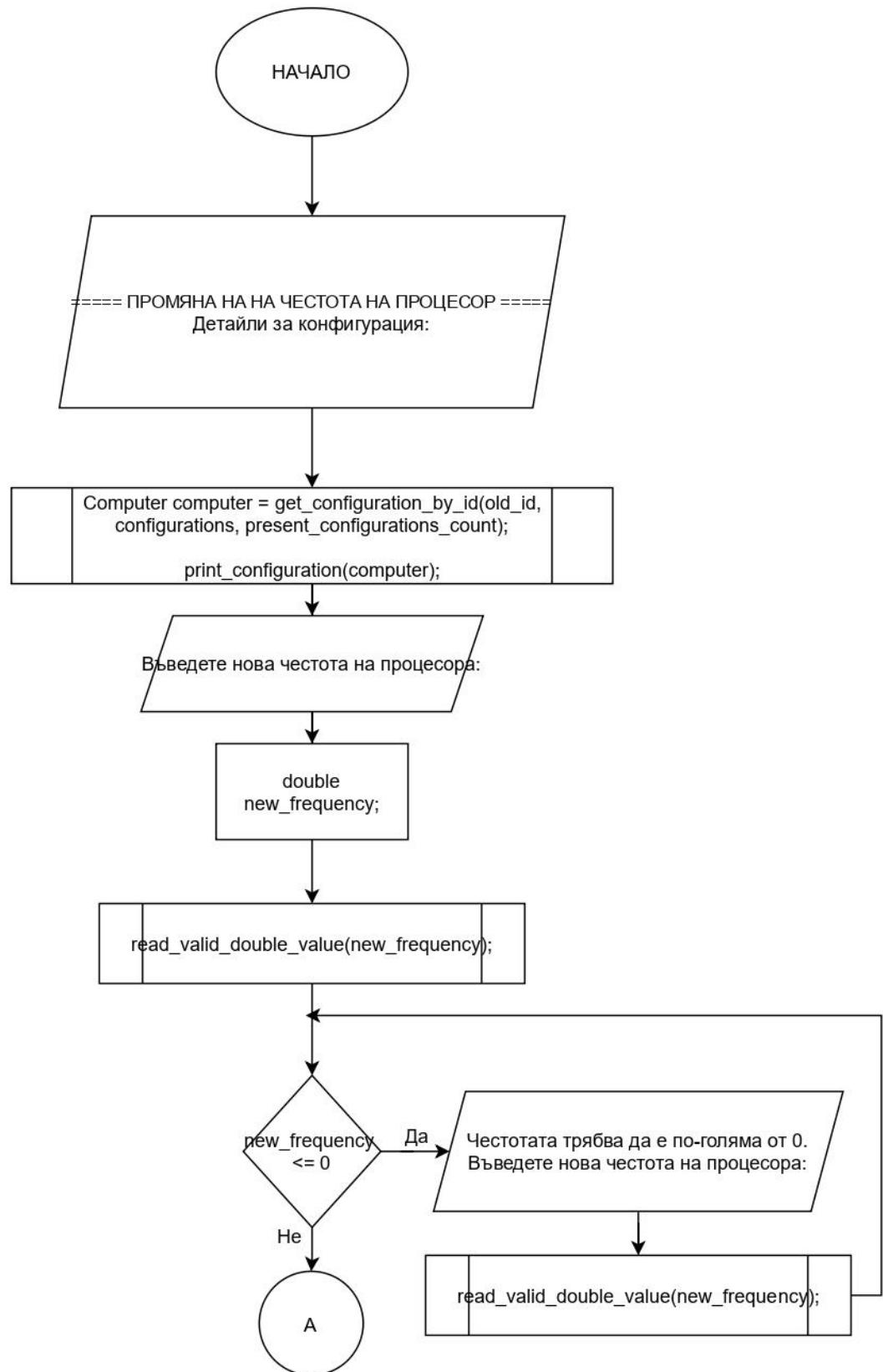


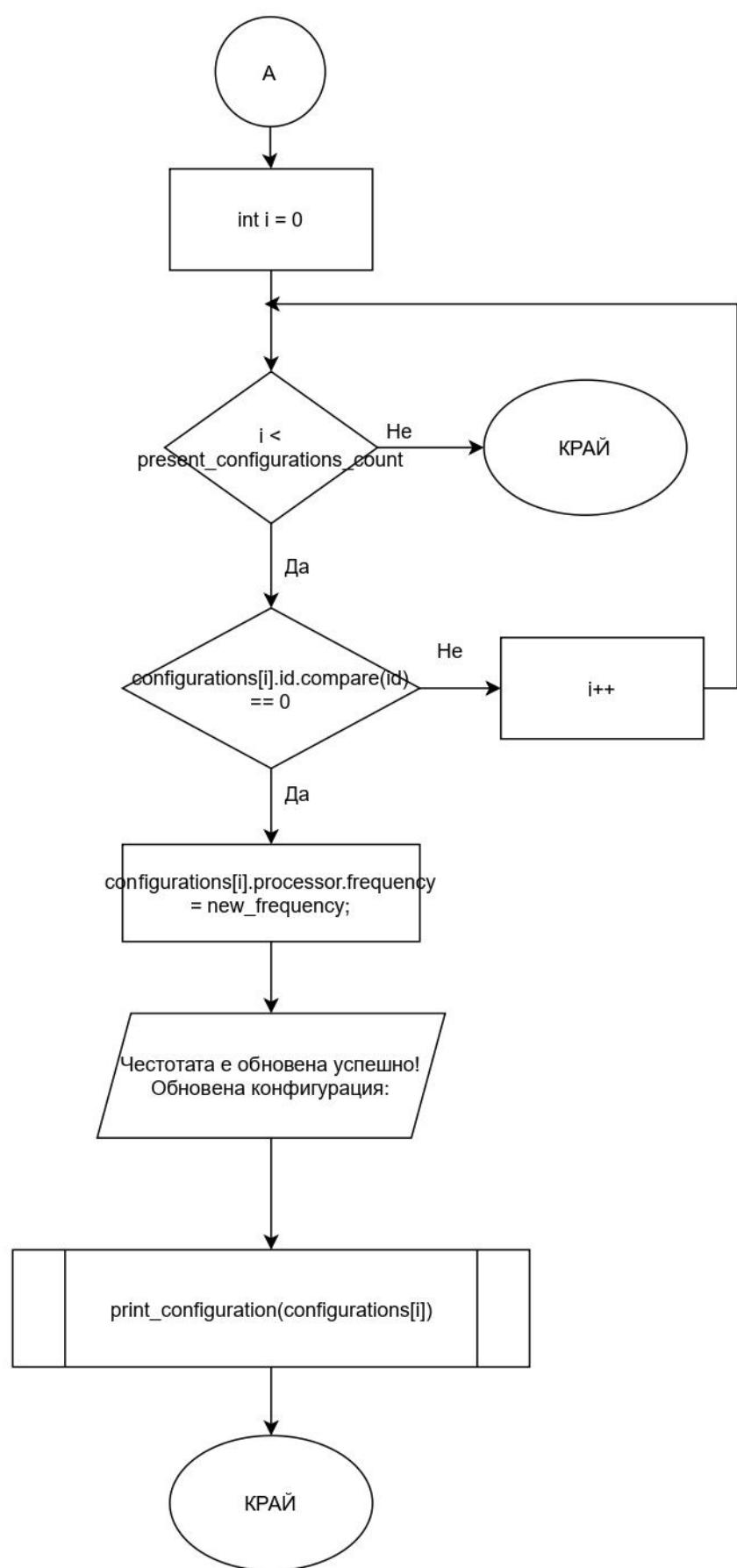
```
void update_processor_model(string& id, Computer configurations[], int present_configurations_count);
```



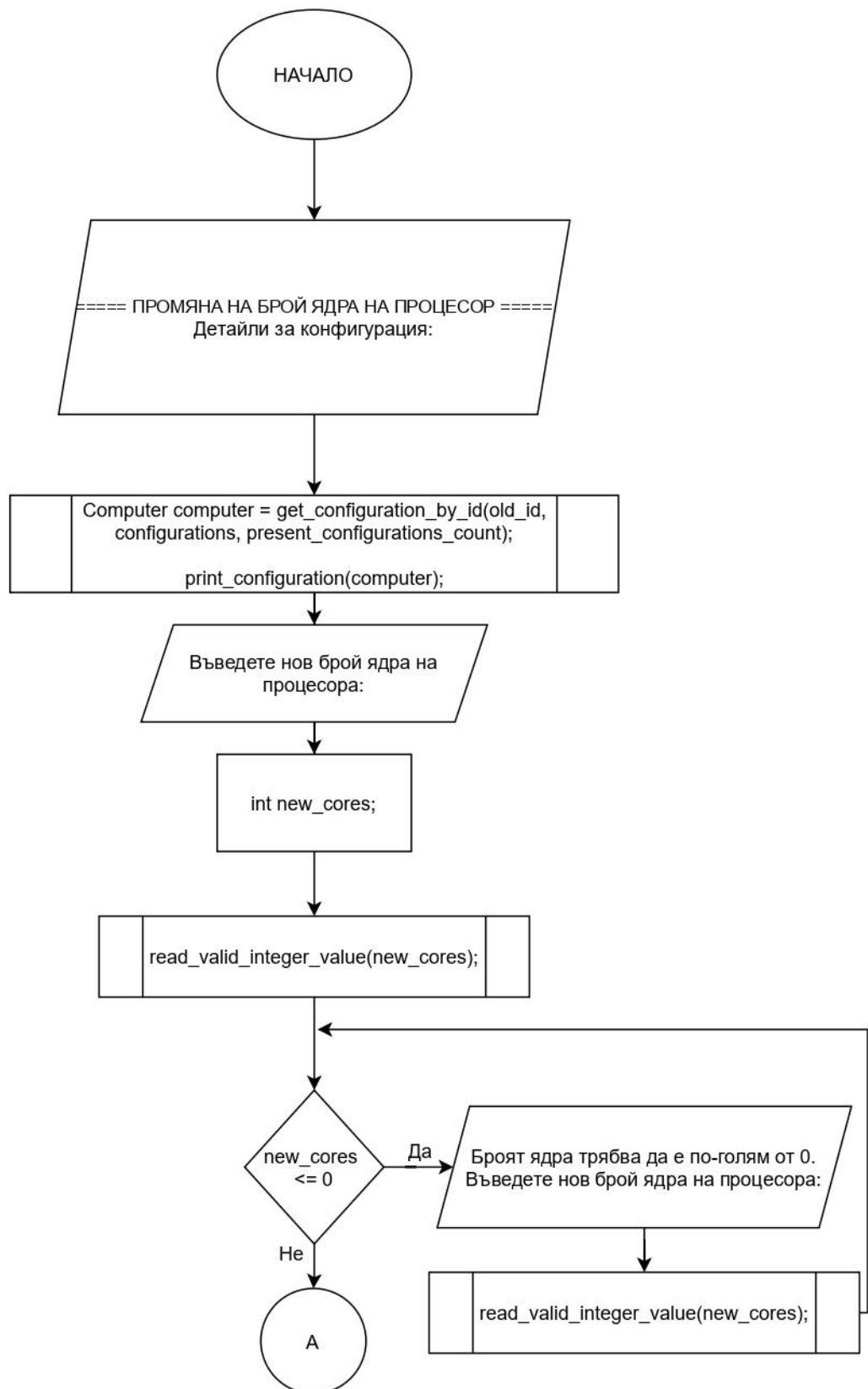


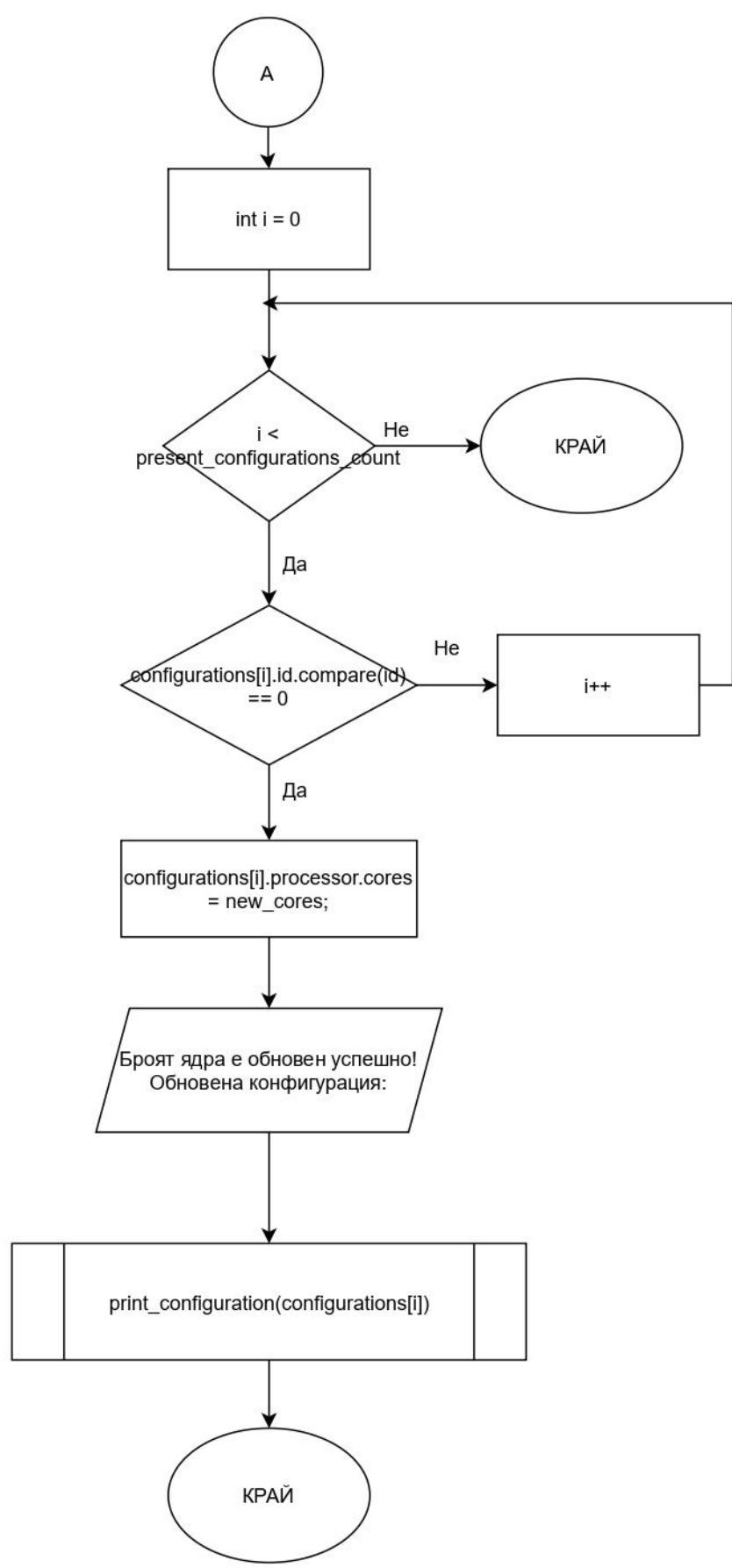
```
void update_configuration_frequency(string& id, Computer configurations[], int present_configurations_count);
```



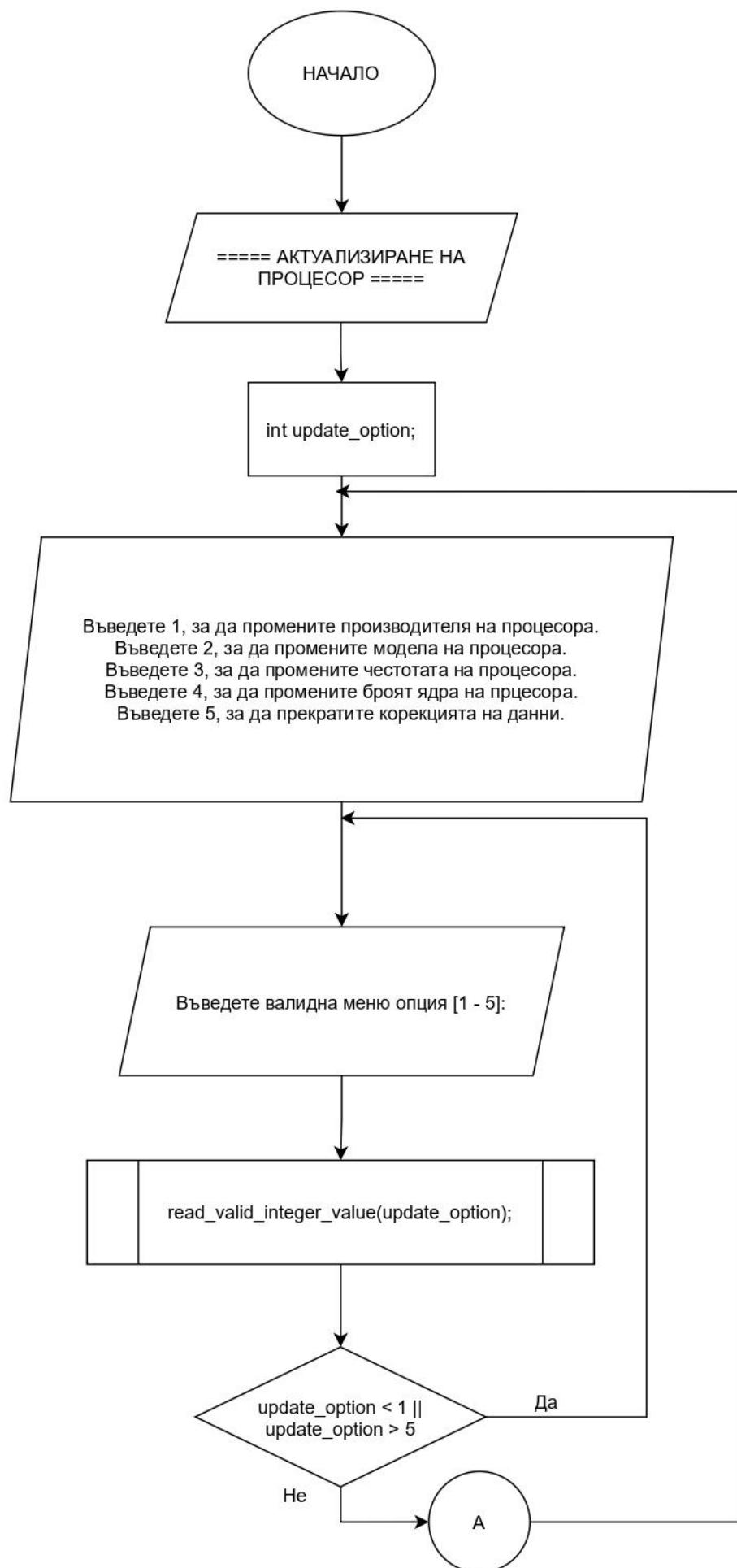


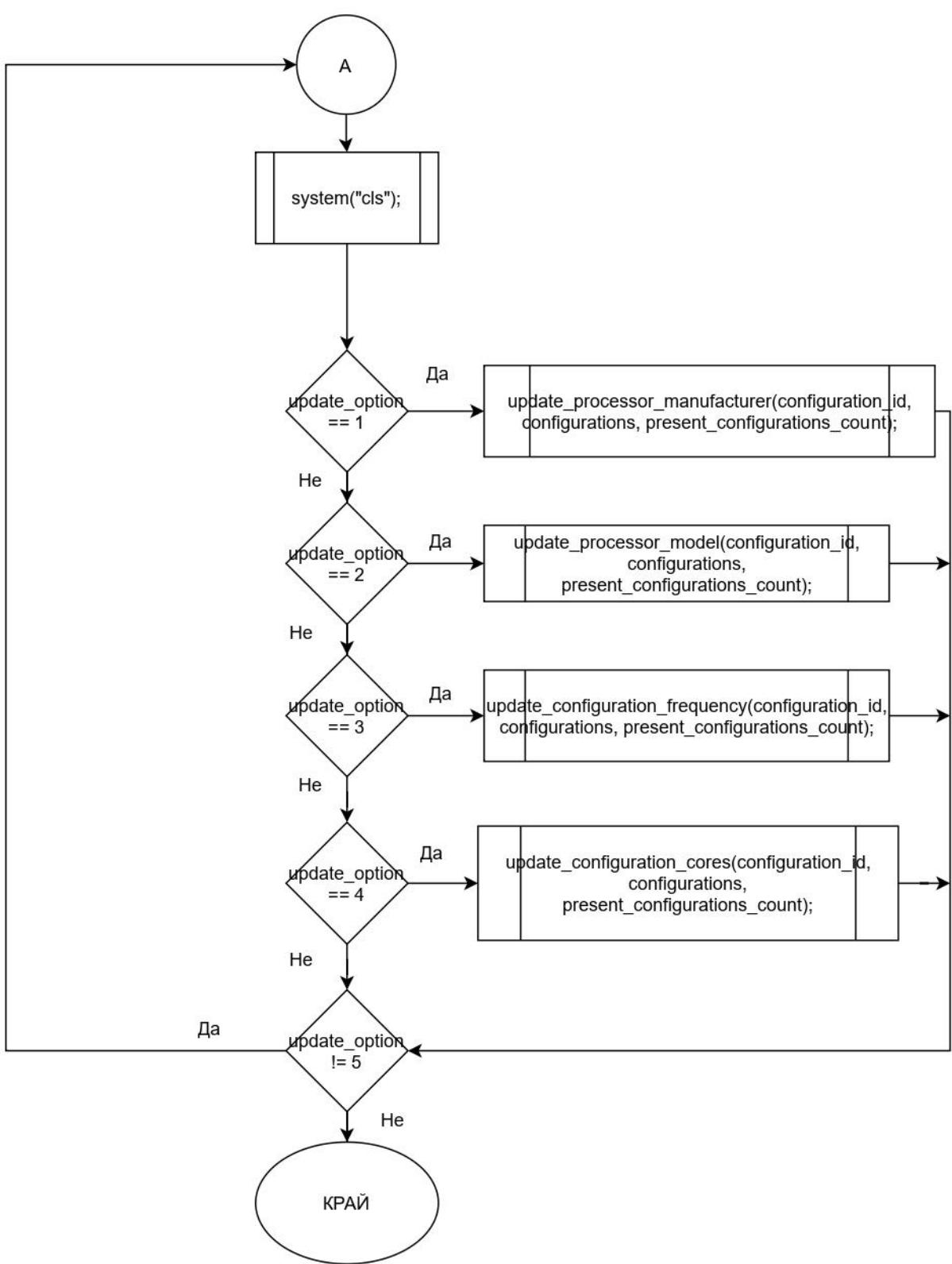
```
void update_configuration_cores(string& id, Computer configurations[], int present_configurations_count);
```



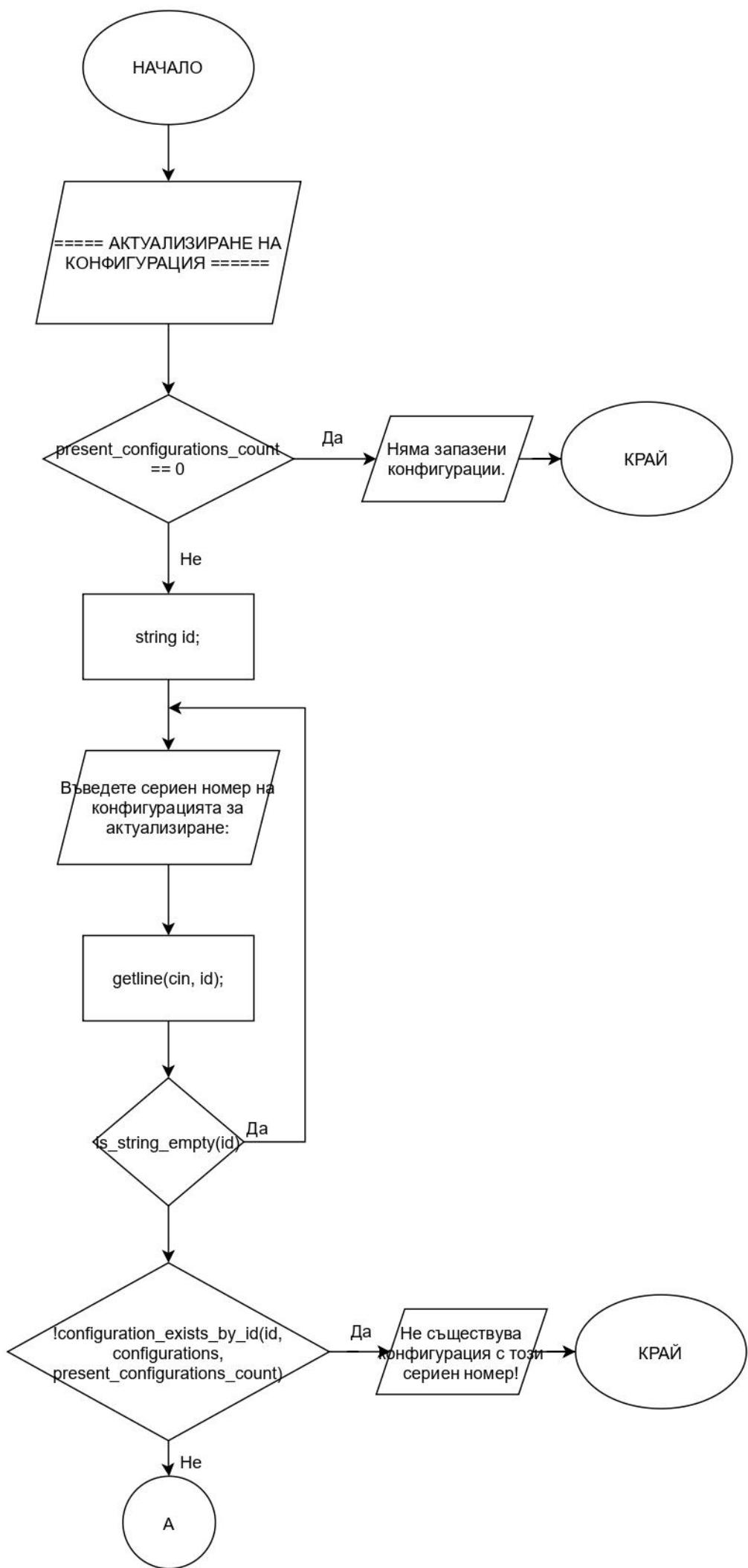


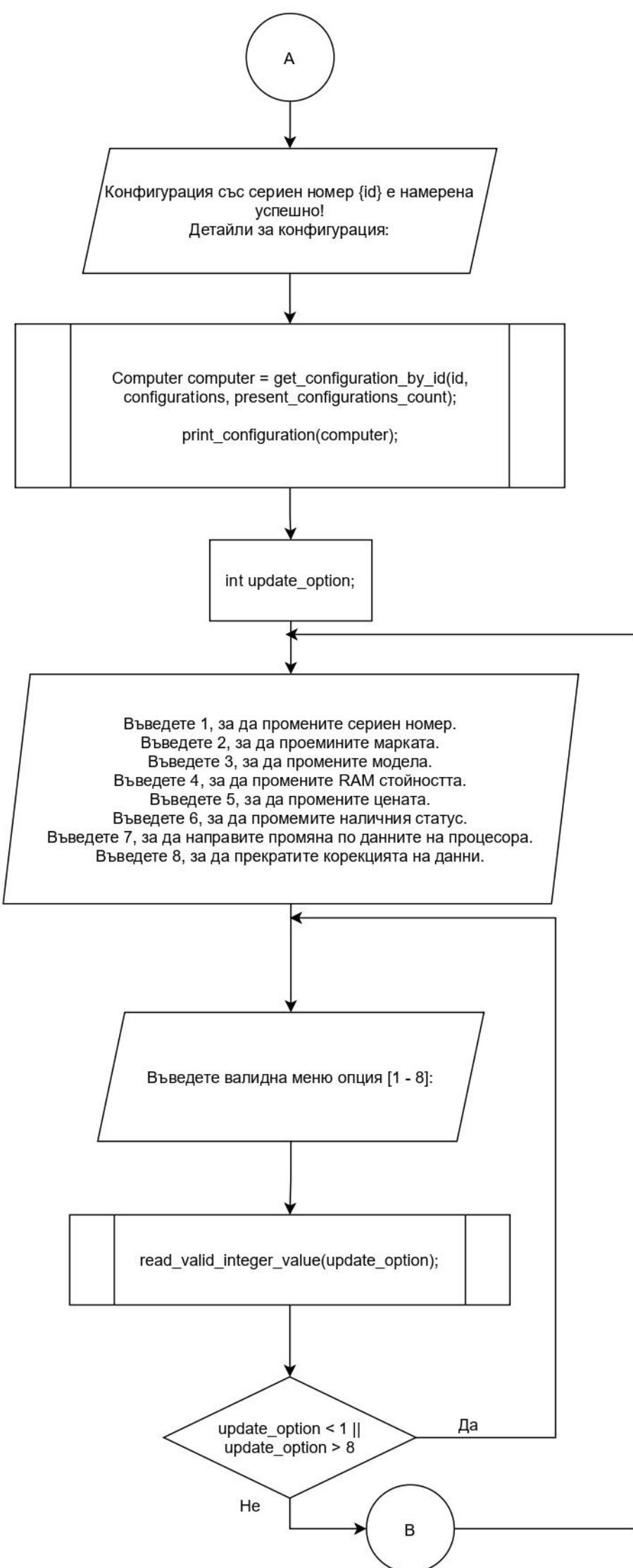
```
void update_processor(string& configuration_id, Computer configurations[], int present_configurations_count);
```

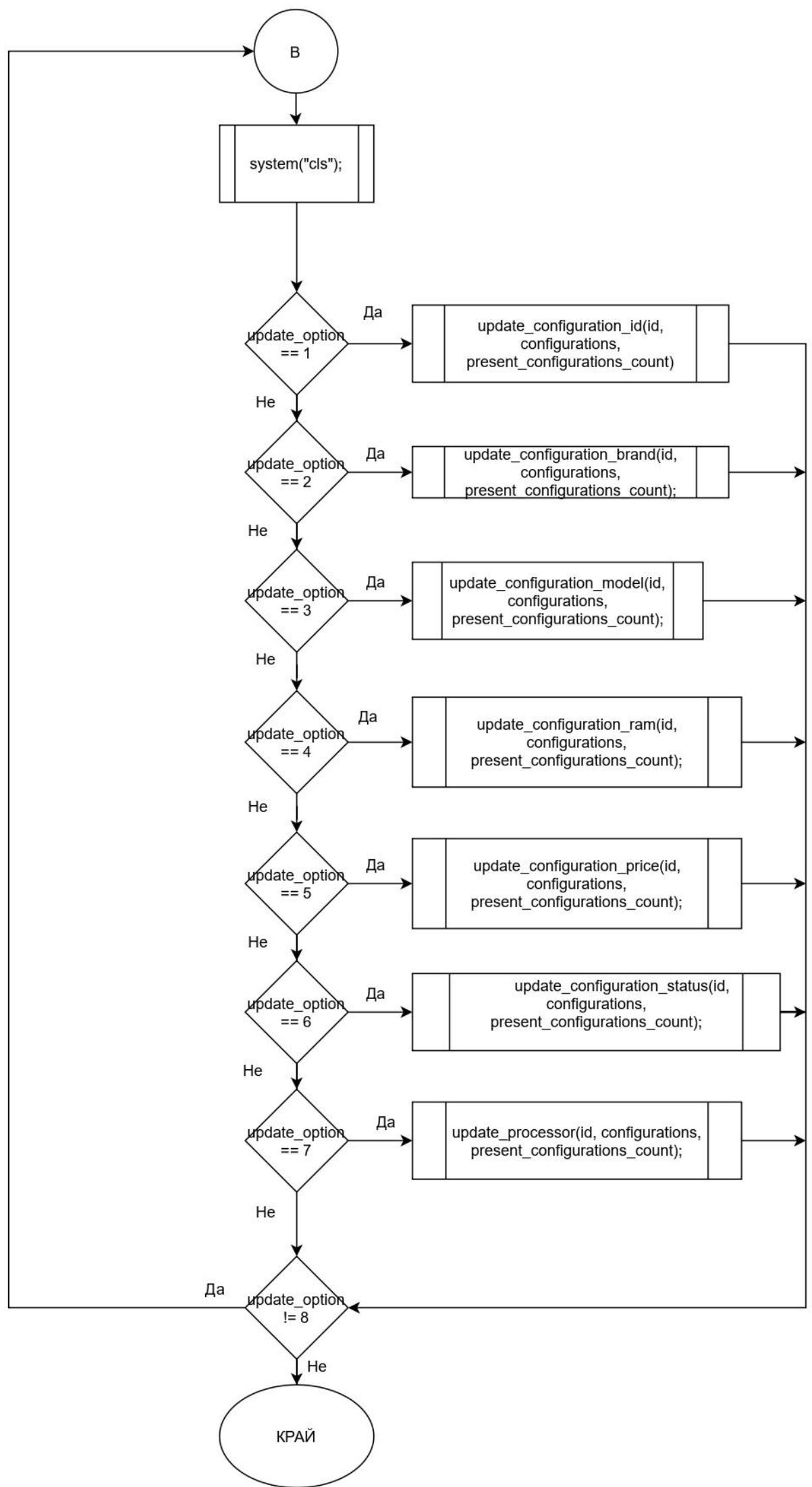




```
void update_configuration(Computer configurations[], int present_configurations_count);
```







Екранни снимки с примерни входни и изходни данни

- a. Изглед при опит на редакция на конфигурация с невалиден сериен номер

```
===== АКТУАЛИЗИРАНЕ НА КОНФИГУРАЦИЯ =====  
  
Въведете сериен номер на конфигурацията за актуализиране: no existing  
  
Не съществува конфигурация с този сериен номер!
```

- b. Излгед при опит на редакция на продадена конфигурация

```
===== АКТУАЛИЗИРАНЕ НА КОНФИГУРАЦИЯ =====  
  
Въведете сериен номер на конфигурацията за актуализиране: 15-58-64-54-29-C1  
  
Конфигурацията е продадена и не може да бъде редактирана!
```

- c. Изглед при успешно обновяване на марката на конфигурацията

```
===== АКТУАЛИЗИРАНЕ НА КОНФИГУРАЦИЯ =====  
  
Въведете сериен номер на конфигурацията за актуализиране: 2  
  
Конфигурация със сериен номер 2 е намерена успешно!  
  
Детайли за конфигурация:  
  
Сериен номер: 2  
Марка: Acer  
Модел: Nitro  
RAM памет: 16,00 GB  
Процесор:  
    Производител: Intel  
    Модел: i7  
    Тактова честота: 3,00 GHz  
    Брой ядра: 6  
Цена: 2200,00 лв.  
Наличен статус: в продажба  
  
Въведете 1, за да промените сериен номер.  
Въведете 2, за да проемините марката.  
Въведете 3, за да промените модела.  
Въведете 4, за да промените RAM стойността.  
Въведете 5, за да промените цената.  
Въведете 6, за да промените наличния статус.  
Въведете 7, за да направите промяна по данните на процесора.  
Въведете 8, за да прекратите корекцията на данни.  
Въведете валидна меню опция [1 - 8]:
```

Въведете нова марка на конфигурацията: Asus

Марката е обновена успешно!

Обновена конфигурация:

Сериен номер: 2

Марка: Asus

Модел: Nitro

RAM памет: 16,00 GB

Процесор:

Производител: Intel

Модел: i7

Тактова честота: 3,00 GHz

Брой ядра: 6

Цена: 2200,00 лв.

Наличен статус: в продажба

Въведете 1, за да промените сериен номер.

Въведете 2, за да проемините марката.

Въведете 3, за да промените модела.

Въведете 4, за да промените RAM стойността.

Въведете 5, за да промените цената.

Въведете 6, за да промените наличния статус.

Въведете 7, за да направите промяна по данните на процесора.

Въведете 8, за да прекратите корекцията на данни.

Въведете валидна меню опция [1 - 8]:

E. Продажба на компютърна конфигурация

- a. Въвежда се сериен номер и цената на конфигурацията
- b. Въвеждат се характеристики и след това се избира конфигурацията за продажба

```
void change_configuration_availability_status_to_sold(string& id, Computer configurations[], int present_configurations_count);

void sell_configuration_by_id(Computer configurations[], int present_configurations_count);

void read_processor_selling_data(string& manufacturer, string& model, string& frequency, string& cores);

void read_computer_selling_data(string& brand, string& model, string& ram, string& price);

void find_computers_with_requirements(Computer configurations[], int present_configurations_count, Computer found_configurations[], int& found_configurations_count, string& processor_manufacturer, string& processor_model, string& processor_frequency, string& processor_cores, string& computer_brand, string& computer_model, string& computer_ram, string& computer_price, int& selected_features, int& actual_features);

void sell_configuration_by_requirements(Computer configurations[], int present_configurations_count);

void sell_configuration(Computer configurations[], int present_configurations_count);
```

Функцията *change_configuration_availability_status_to_sold* се използва за отбелоязване на дадена конфигурация като продадена. Приема като параметри серииният номер на конфигурацията, масив с всички запазени конфигурации, брой запазени конфигурации. Обхождат се всички конфигурации и се проверява за съвпадение в серииният номер. След като се намери конфигурацията, наличният ѝ статус се променя на „продадена“.

Функцията *sell_configuration_by_id* се използва за осъществяване на продажба по сериен номер. Приема като параметри масив с всички запазени конфигурации и брой запазени конфигурации. Извежда информативно съобщение и приканва потребителя да въведе сериен номер на конфигурацията за продажба. Входът е валидиран с do-while цикъл и *is_string_empty*. Ако конфигурация с този сериен номер не е намерена или намерената конфигурация е продадена, на екрана се извежда съобщение и функцията прекратява изпълнението си. Ако конфигурацията е намерена успешно и е налична се извеждат детайли за нея чрез функцията *print_configuration*. След това потребителят трябва да въведе цена, с която ще бъде закупена конфигурацията. Входът за цената е валидиран с while цикъл. Ако въведената цена не е достатъчна за покупка на екрана се извежда съобщение и функцията прекратява изпълнението си. В противен случай, наличният статус на конфигурацията се променя и се извежда ресто.

Функцията *read_processor_selling_data* се използва за прочитане на желаните характеристики на процесора при осъществяване на продажба. Приема като входни параметри променливи по адрес от тип string, в които ще бъдат записани желаните характеристики на процесора.

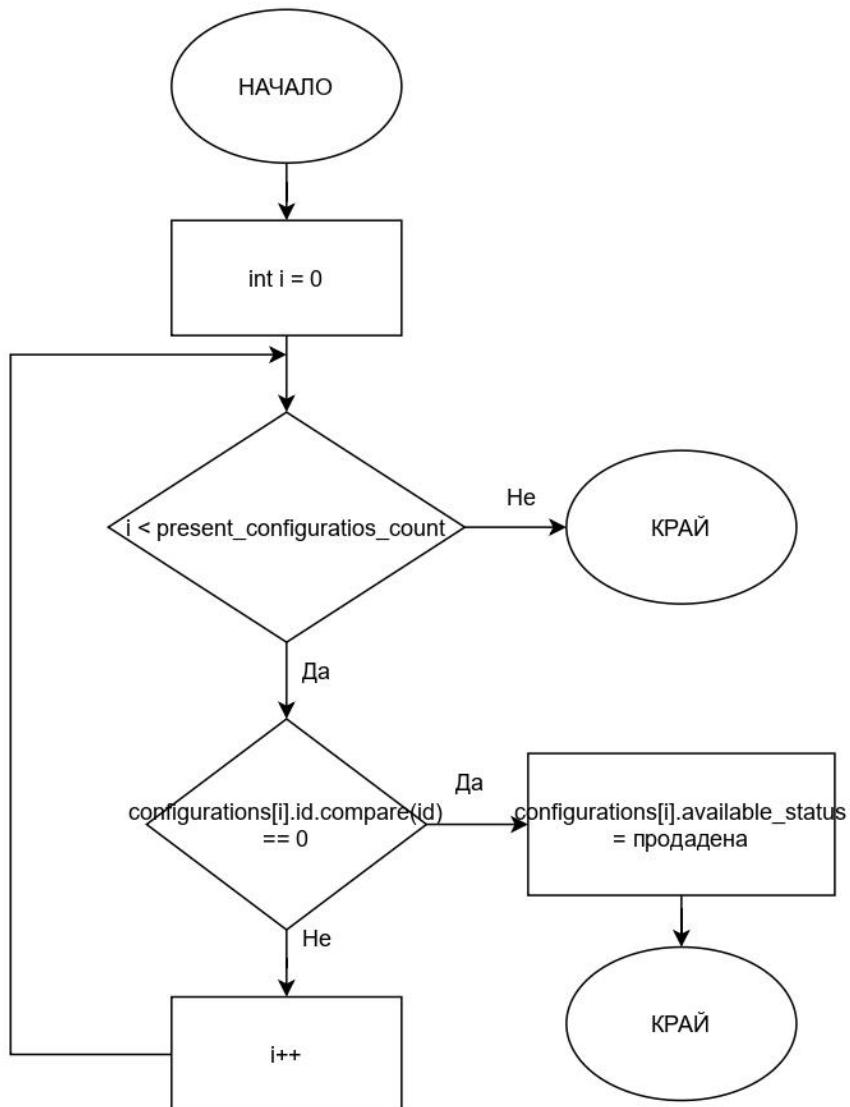
Функцията *read_computer_selling_data* се използва за прочитане на желаните характеристики на компютъра при осъществяване на продажба. Приема като входни параметри променливи по адрес от тип string, в които ще бъдат записани желаните характеристики на компютъра.

Функцията *find_computers_with_requirements* се използва за намиране на конфигурации по посочени характеристики. Приема като параметри масив с всички запазени конфигурации, брой запазени конфигурации, масив, в който да бъдат запазени намерените конфигурации и брояч за намерените конфигурации. Освен това се подават желаните характеристики на процесата и компютра, както и две променливи от тип int, които се използват за проверка дали дадена конфигурация покрива всички желани изисквания. Това са променливите *selected_features* и *actual_features*. Обхожда всички запазени конфигурации като при всяка итерация стойностите на променливите *selected_features* и *actual_features* са 0. Проверява се всяка една подадена характеристика дали е различна от празен string. Ако не е празен string, тогава потребителят иска конфигурацията да разполага с въведената характеристика. Стойността на *selected_features* нараства с 1. След като се открие поле, което не е празен string се проверява дали конфигурацията разполага с дадената характеристика. String стойностите се сравняват без значение от малка и главна буква. Ако конфигурацията разполага с желаната характеристика броячът *actual_features* също се увеличава с 1. Ако стойностите на променливите *selected_features* и *actual_features* са равни, конфигурацията отговаря на изискванията и се добавя към масива с намерените конфигурации.

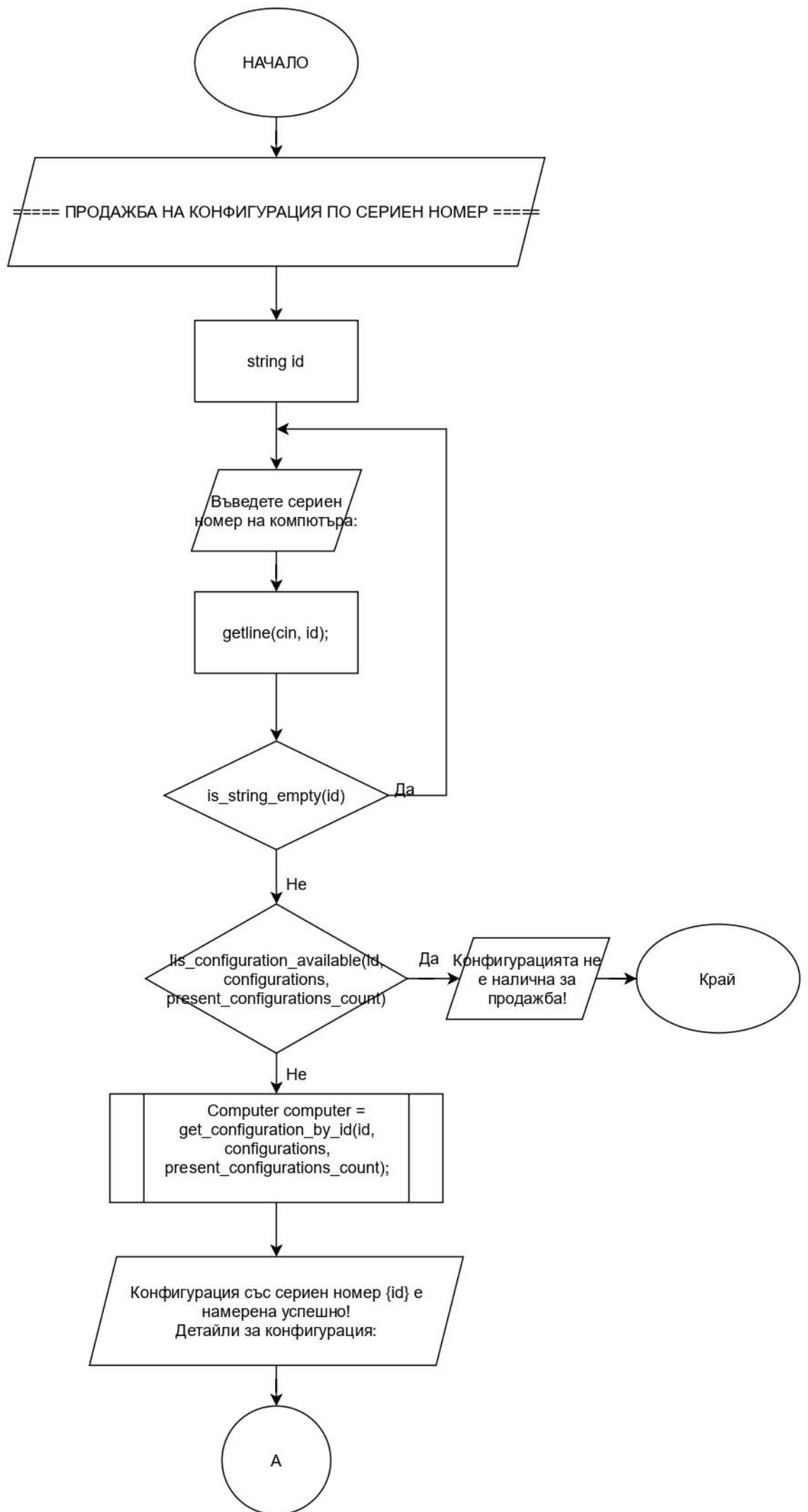
Функцията *sell_configuration_by_requirements* се използва за осъществяване на продажба по желани характеристики. Приема като параметри масив с всички запазени конфигурации и брой запазени конфигурации. Извежда информативно съобщение. Декларира string променливи за всяка една характеристика на конфигурацията. След това се прочитат стойности за променливите чрез функциите *read_processor_selling_data* и *read_computer_selling_data*. След като стойностите са прочетени се дефинира масив, в който да бъдат запазени конфигурациите, отговарящи на изискванията, брояч за намерените конфигурации и две int променливи, които следят броя на желаните характеристики и броя на характеристиките, които конфигурацията покрива. След това се извиква функцията *find_computers_with_requirements* с необходимите параметри. Ако броячът за намерените конфигурации е 0, функцията извежда съобщение на конзолата и прекратява изпълнението си. Намерените конфигурации се извеждат на екрана чрез функцията *print_all_configurations*. След това се изчаква вход от потребителя дали да бъде осъществена продажба. Ако потребителят потвърди, осъществяването на продажбата става с функцията *sell_configuration_by_id*.

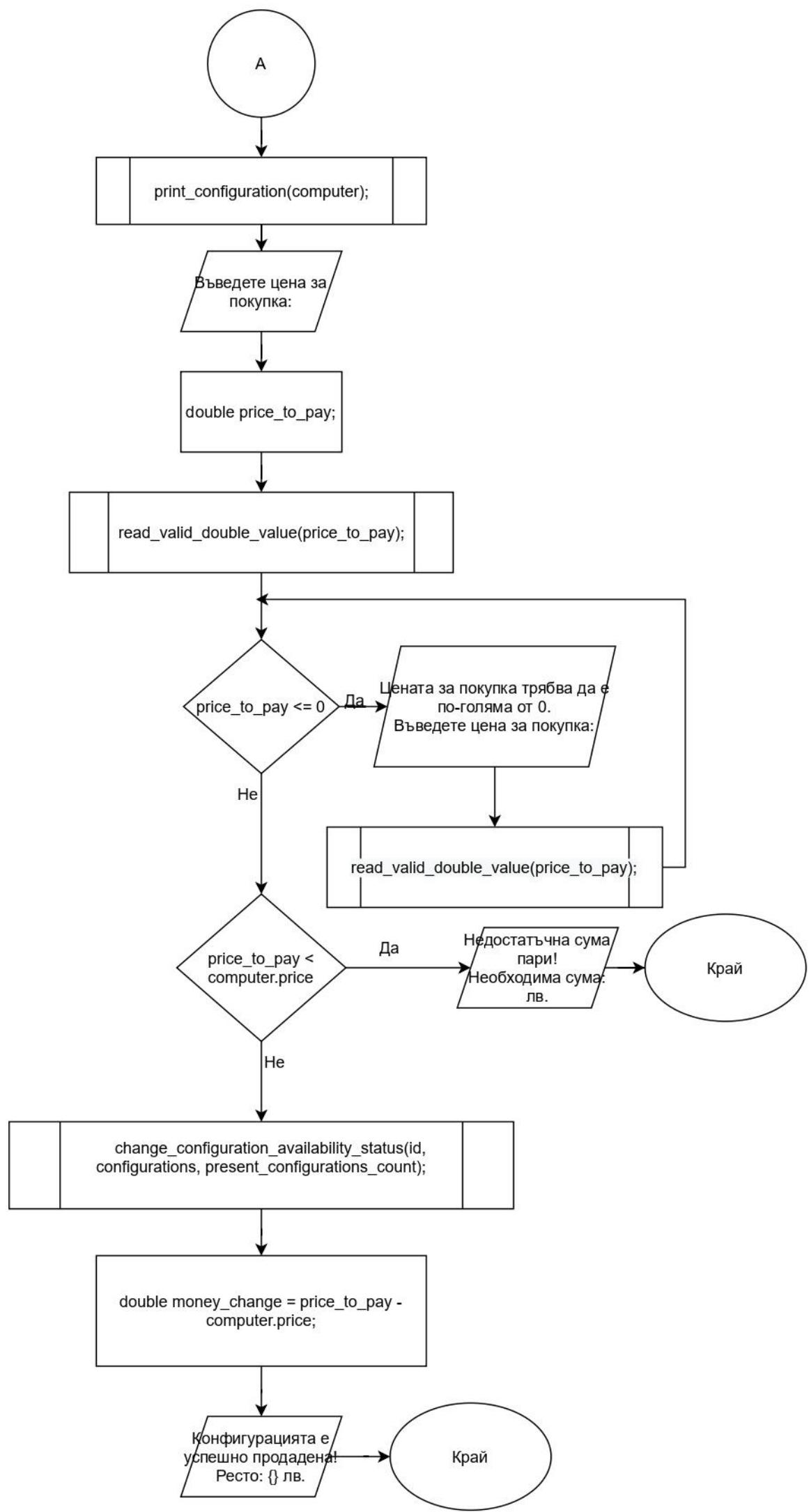
Функционалността за осъществяване на продажба е реализирана във функцията *sell_configuration*. Приема като параметри масив с всички запазени конфигурации и брой запазени конфигурации. Извежда се информативно съобщение на екрана. Ако няма запазени конфигурации се извежда съобщение и функцията приключва изпълнението си. Декларира се int променлива за меню опция. Чрез switch констрикцията и подменю се определя коя функция за продажба да се използва - *sell_configuration_by_id* или *sell_configuration_by_requirements*.

```
void change_configuration_availability_status(string& id, Computer configurations[], int present_configurations_count);
```

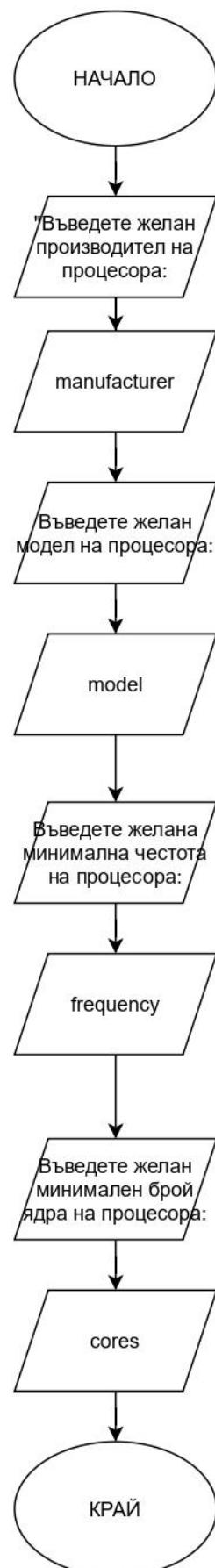


```
void sell_configuration_by_id(Computer configurations[], int present_configurations_count);
```

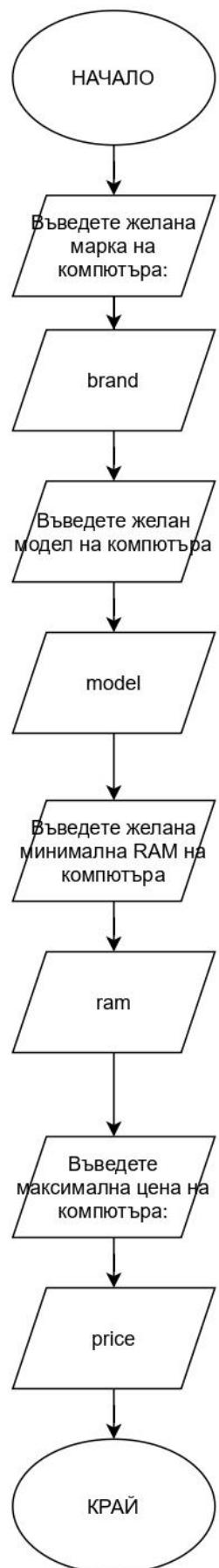




```
void read_precessor_selling_data(string& manufacturer, string& model, string& frequency, string& cores);
```



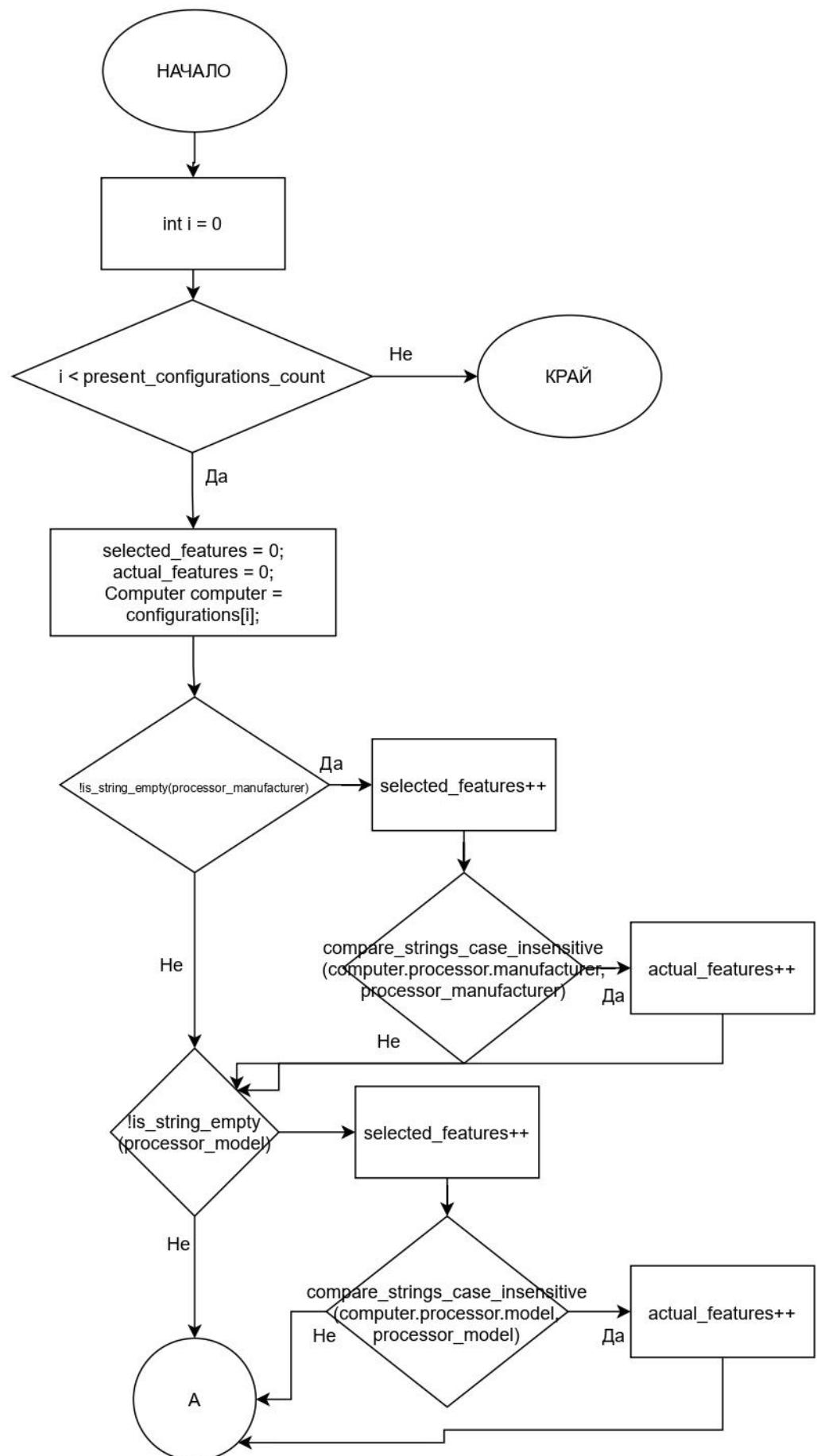
```
void read_computer_selling_data(string& brand, string& model, string& ram, string& price)
```

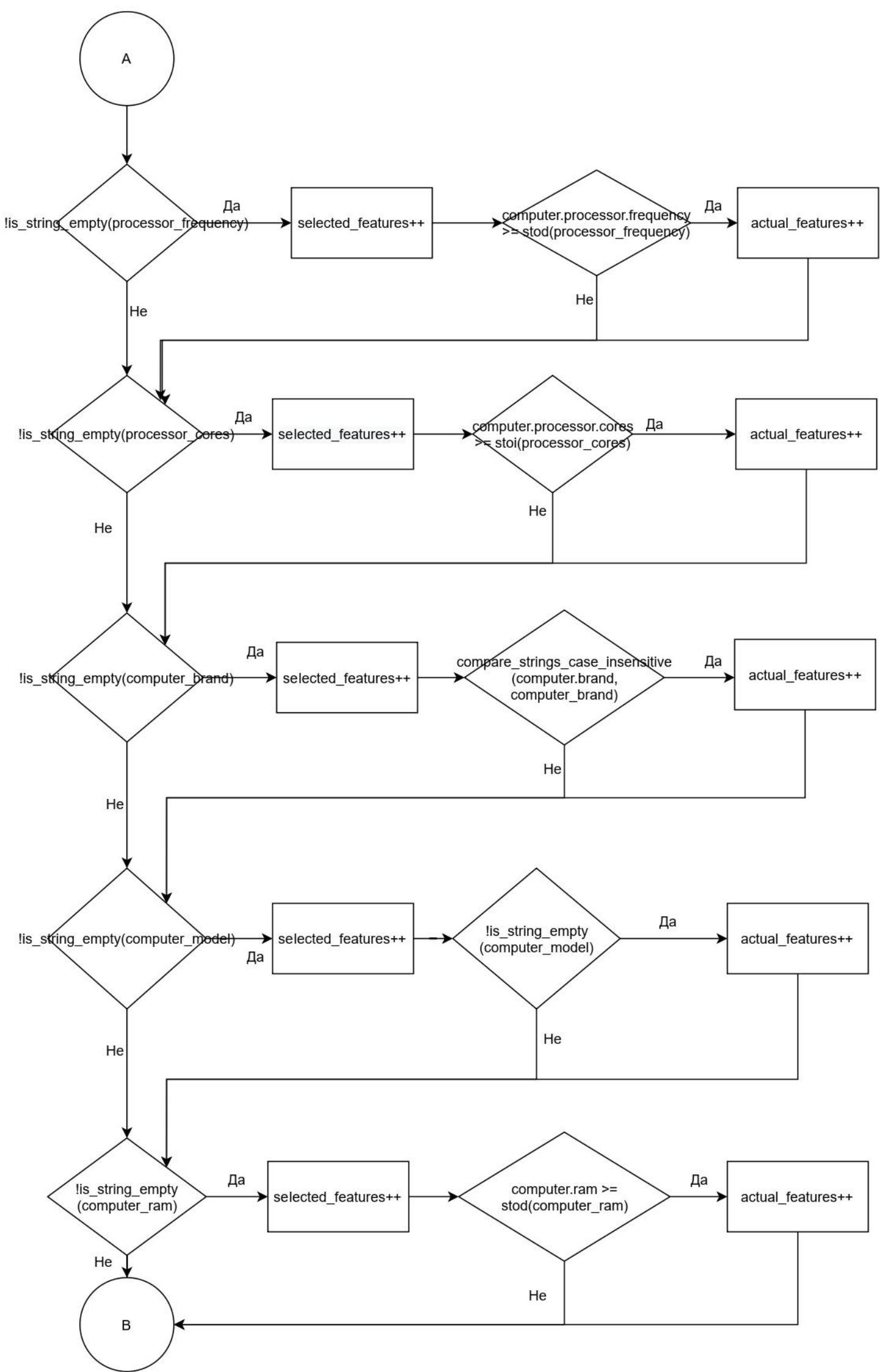


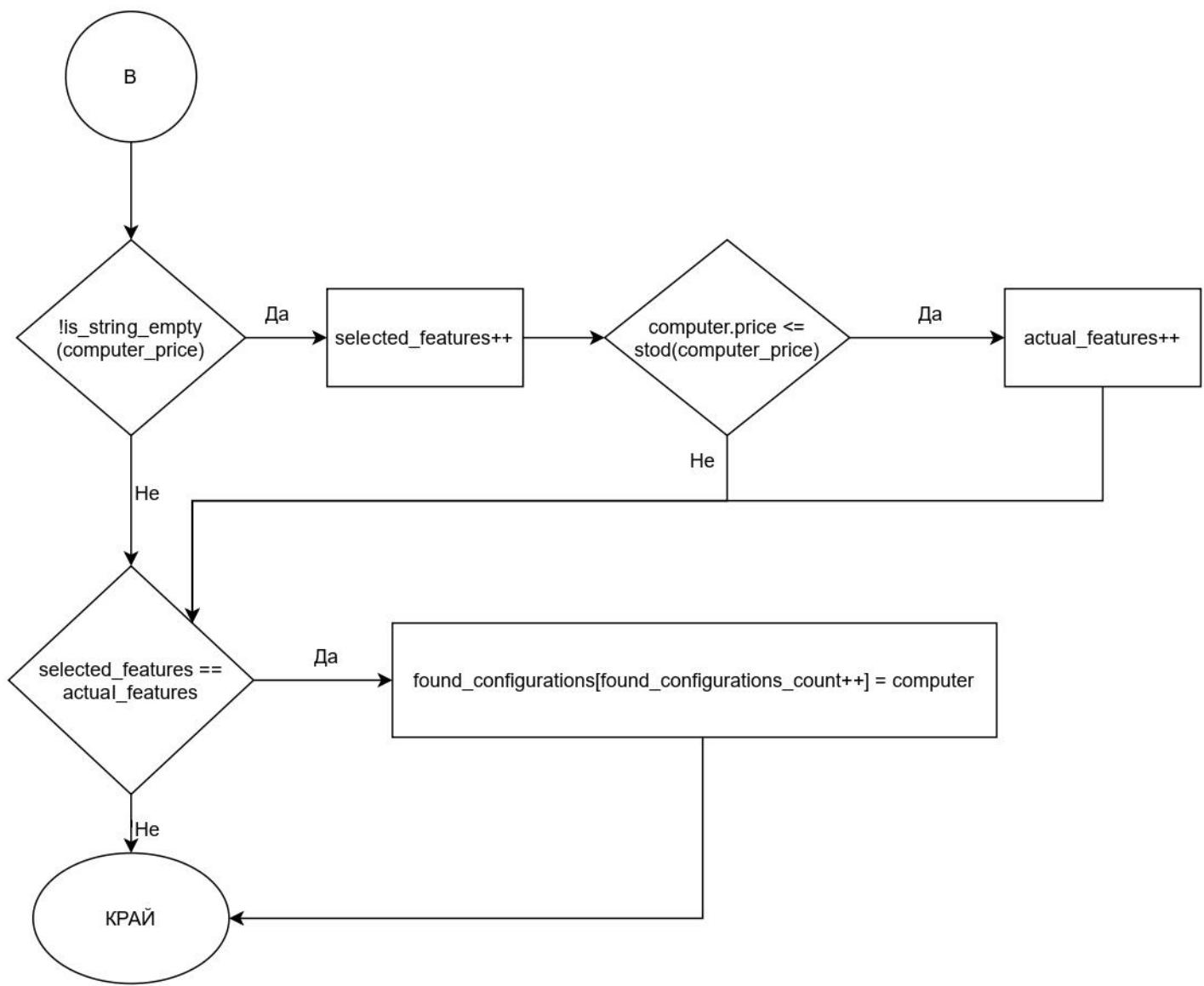
```

void find_computers_with_requirements(Computer configurations[], int present_configurations_count, Computer found_configurations[],
int& found_configurations_count,
    string& processor_manufacturer, string& processor_model, string& processor_frequency, string& processor_cores, string&
computer_brand,
    string& computer_model, string& computer_ram, string& computer_price, int& selected_features, int& actual_features);

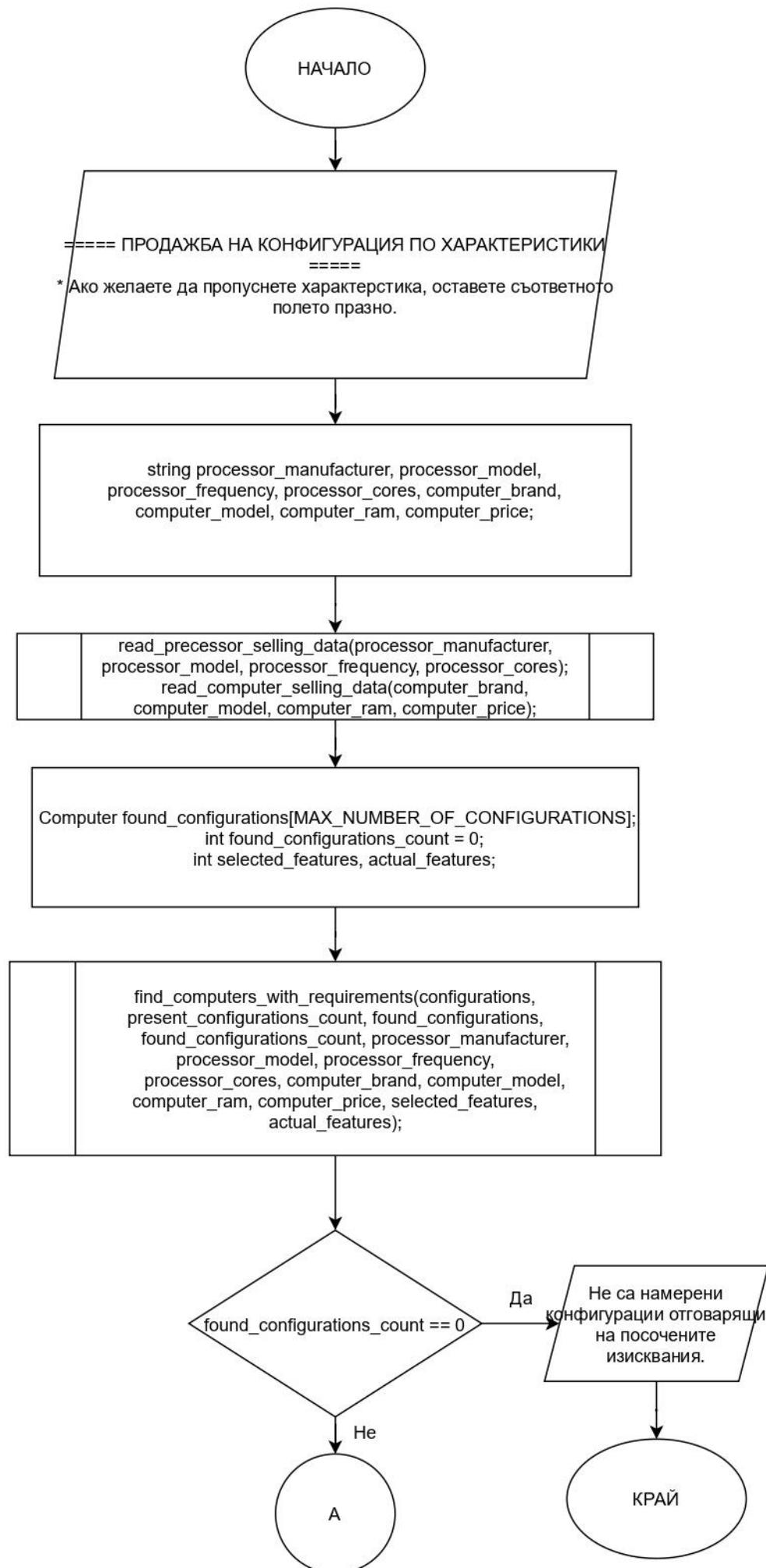
```

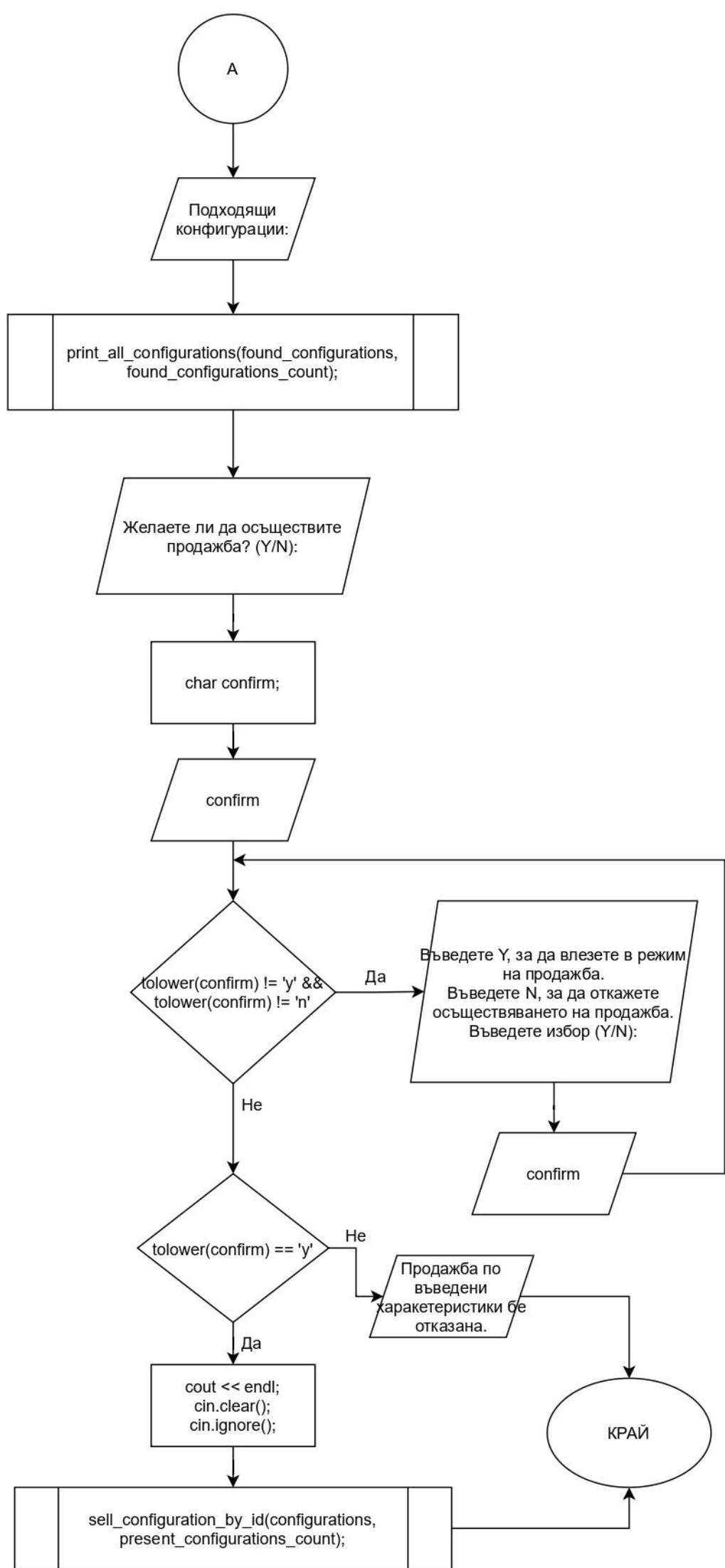




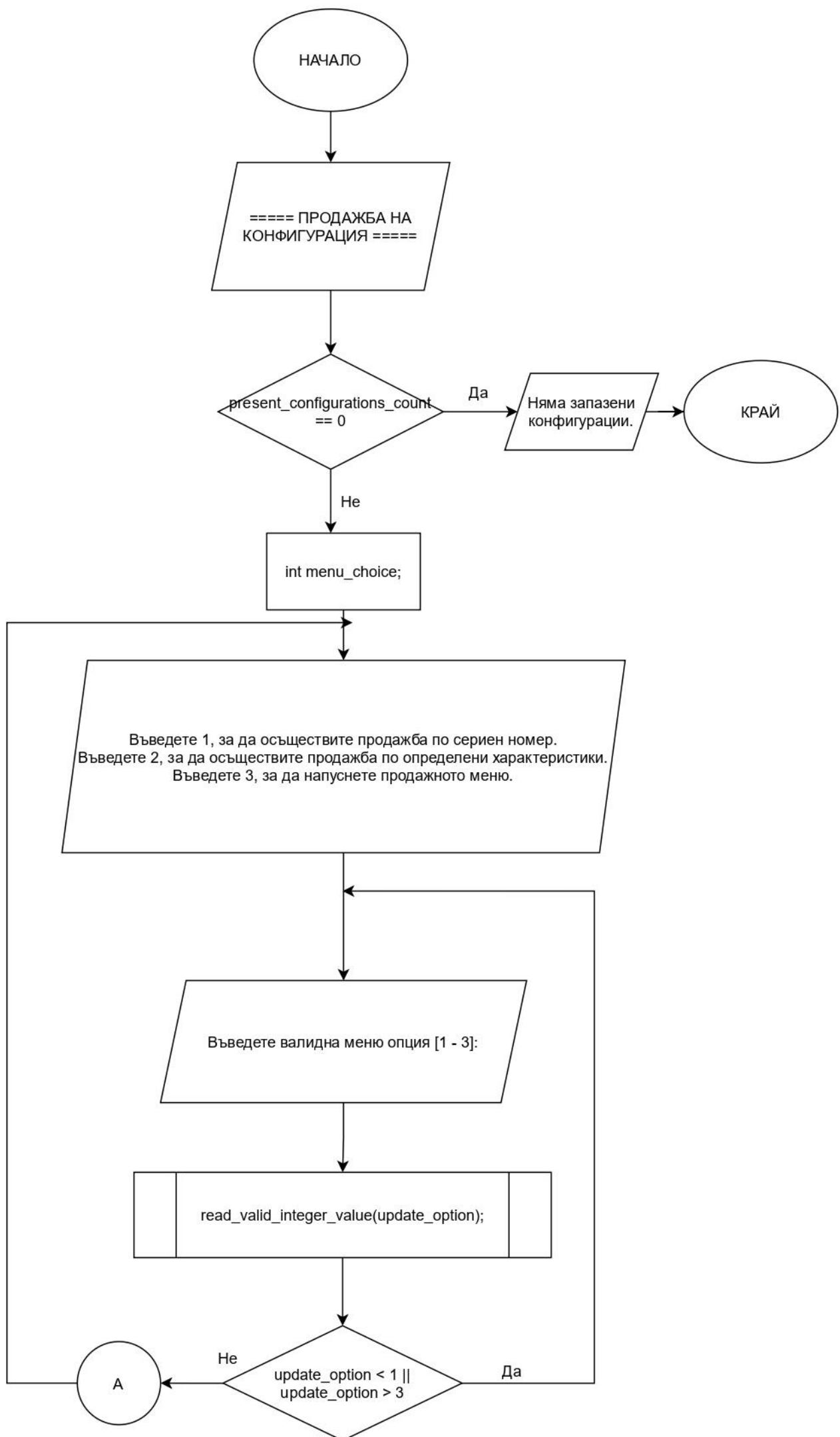


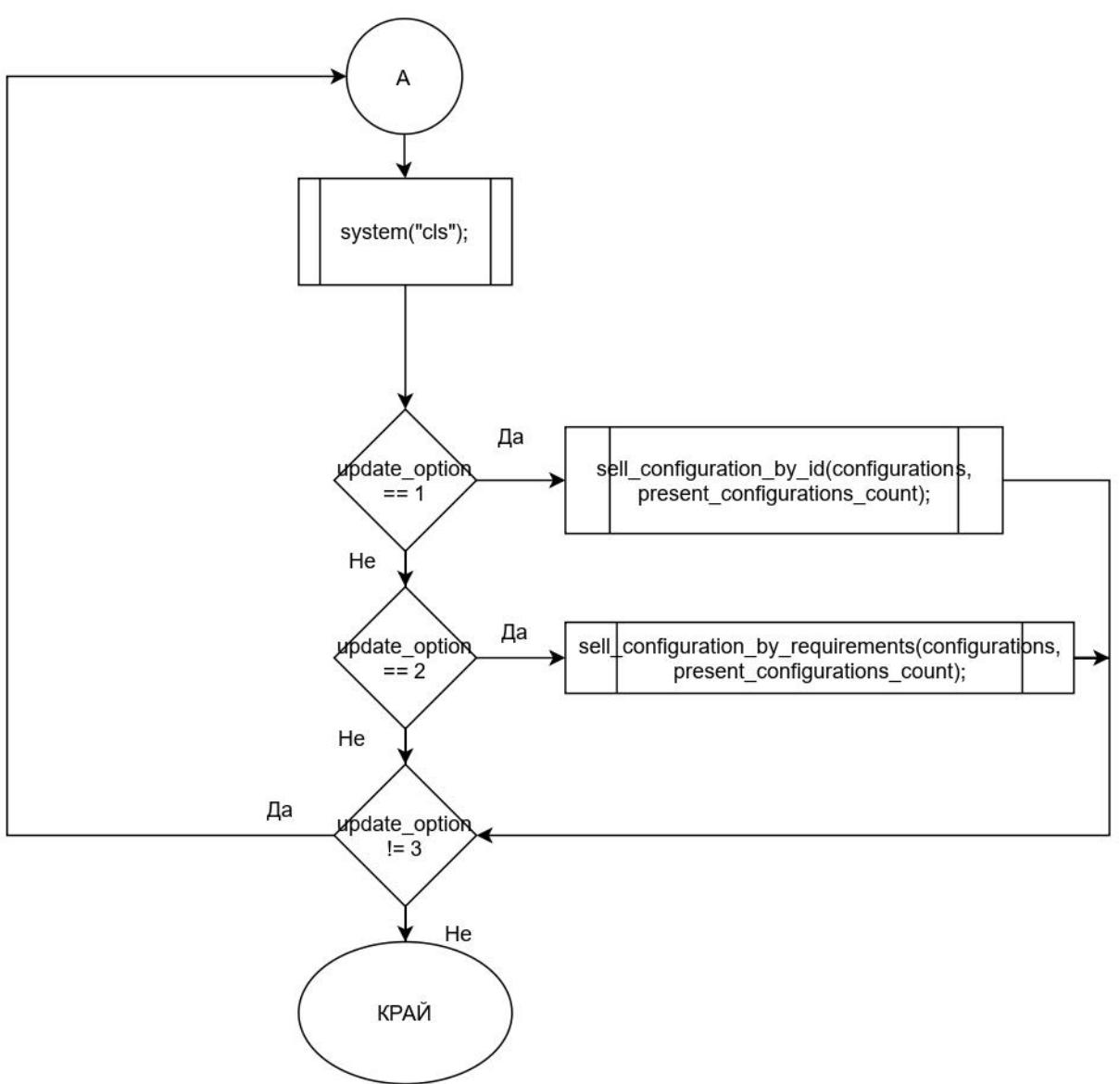
```
void sell_configuration_by_requirements(Computer configurations[], int present_configurations_count);
```





```
void sell_configuration(Computer configurations[], int present_configurations_count);
```





Екранни снимки с примерен вход и изход

a. Изглед при успешна конфигурация по сериен номер

```
===== ПРОДАЖБА НА КОНФИГУРАЦИЯ ПО СЕРИЕН НОМЕР =====

Въведете сериен номер на компютъра: 15-58-64-54-29-C1

Конфигурация със сериен номер 15-58-64-54-29-C1 е намерена успешно!

Детайли за конфигурация:

Сериен номер: 15-58-64-54-29-C1
Марка: Acer
Модел: Nitro
RAM памет: 16,00 GB
Процесор:
    Производител: Intel
    Модел: i7
    Тактова честота: 3,00 GHz
    Брой ядра: 6
Цена: 2200,00 лв.
Наличен статус: в продажба

Въведете цена за покупка: 2300.99

Конфигурацията е успешно продадена!
Ресто: 100,99 лв.
```

b. Изглед след успешна продажба по характеристики

```
===== ПРОДАЖБА НА КОНФИГУРАЦИЯ ПО ХАРАКТЕРИСТИКИ =====

* Ако желаете да пропуснете характеристика, оставете съответното поле празно.

Въведете желан производител на процесора: Intel
Въведете желан модел на процесора: i7
Въведете желана минимална честота на процесора:
Въведете желан минимален брой ядра на процесора:
Въведете желана марка на компютъра:
Въведете желан модел на компютъра:
Въведете желана минимална RAM на компютъра:
Въведете максимална цена на компютъра:

Подходящи конфигурации:

Сериен номер: 15-58-64-54-29-C1
Марка: Acer
Модел: Nitro
RAM памет: 16,00 GB
Процесор:
    Производител: Intel
    Модел: i7
    Тактова честота: 3,00 GHz
    Брой ядра: 6
Цена: 2200,00 лв.
Наличен статус: в продажба
```

Желаете ли да осъществите продажба? (Y/N): Y

===== ПРОДАЖБА НА КОНФИГУРАЦИЯ ПО СЕРИЕН НОМЕР =====

Въведете сериен номер на компютъра: 15-58-64-54-29-C1

Конфигурация със сериен номер 15-58-64-54-29-C1 е намерена успешно!

Детайли за конфигурация:

Сериен номер: 15-58-64-54-29-C1

Марка: Acer

Модел: Nitro

RAM памет: 16,00 GB

Процесор:

Производител: Intel

Модел: i7

Тактова честота: 3,00 GHz

Брой ядра: 6

Цена: 2200,00 лв.

Наличен статус: в продажба

Въведете цена за покупка: 2200

Конфигурацията е успешно продадена!

Ресто: 0,00 лв.

Въведете 1, за да осъществите продажба по сериен номер.

Въведете 2, за да осъществите продажба по определени характеристики.

Въведете 3, за да напуснете продажното меню.

Въведете валидна меню опция [1 - 3]:

F. Одит на конфигурациите в под меню

- a. Извеждане на всички конфигурации, които са в продажба, сортирани по сериен номер
- b. Извеждане на всички конфигурации с даден модел процесор и RAM памет, сортирани по цена от най-скъпия към най-евтиния
- c. Извеждане на продадените конфигурации, сортирани по модел на процесора

```
bool search_for_configurations_by_availability_status(Computer configurations[], int present_configurations_count, string available_status);

int find_min_number(int first, int second);

void sort_configurations_by_id(Computer configurations[], int present_configurations_count, Computer sorted_configurations[]);

void print_configurations_by_availability_status(Computer configurations[], int present_configurations_count, string available_status);

void print_available_configurations_sorted_by_id(Computer configurations[], int present_configurations_count);

bool configuration_exists_by_processor_model_and_ram(Computer configurations[], int present_configurations_count, string& processor_model, double ram);

void sort_configurations_by_price_desc(Computer configurations[], int present_configurations_count, Computer sorted_configurations[]);

void print_configurations_by_brand_and_ram(Computer configurations[], int present_configurations_count, string& processor_model, double ram);

void print_configurations_by_brand_and_ram_sorted_by_price_desc(Computer configurations[], int present_configurations_count);

void sort_configurations_by_processor_model(Computer configurations[], int present_configurations_count, Computer sorted_configurations[]);

void print_unavailable_configurations_sorted_by_processor_model(Computer configurations[], int present_configurations_count);

void make_configurations_audit(Computer configurations[], int present_configurations_count);
```

Функцията `search_for_configurations_by_availability_status` проверява за съществуваща конфигурация по даден наличен статус. Връща като резултат булева стойност. Приема като параметри масива, в който са записани всички конфигурации, броя на записани конфигурации и наличният статус за проверка. Обхождат се всички конфигурации и се проверява дали текущата конфигурация има посочения наличие статус. Ако условието е изпълнено функцията връща true, в противен случай – false.

Функцията `find_min_number` сравнява две числа и връща по-малкото между тях. Приема като параметри две целочислени стойности. Връща int стойност. Ако числата са равни ще се върне стойността на второто число.

Функцията *sort_configurations_by_id* сортира запазените конфигурации по сериен номер в нов масив. Приема като параметри масива, в който са записани всички конфигурации, броя на записани конфигурации и нов масив, в който се запазват сортирани конфигурациите. Обхождат се всички конфигурации в първия масив и се копират във втория масив (този за сортирани конфигурации). Конфигурациите се сортират във възходящ ред чрез метода на мехурчето. Алгоритъмът е оптимизиран с булева променлива, която проверява дали се е осъществила размяна на елементи. При сортирането по сериен номер се сравняват ASCII стойностите на символите в серийните номера. Обхожда се по-краткият сериен номер чрез функцията *find_min_number*.

Функцията *print_configurations_by_availability_status* извежда всички конфигурации с даден наличен статус. Приема като параметри масива, в който са записани всички конфигурации, броя на записани конфигурации и наличният статус за проверка. Обхождат се всички конфигурации и се проверява дали текущата конфигурация има посочения наличен статус. Ако условието е изпълнено конфигурацията се извежда чрез *print_configuration*.

Функционалността за извеждането на наличните конфигурации сортирани по сериен номер е реализирана във функцията *print_available_configurations_sorted_by_id*. Приема като параметри масива, в който са записани всички конфигурации и броя на записани конфигурации. Извежда информативно съобщение экрана. Ако няма запазени конфигурации или няма налични такива се извежда съответното съобщение на экрана и функцията прекратява своето изпълнение. Ако съществуват налични конфигурации, те се сортират в декларирания масив и се извеждат на экрана чрез *sort_configurations_by_id* и *print_configurations_by_availability_status*.

Функцията *configuration_exists_by_processor_model_and_ram* проверява за съществуваща конфигурация по модел на процесора и RAM стойност. Връща като резултат булева стойност. Приема като параметри масива, в който са записани всички конфигурации, броя на записани конфигурации, моделът на процесора и RAM стойността за проверка. Обхождат се всички конфигурации и се проверява дали текущата конфигурация има посочения модел процесор и RAM. Ако условието е изпълнено функцията връща true, в противен случай – false.

Функцията *sort_configurations_by_price_desc* сортира запазените конфигурации по цена в низходящ ред в нов масив. Приема като параметри масива, в който са записани всички конфигурации, броя на записани конфигурации и нов масив, в който се запазват сортирани конфигурациите. Обхождат се всички конфигурации в първия масив и се копират във втория масив (този за сортирани конфигурации). Конфигурациите се сортират във възходящ ред чрез метода на мехурчето. Алгоритъмът е оптимизиран с булева променлива, която проверява дали се е осъществила размяна на елементи.

Функцията *print_configurations_by_brand_and_ram* извежда всички конфигурации с даден модел на процесора и RAM памет. Приема като параметри масива, в който са записани всички конфигурации, броя на записани конфигурации, моделът на процесора и RAM стойността за проверка. Обхождат се всички конфигурации и се проверява дали текущата конфигурация има посочените параметри. Ако условието е изпълнено конфигурацията се извежда чрез *print_configuration*.

Функционалността за извеждането на конфигурации с даден модел на процесора и RAM памет сортирани по цена в низходящ ред е реализирана във функцията *print_configurations_by_brand_and_ram_sorted_by_price_desc*. Приема като параметри масива, в който са записани всички конфигурации и броя на записани конфигурации. Извежда информативно съобщение экрана. Прочита стойности за желаният модел процесор и RAM памет. Двата входа са валидирани. Ако няма запазени конфигурации или няма конфигурации с желаният модел процесор и RAM памет се извежда съответното съобщение на экрана и функцията прекратява своето изпълнение. Ако съществуват конфигурации, те се сортират в

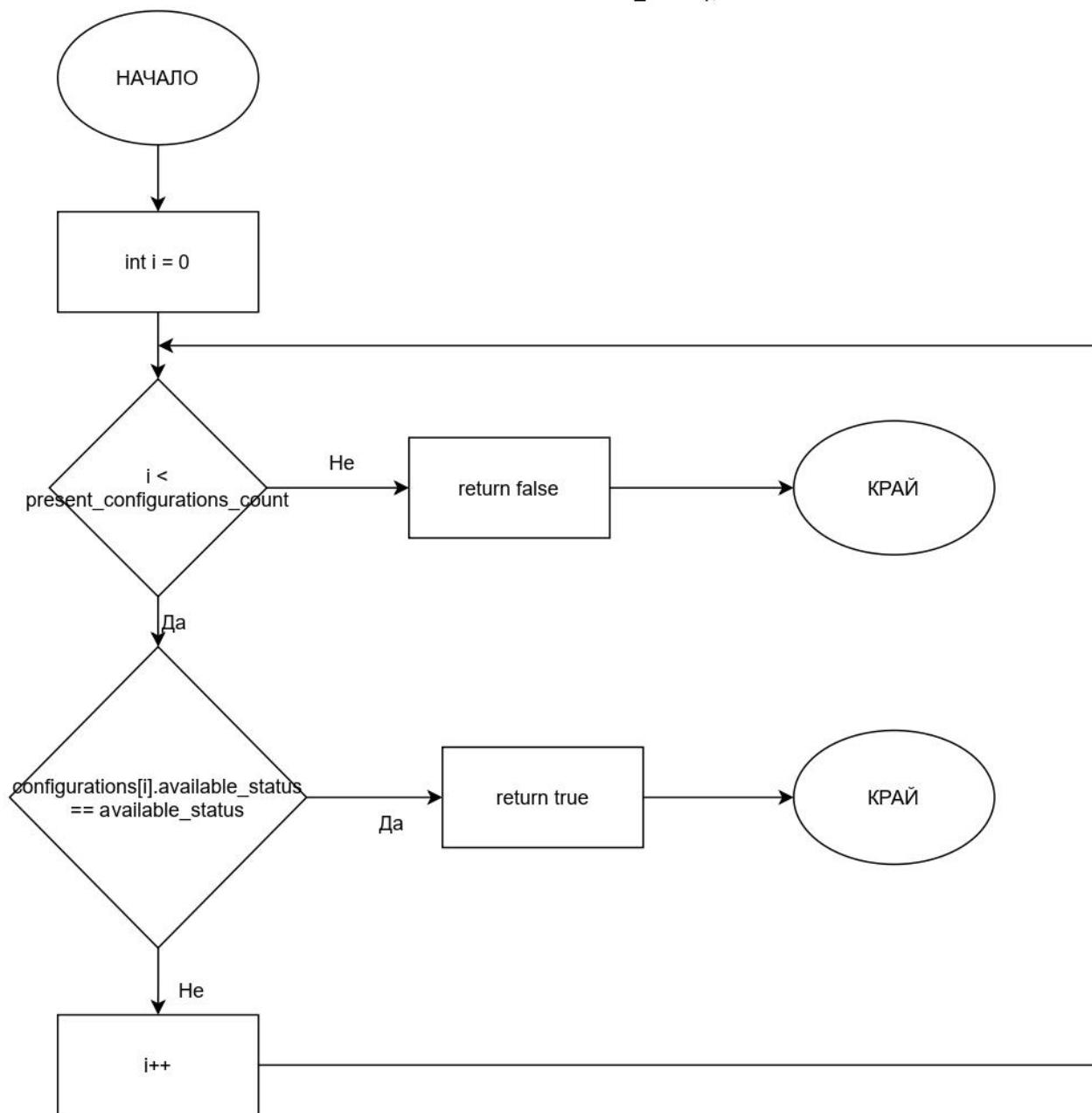
декларирация масив и се извеждат на екрана чрез *sort_configurations_by_price_desc* и *print_configurations_by_availability_status*.

Функцията *sort_configurations_by_processor_model* сортира запазените конфигурации по модел процесор в нов масив. Приема като параметри масива, в който са записани всички конфигурации, броя на записани конфигурации и нов масив, в който се запазват сортирани конфигурациите. Обхождат се всички конфигурации в първия масив и се копират във втория масив (този за сортирани конфигурации). Конфигурациите се сортират във възходящ ред чрез метода на мехурчето. Алгоритъмът е оптимизиран с булева променлива, която проверява дали се е осъществила размяна на елементи. При сортирането по модел на процесора се сравняват ASCII стойностите на символите в модела на процесора. Обхожда се по-краткият сериен номер чрез функцията *find_min_number*.

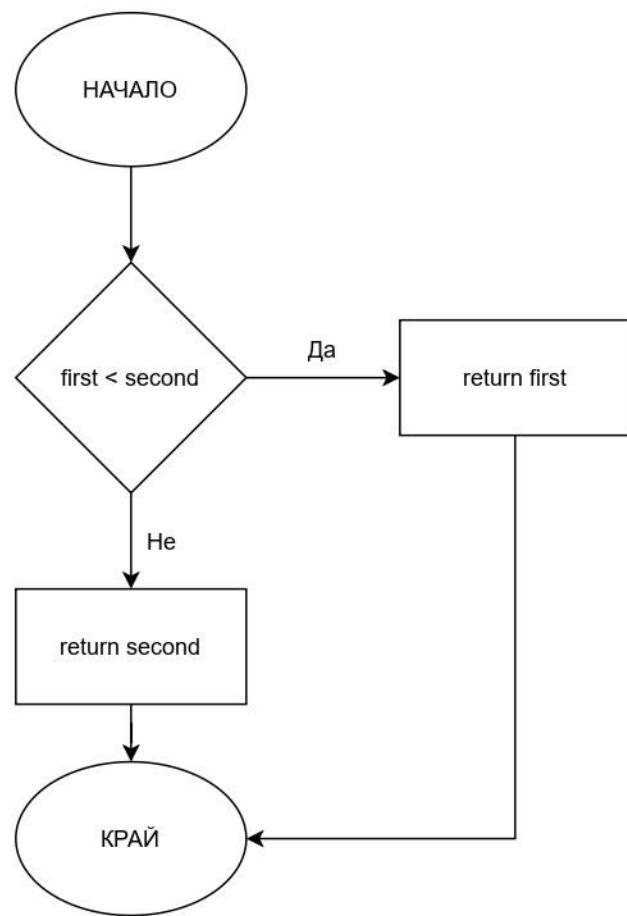
Функционалността за извеждането на продадените конфигурации сортирани по модел на процесора е реализирана във функцията *print_unavailable_configurations_sorted_by_processor_model*. Приема като параметри масива, в който са записани всички конфигурации и броя на записани конфигурации. Извежда информативно съобщение екрана. Ако няма запазени конфигурации или няма продадени такива се извежда съответното съобщение на екрана и функцията прекратява своето изпълнение. Ако съществуват продадени конфигурации, те се сортират в декларирания масив и се извеждат на екрана чрез *sort_configurations_by_processor_model* и *print_configurations_by_availability_status*.

Функционалността за осъществяване на одит е реализирана във функцията *make_configurations_audit*. Приема като параметри масива, в който са записани всички конфигурации и броя на записани конфигурации. Извежда информативно съобщение екрана. Ако няма запазени конфигурации се извежда съответното съобщение на екрана и функцията прекратява своето изпълнение. Дефинира се променлива за меню опция и чрез switch конструкция и подменю се определя какъв одит да бъде направен.

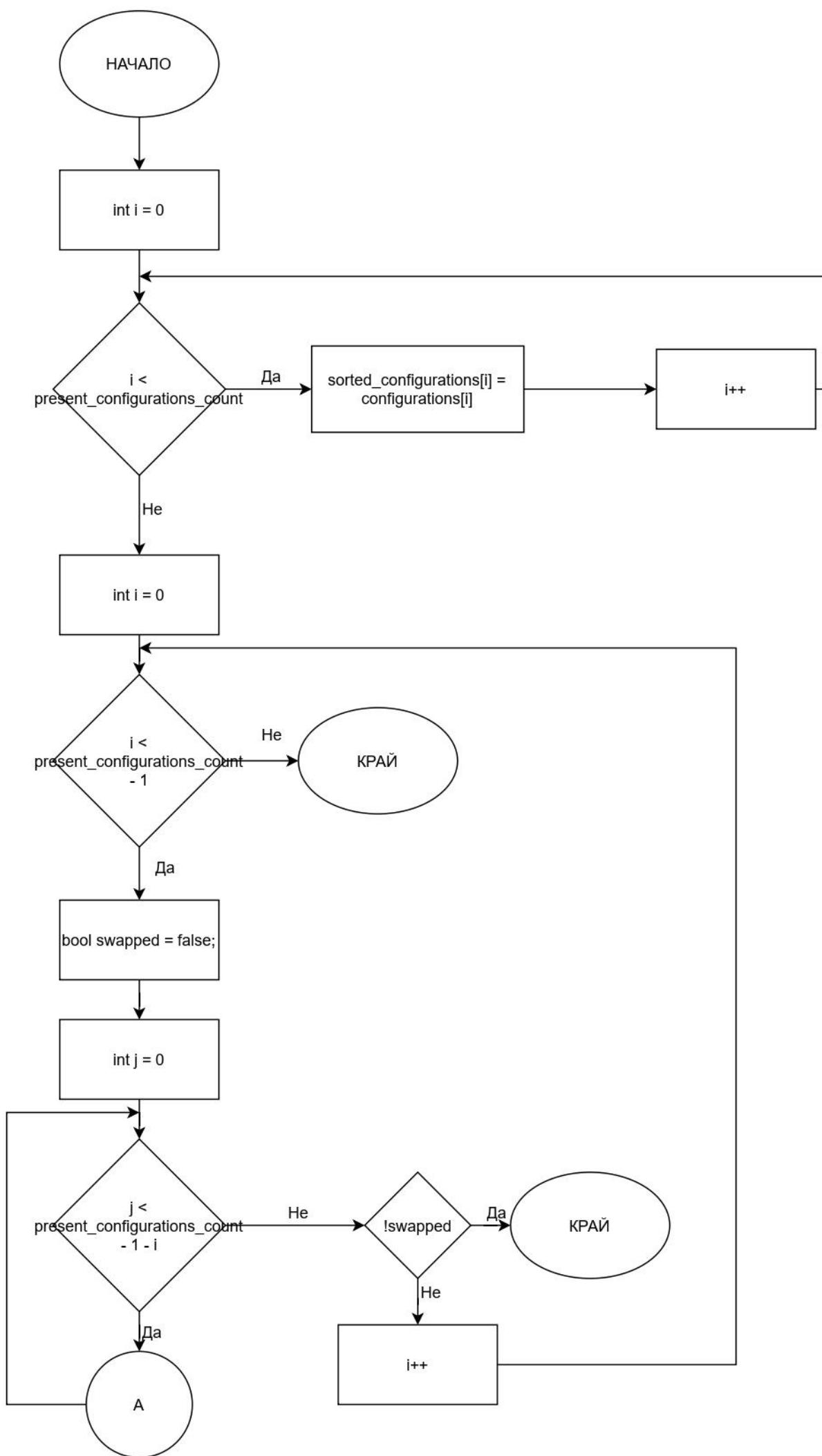
```
bool search_for_configurations_by_availability_status(Computer configurations[], int present_configurations_count, string available_status);
```

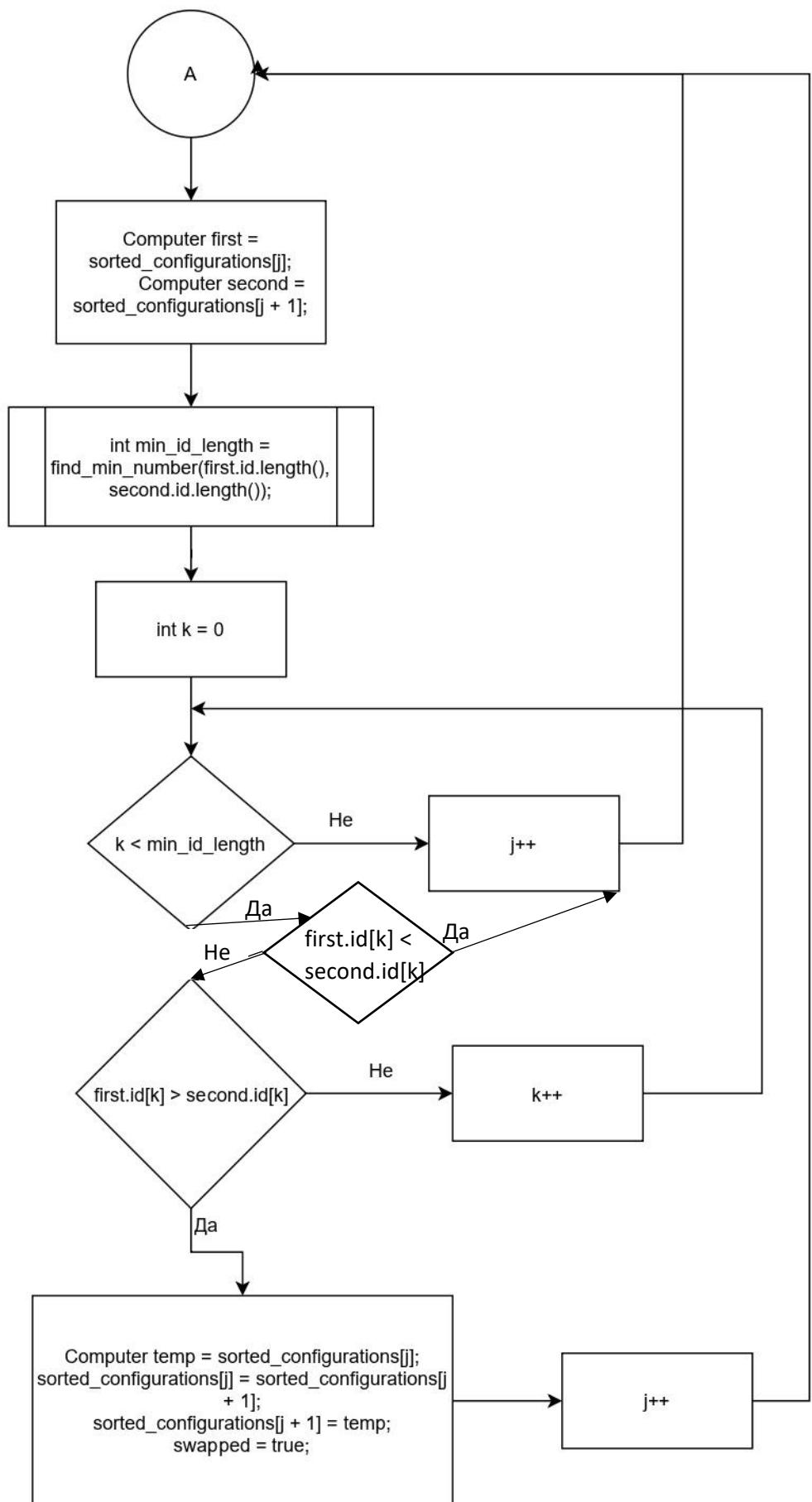


```
int find_min_number(int first, int second);
```

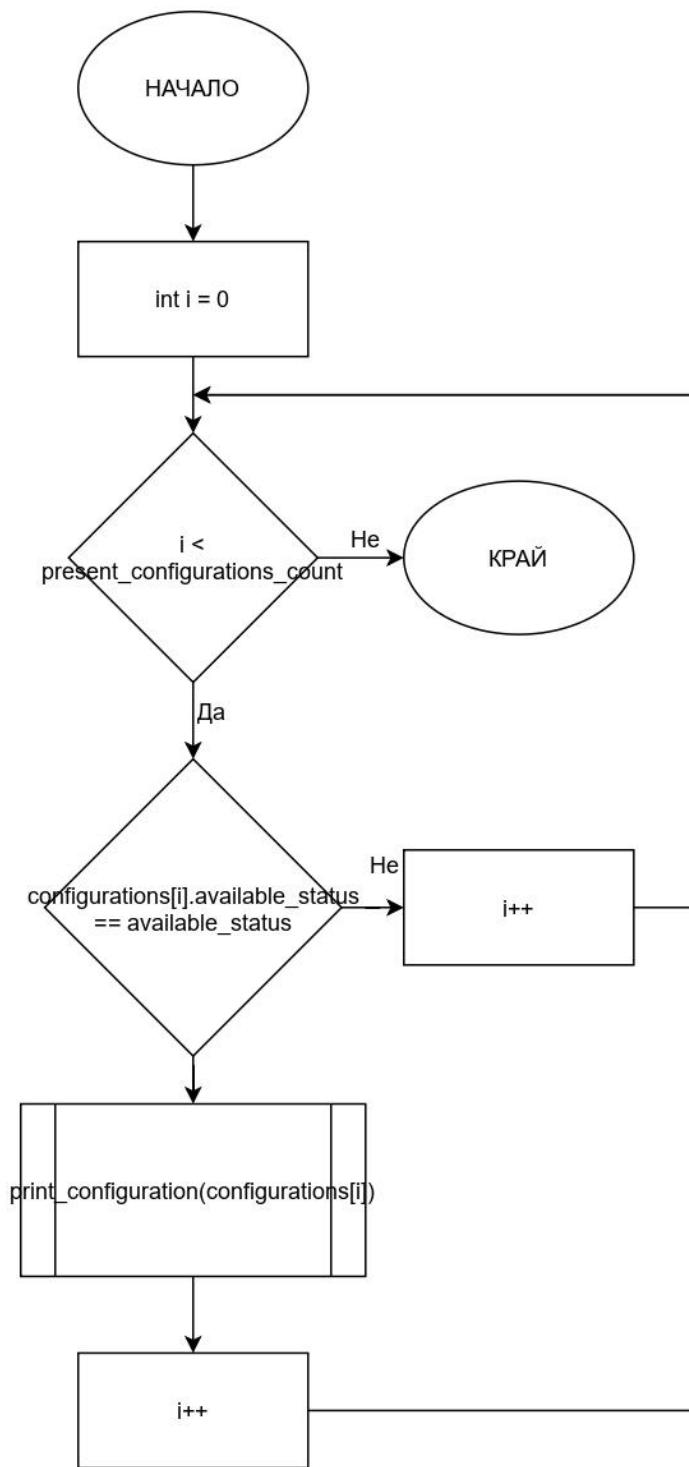


```
void sort_configurations_by_id(Computer configurations[], int present_configurations_count, Computer sorted_configurations[]);
```

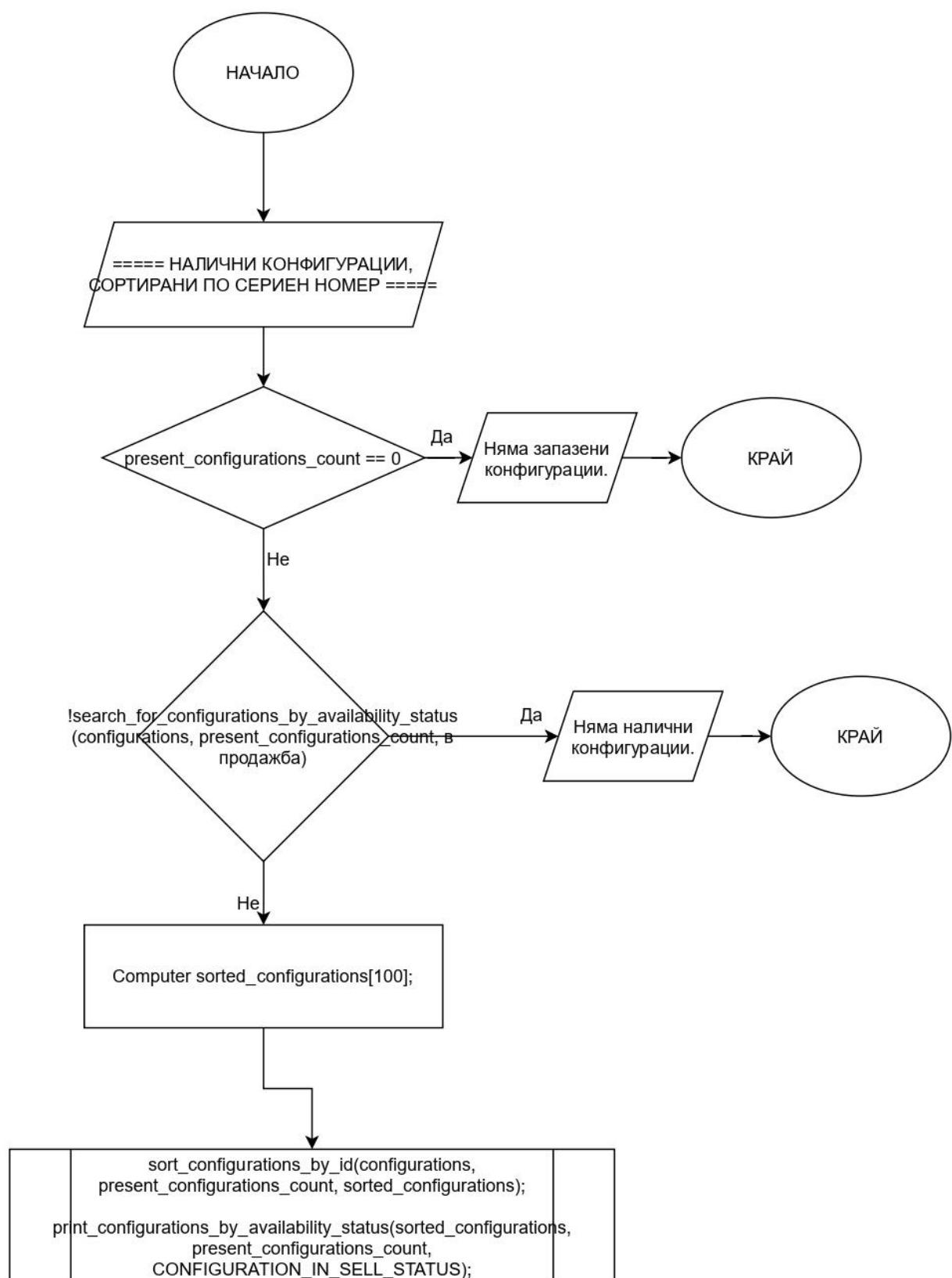




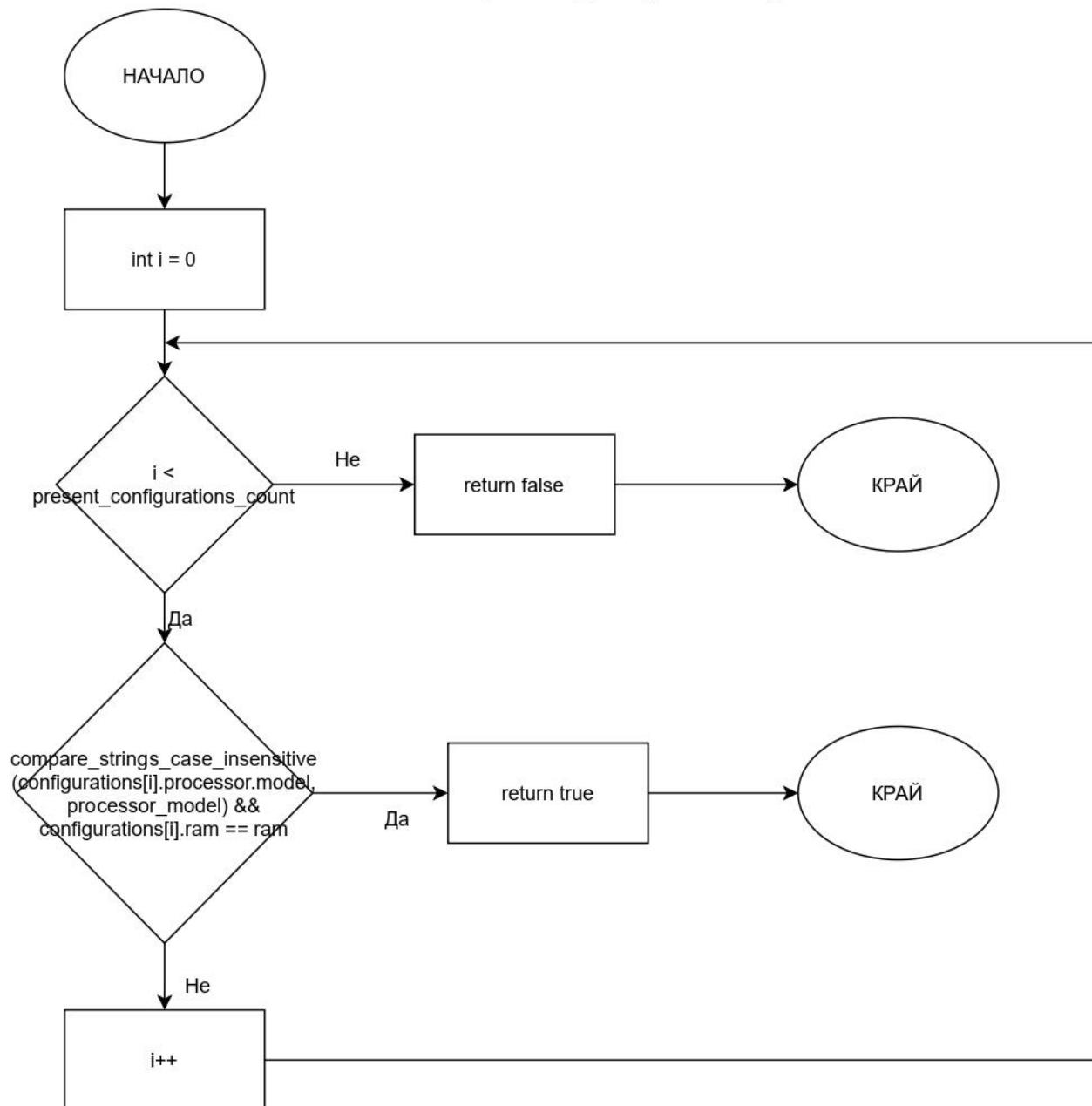
```
void print_configurations_by_availability_status(Computer configurations[], int present_configurations_count, string available_status);
```



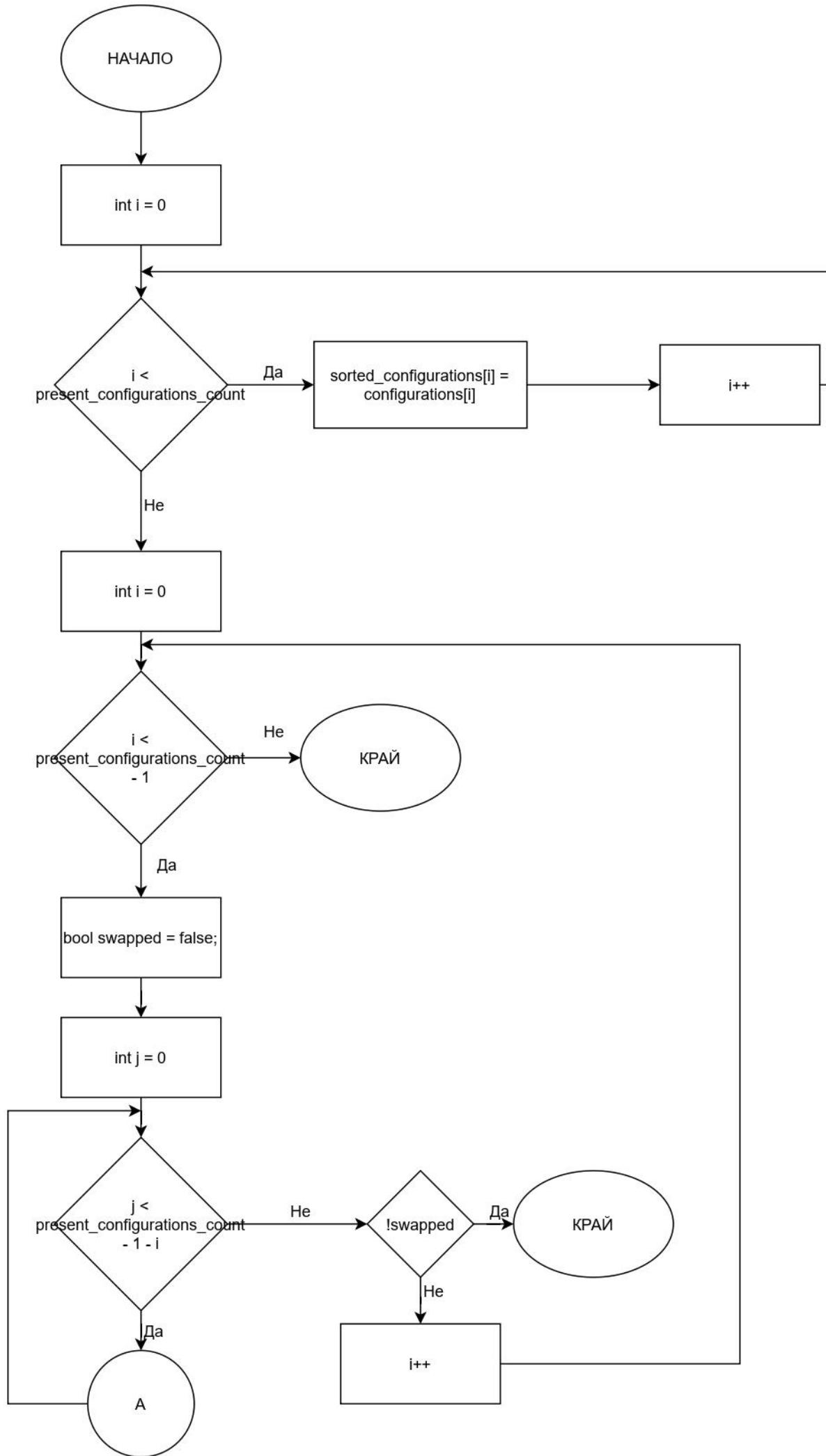
```
void print_available_configurations_sorted_by_id(Computer configurations[], int present_configurations_count)
```

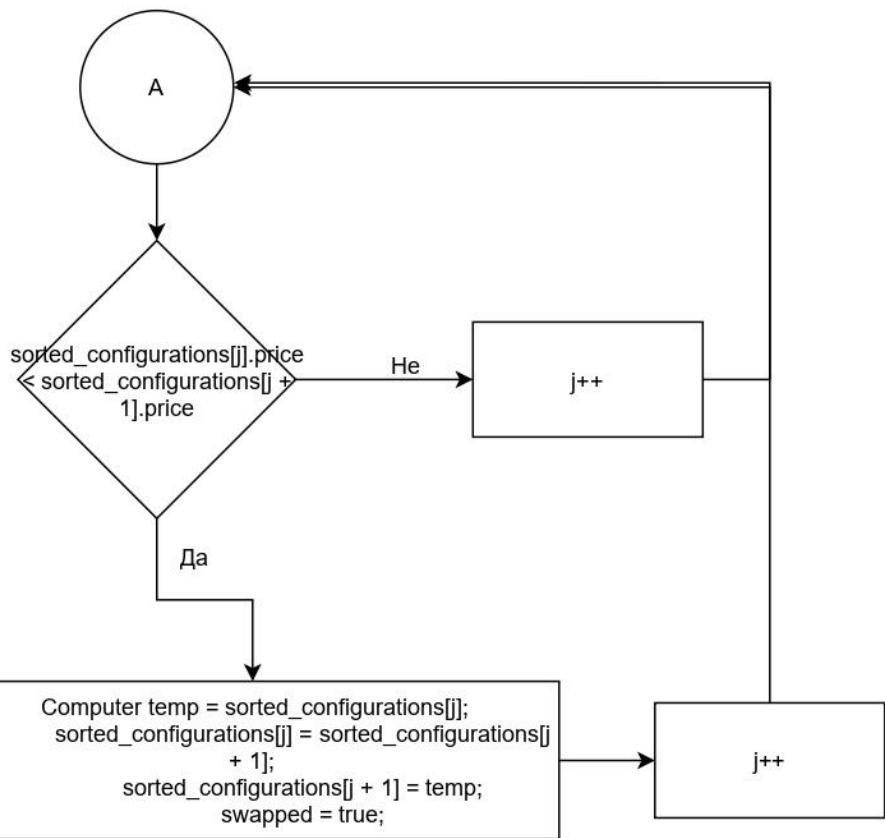


```
bool configuration_exists_by_processor_model_and_ram(Computer configurations[], int present_configurations_count, string& processor_model, double ram);
```

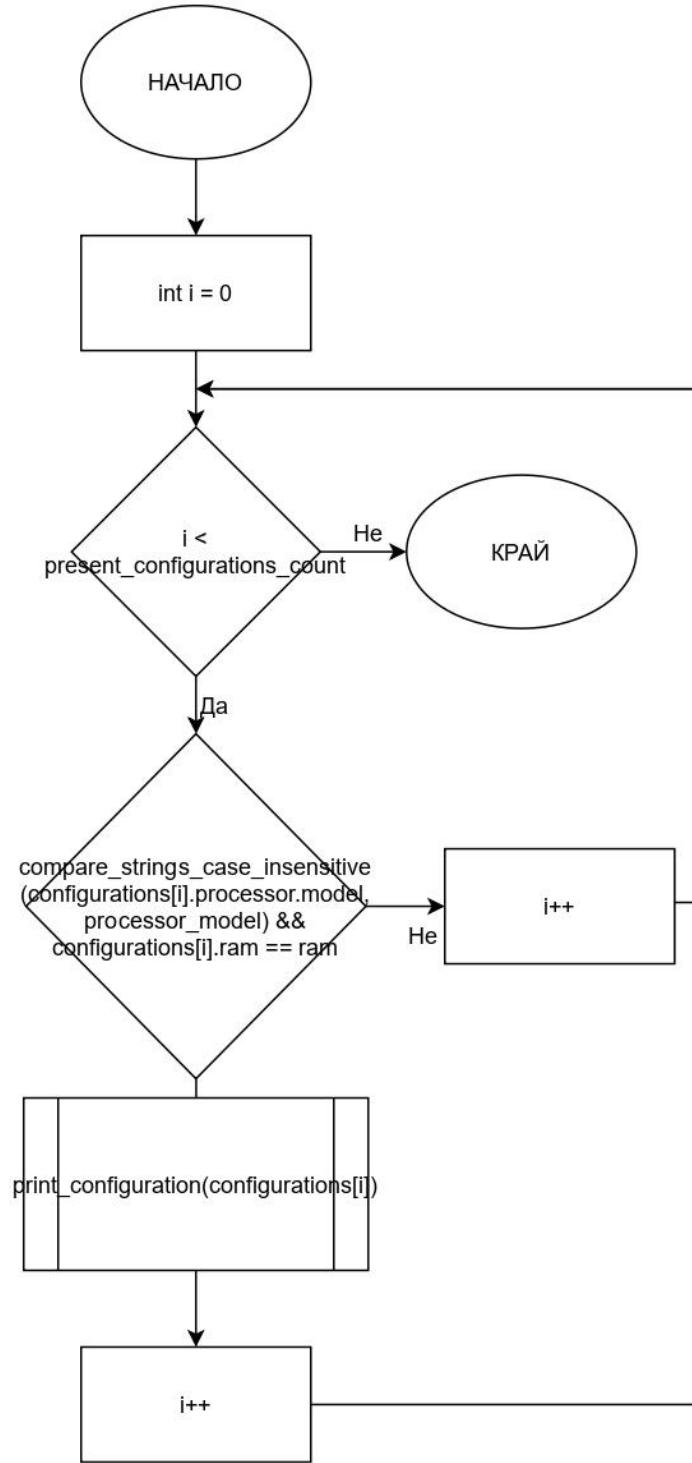


```
void sort_configurations_by_price_desc(Computer configurations[], int present_configurations_count, Computer sorted_configurations[]);
```

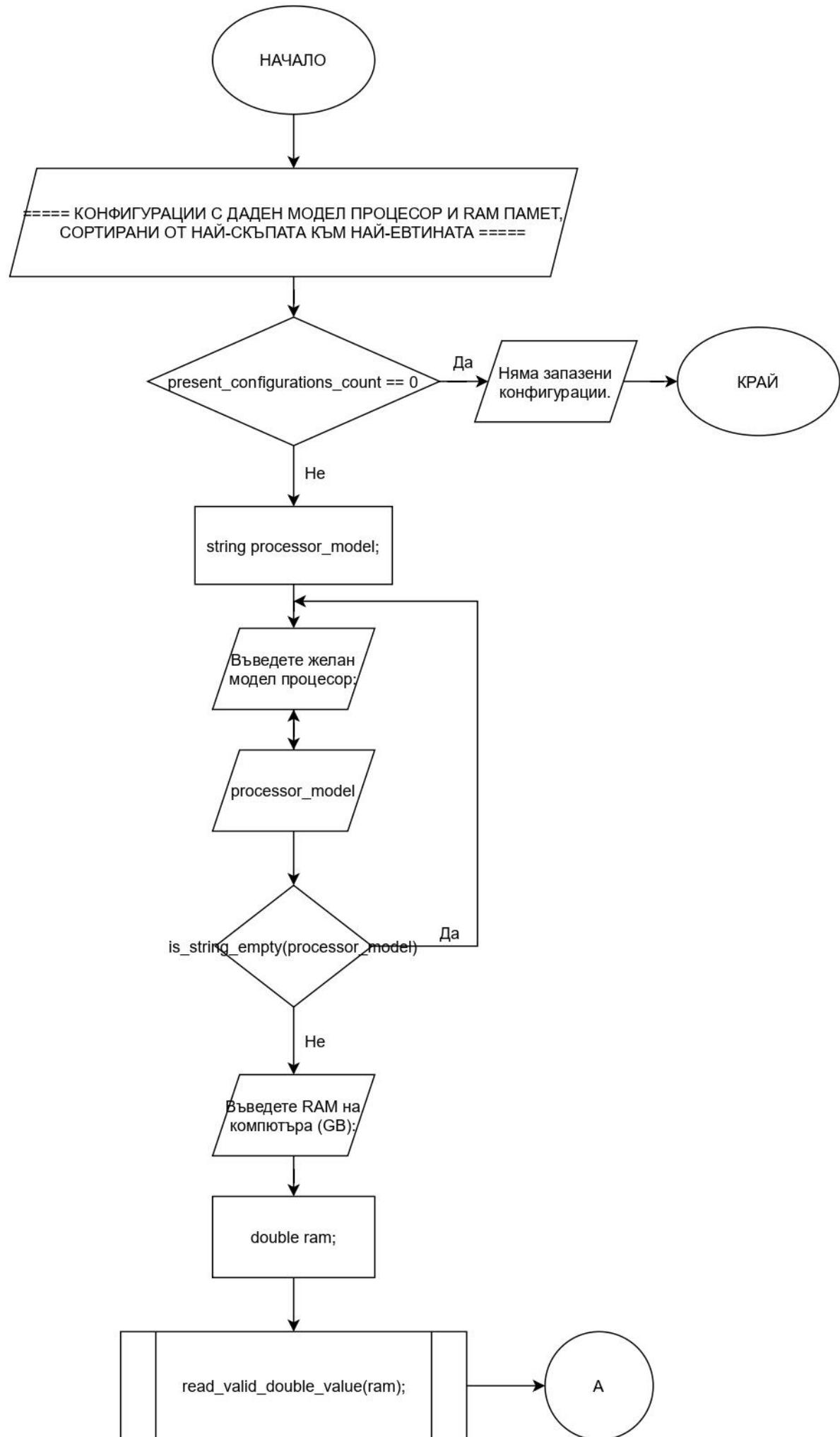


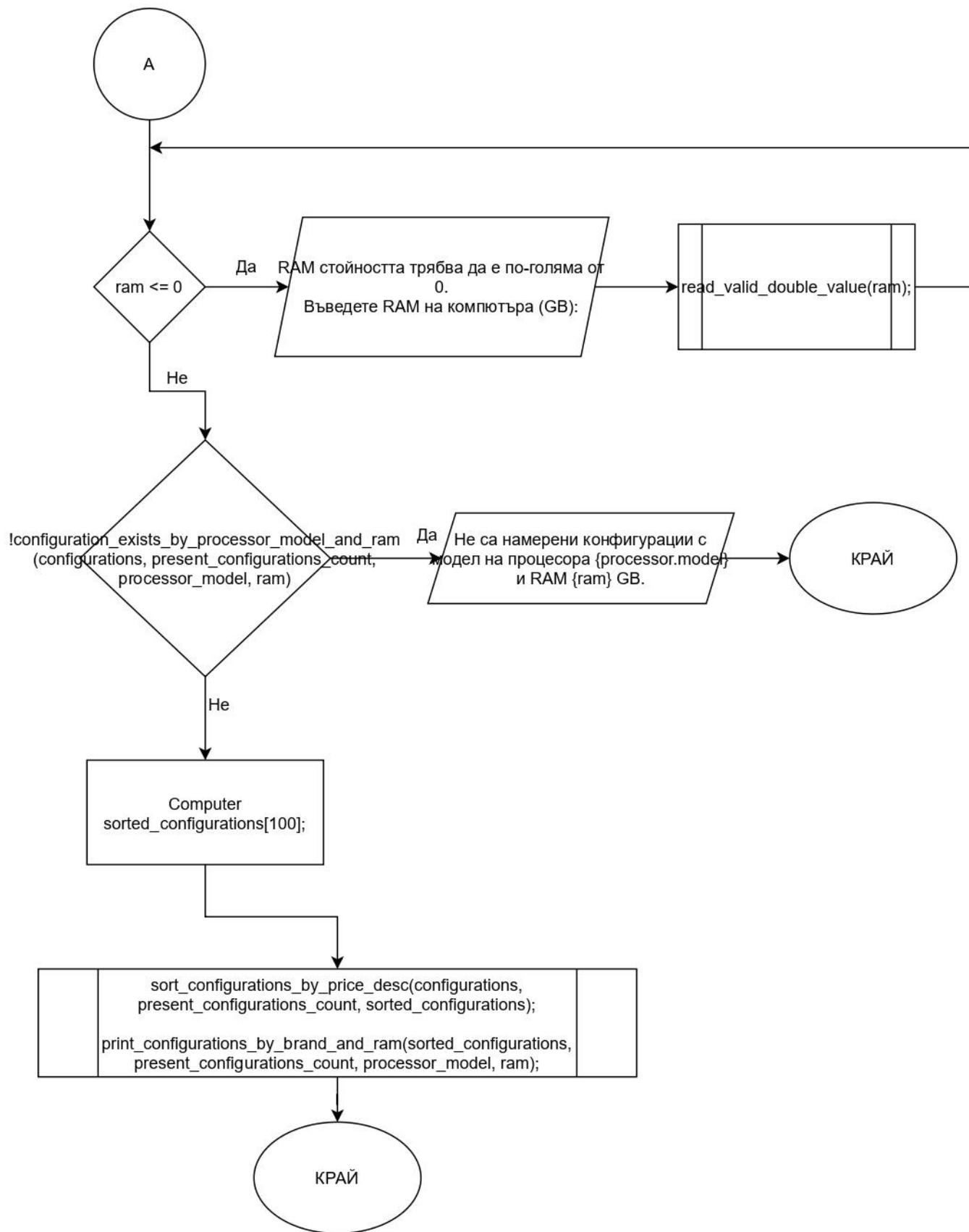


```
void print_configurations_by_brand_and_ram(Computer configurations[], int present_configurations_count, string& processor_model,  
                                         double ram);
```

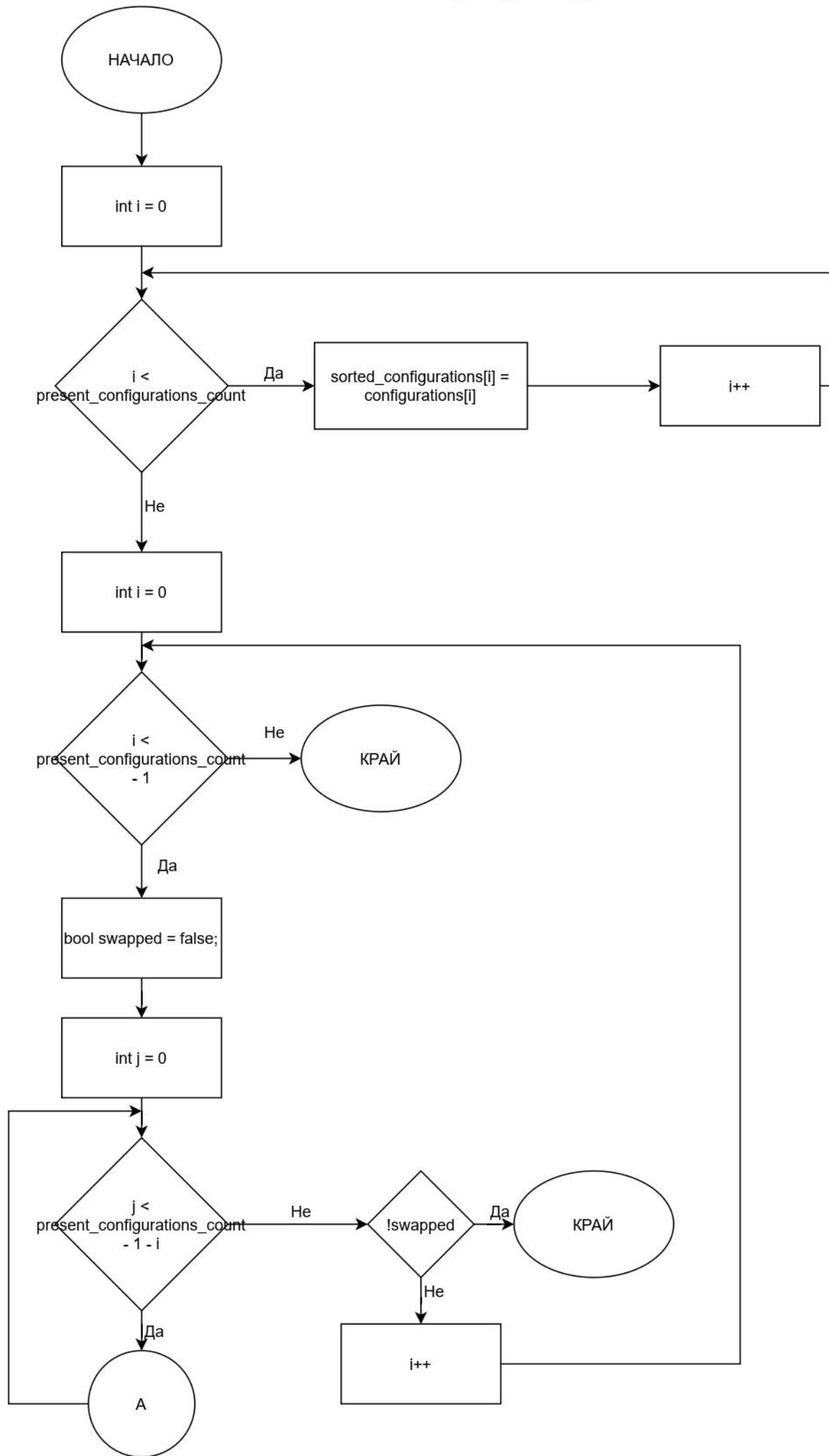


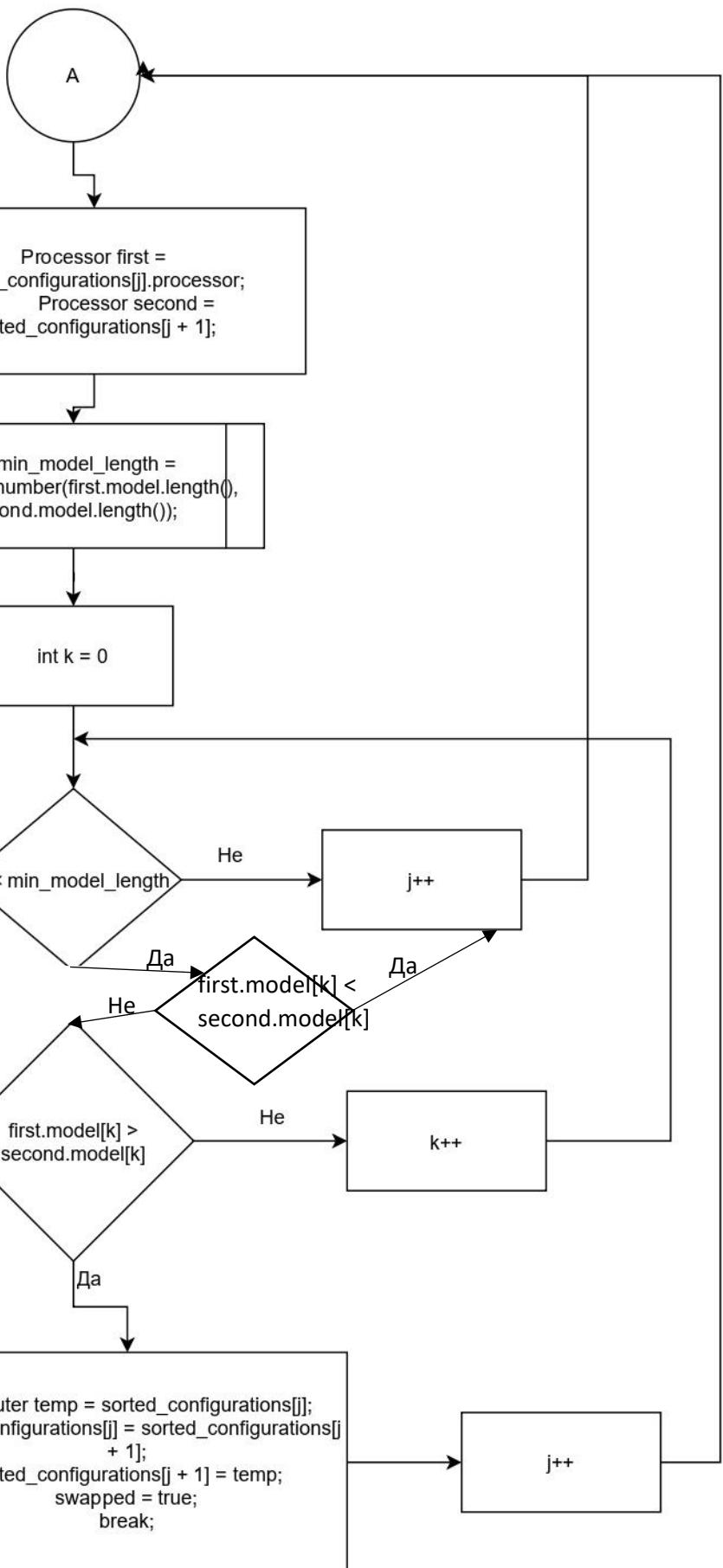
```
void print_configurations_by_brand_and_ram_sorted_by_price_desc(Computer configurations[], int present_configurations_count);
```



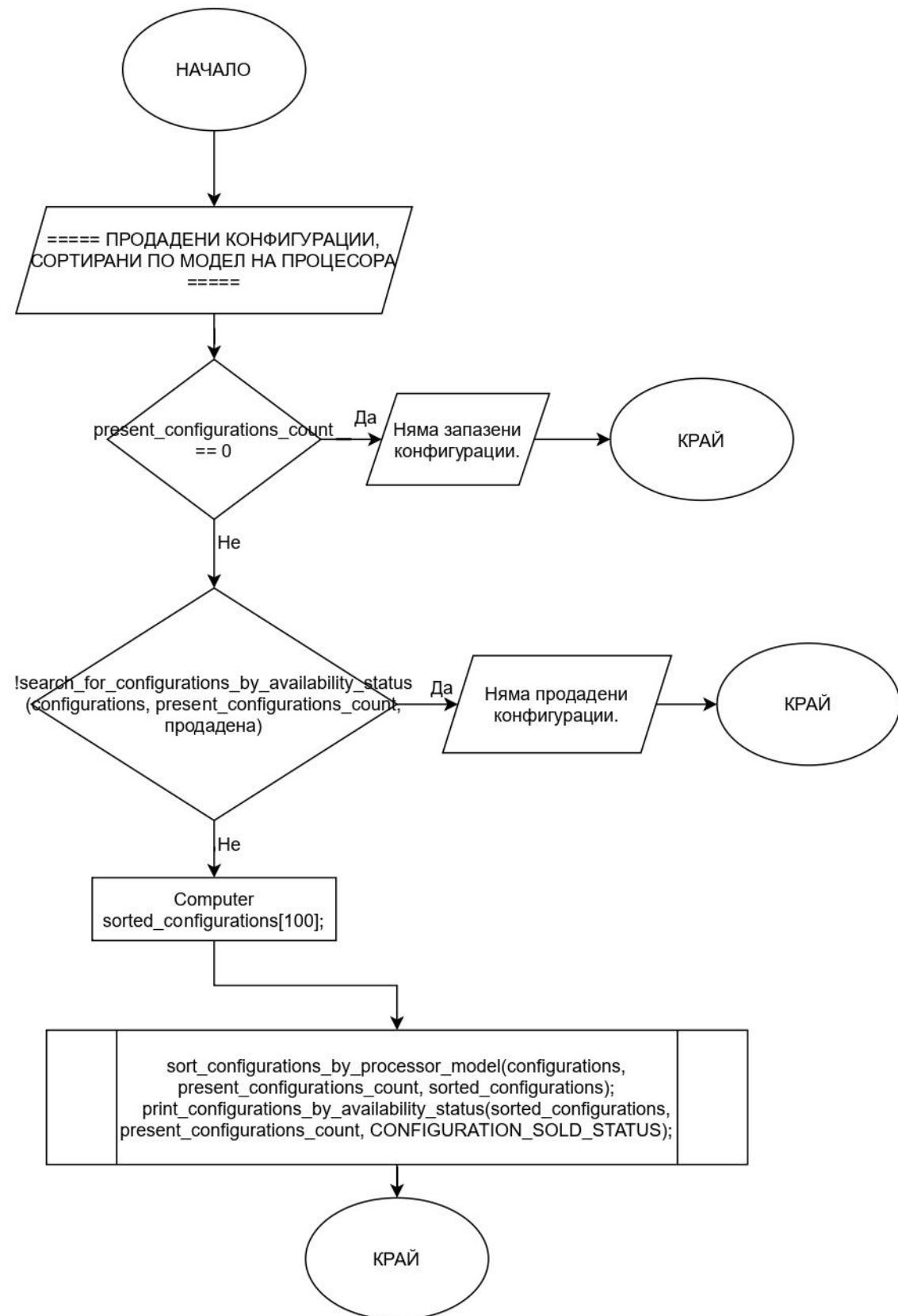


```
void sort_configurations_by_processor_model(Computer configurations[], int present_configurations_count, Computer sorted_configurations[]);
```

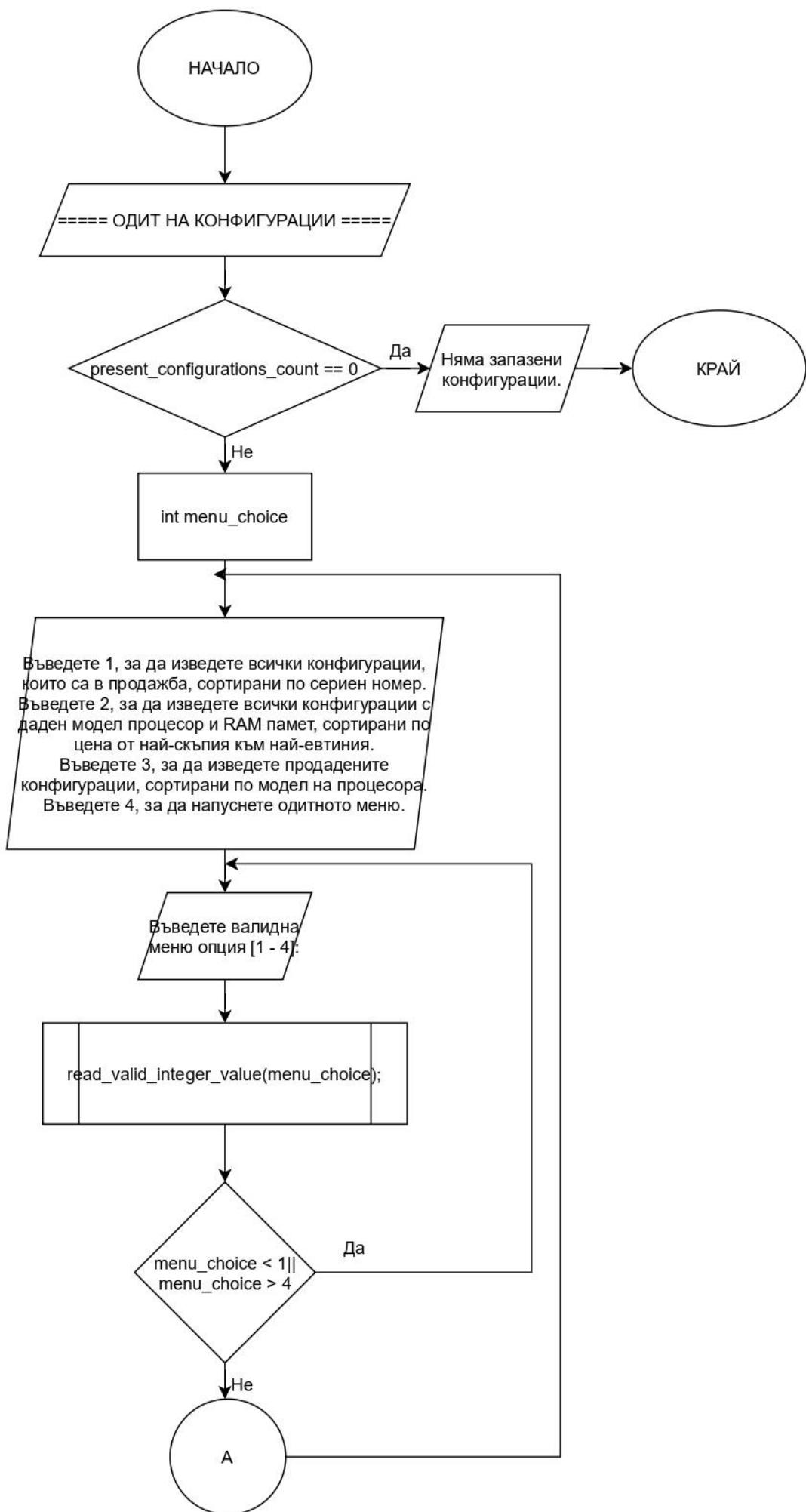


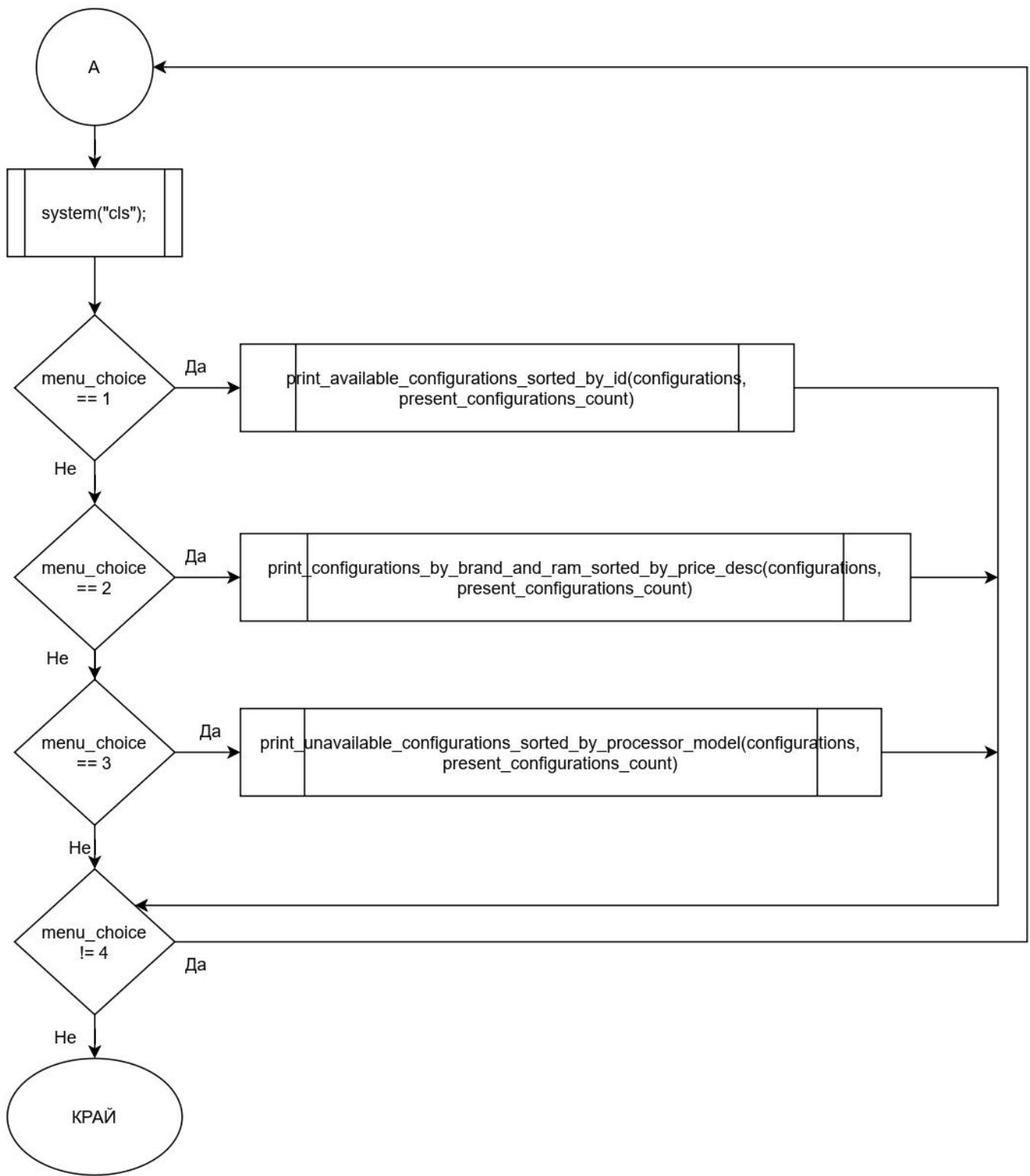


```
void print_unavailable_configurations_sorted_by_processor_model(Computer configurations[], int present_configurations_count);
```



```
void make_configurations_audit(Computer configurations[], int present_configurations_count)
```





Екранни снимки с примерен вход и изход

- а. Изглед след сортиране на наличните конфигурации по сериен номер

===== НАЛИЧНИ КОНФИГУРАЦИИ, СОРТИРАНИ ПО СЕРИЕН НОМЕР =====

Сериен номер: A3-B3-83-92-55-C2

Марка: Asus

Модел: Vivobook

RAM памет: 16,00 GB

Процесор:

Производител: AMD

Модел: Ryzen 5

Тактова честота: 5,00 GHz

Брой ядра: 6

Цена: 2500,00 лв.

Наличен статус: в продажба

Сериен номер: A7-87-F9-83-31-A1

Марка: HP

Модел: Omen

RAM памет: 8,00 GB

Процесор:

Производител: Intel

Модел: i5

Тактова честота: 3,00 GHz

Брой ядра: 4

Цена: 1200,00 лв.

Наличен статус: в продажба

Сериен номер: C5-58-64-54-29-C1

Марка: Acer

Модел: Nitro 5

RAM памет: 16,00 GB

Процесор:

Производител: Intel

Модел: i7

Тактова честота: 5,00 GHz

Брой ядра: 6

Цена: 2199,00 лв.

Наличен статус: в продажба

- b. Излгед след извеждане на всички конфигурации с даден модел процесор и RAM памет, сортирани по цена от най-скъпата към най-евтиния

===== КОНФИГУРАЦИИ С ДАДЕН МОДЕЛ ПРОЦЕСОР И RAM ПАМЕТ, СОРТИРАНИ ОТ НАЙ-СКЪПАТА КЪМ НАЙ-ЕВТИНАТА =====

Въведете желан модел процесор: Ryzen 7

Въведете RAM на компютъра (GB): 16

Сериен номер: 38-8B-34-1F-8A-CB

Марка: Apple

Модел: MacbookPro 13

RAM памет: 16,00 GB

Процесор:

Производител: AMD

Модел: Ryzen 7

Тактова честота: 5,00 GHz

Брой ядра: 6

Цена: 3200,00 лв.

Наличен статус: продадена

Сериен номер: A3-B3-83-92-55-C2

Марка: Asus

Модел: Vivobook

RAM памет: 16,00 GB

Процесор:

Производител: AMD

Модел: Ryzen 7

Тактова честота: 5,00 GHz

Брой ядра: 6

Цена: 2500,00 лв.

Наличен статус: в продажба

- c. Изглед след извеждане на продадените конфигурации, сортирани по модел на процесора

===== ПРОДАДЕНИ КОНФИГУРАЦИИ, СОРТИРАНИ ПО МОДЕЛ НА ПРОЦЕСОРА =====

Сериен номер: 4C-A0-77-7F-06-C8

Марка: Apple

Модел: MacbookPro 15

RAM памет: 32,00 GB

Процесор:

Производител: Apple

Модел: M1

Тактова честота: 5,00 GHz

Брой ядра: 12

Цена: 4200,00 лв.

Наличен статус: продадена

Сериен номер: 38-8B-34-1F-8A-CB

Марка: Apple

Модел: MacbookPro 13

RAM памет: 16,00 GB

Процесор:

Производител: AMD

Модел: Ryzen 7

Тактова честота: 5,00 GHz

Брой ядра: 6

Цена: 3200,00 лв.

Наличен статус: продадена

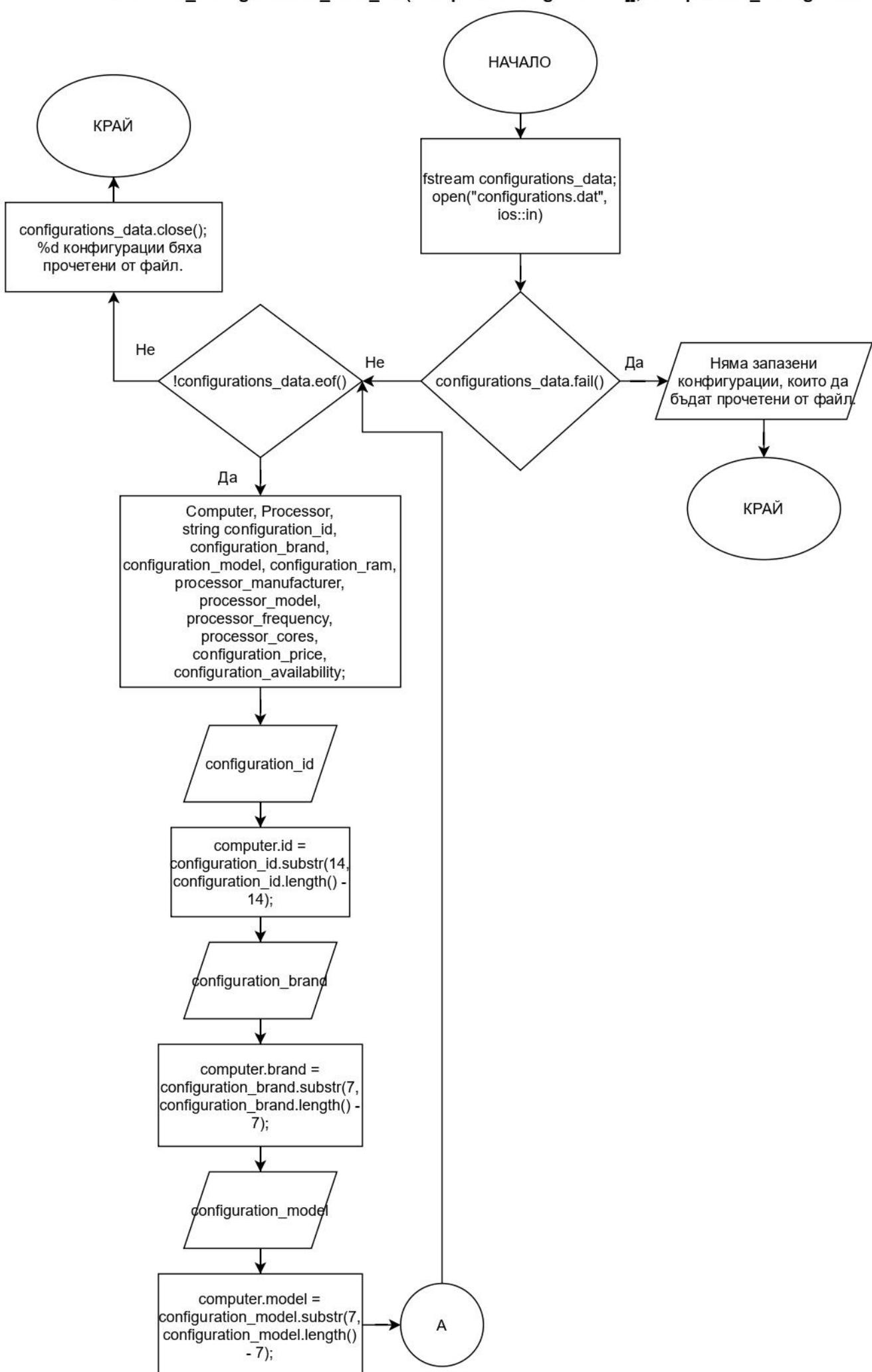
G. Данните в програмата да могат да се запазват във файл между две стартирания на програмата

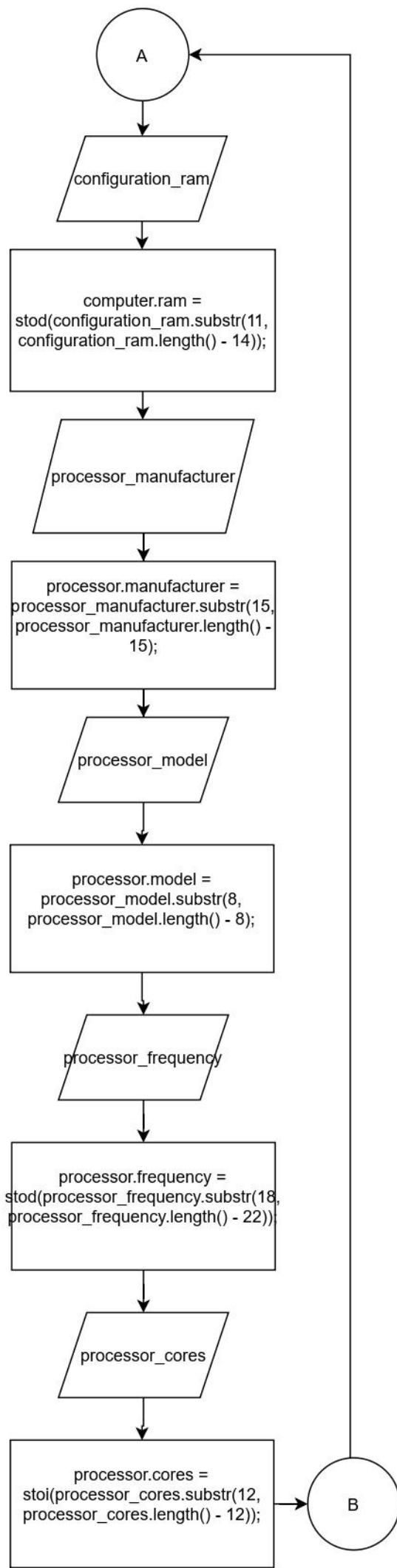
```
void read_configurations_from_file(Computer configurations[], int& present_configurations_count);  
void store_configurations_in_file(Computer configurations[], int present_configurations_count);
```

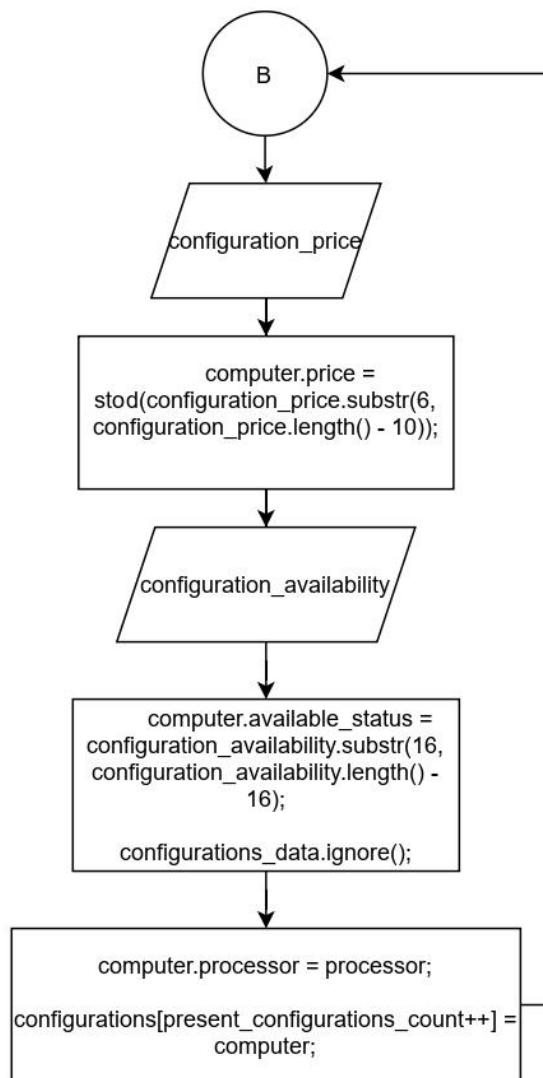
Функцията *read_configurations_from_file* прочира конфигурациите от файл и ги записва в масива за съхранение на информация. Приема като параметри масива за записване на конфигурациите и броят на запазените конфиграции. Дефинира променлива от тип fstream и отваря файл с име configurations.dat. Ако файлът не се отвори успешно се извежда съобщение за грешка и функцията прекратява изпълнението си. При успешно отваряне на файла, той се прочита целия като за всяка записана конфигурация се създава променлива от тип структура Computer и се добавя към масива. Накрая четенето от файла се затваря и се извежда броят на конфигурациите прочетени от файла.

Функцията *store_configurations_in_file* записва съществуващите конфигурации във файл configurations.dat. Приема като параметри масива за записване на конфигурациите и броят на запазените конфиграции. Дефинира променлива от тип fstream и отваря файл с име configurations.dat. Ако няма запазени конфигурации или файлът не се отвори успешно се извежда съобщение за грешка и функцията прекратява изпълнението си. При успешно отваряне на файла, информацията за всички конфигурации се запазва в него.

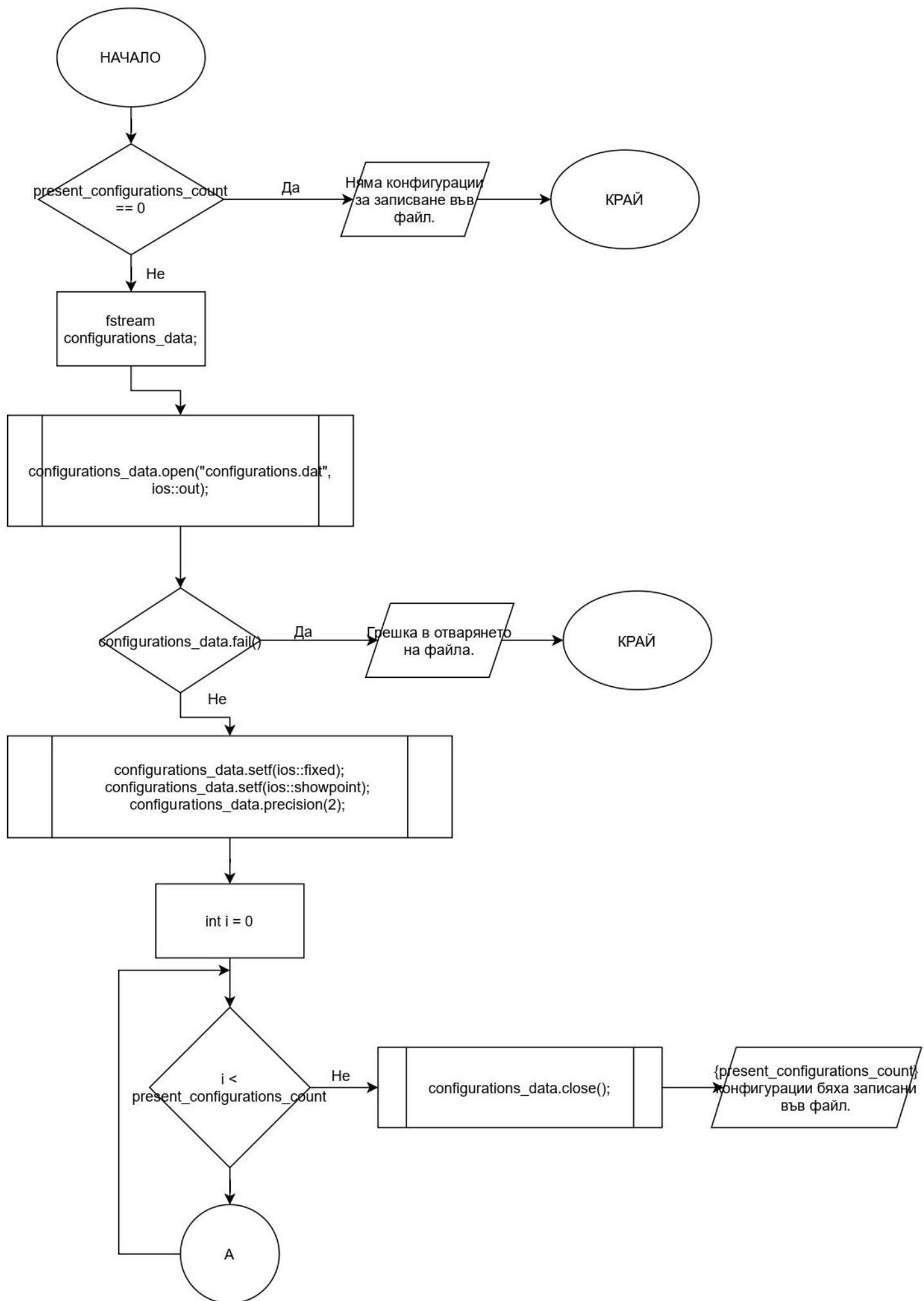
```
void read_configurations_from_file(Computer configurations[], int& present_configurations_count)
```

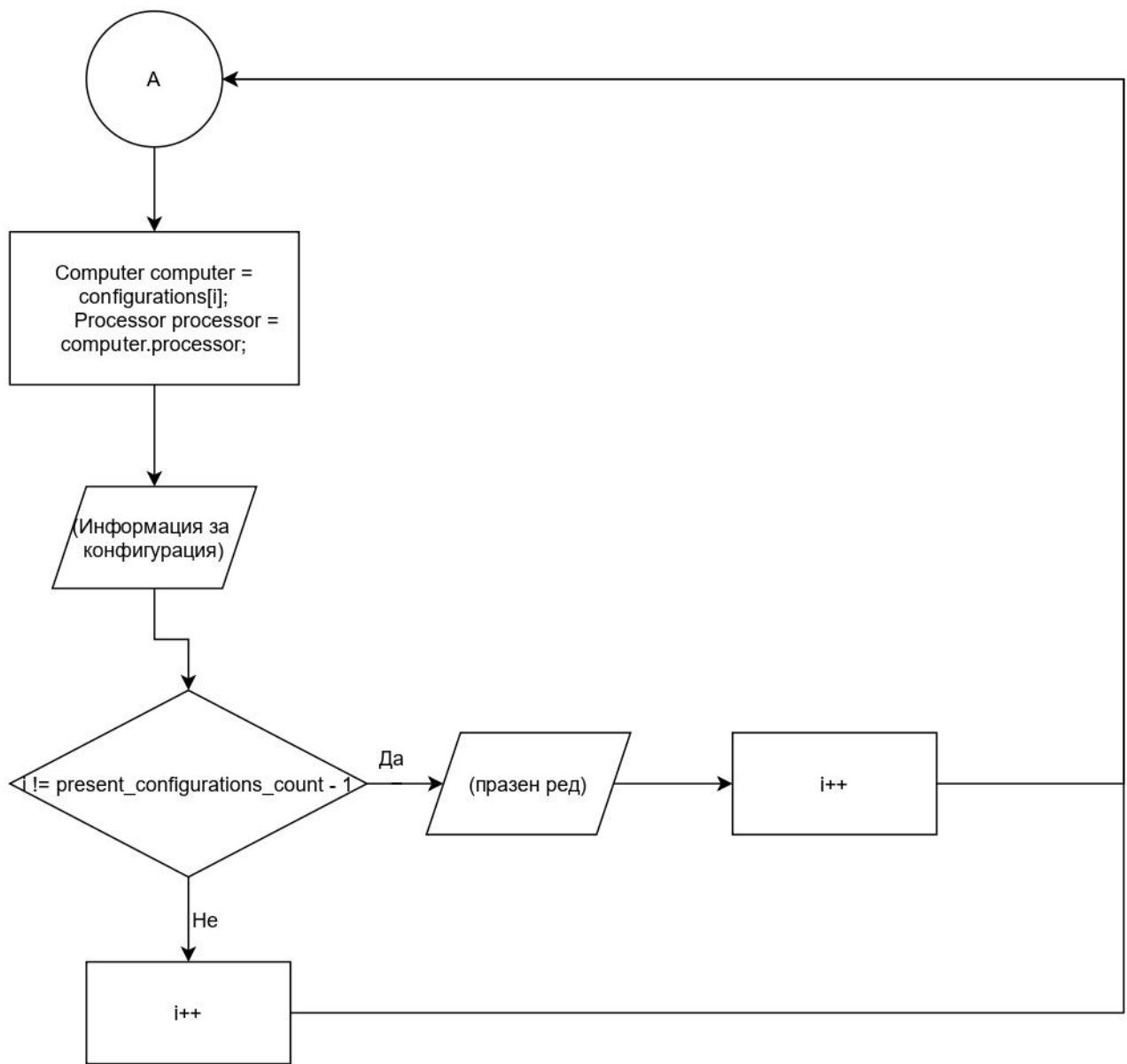






```
void store_configurations_in_file(Computer configurations[], int present_configurations_count);
```





Екранни снимки с примерен вход и изход

- a. Изглед след успешно прочитане на конфигурации от файл

5 конфигурации бяха прочетени от файл.

- b. Изглед след успешно записване на конфигурации във файл

5 конфигурации бяха записани във файл.