**РУСЕНСКИ УНИВЕРСИТЕТ “АНГЕЛ КЪНЧЕВ”**

**КУРСОВА РАБОТА**

по

САА

Студент:

Факултетен номер:

Група:

Специалност:

Дата: ПРОВЕРИЛ:.....................

РУСЕ

2015 г.

# Задача.

**19**. Японски учени са измислили най-добрия симулатор на шах. За да успеят българските им колеги да ги задминат им е необходим алгоритъм, реализиращ създаването на динамичен списък със следните условия: в началото, списъкът се запълва с 32-те фигури, използвани на шахматната дъска. Елементите на списъка съдържат името на фигурата (пример: „purwi\_4eren\_kon“) и позицията (пример: „b4“). При движение на фигура, тя трябва да променя своята позиция (която вече е зададена в списъка). При отнемане на фигура на някой от състезателите, тя трябва да се премесва от списъка.

Да се състави програма, съдържаща следните подпрограми:

* подпрограма за начално въвеждане на всички фигури в списъка (може и да не се въвеждат всичките 32 фигури);
* подпрограма за извеждане на наличните в списъка фигури;
* подпрограма за промяна на местоположението на дадена фигура. Ако дадена фигура застане на мястото, на което до този момент е била друга фигура, то втората се премахва от списъка.

# Описание на решението.

Информацията за новата фигура за добавяне се въвежда в главната програма main(), след което се предават като параметри на функцията add() и фигурата се добавя в края на списъка, т.е. списъкът е от тип опашка (FIFO).

Изтриването на фигура от списъка става чрез функцията move(). Като параметър се подават главата на списъка като В/И параметър, текущата и новата позиция на фигурата. Първо се прави проверка дали има фигури на двете позиции (чрез функцията find(). Ако фигурата е намерена се променя позицията й и ако е имала друга фигура на тази позиция – тя се изтрива. Използва се схема „Изтриване след елемент, сочен от даден указател P“.

**Постъпково изпълнение на кода:**

*Информацията в елементите е примерна!*

1. След изпълнение на оператора

Pde = P->next;

състоянието на списъка е:

**…**

**12**

**5**

**6**

**7**

**…**

**P**

**Pdell**

1. След изпълнение на оператора

P->next = P->next->next;

състоянието на списъка е:

**…**

**12**

**5**

**6**

**7**

**…**

**P**

**Pdel**

1. След изпълнение на оператора

дelete Pde;

състоянието на списъка е:

**…**

**12**

**5**

**6**

**…**

**P**

# Описание на използваните функции.

**(1)**

Вид на функцията: **int main();**

Действие: **главна програма (функция);**

Параметри: **няма;**

Използвани глобални променливи: **няма;**

Извиквани функции: **add(), print(), move();**

**(2)**

Вид на функцията: **void add(PPiece\* head\_ref, string name, string position)**

Действие: **подпрограма за добавяне на нов елемент в списъка;**

Параметри: **PPiece\* head\_ref** – указател към началото на списъка;

**string name** – името на фигурата; **string position** – позицията на фигурата;

Използвани глобални променливи: **няма;**

Извиквани функции: **няма***;*

**(3)**

Вид на функцията: **void print(PPiece head\_ref)**

Действие: **подпрограма за извеждане на списъка;**

Параметри: **PPiece head\_ref** *– указател към началото на списъка;*

Използвани глобални променливи: **няма;**

Извиквани функции: **няма**;

**(4)**

Вид на функцията: **PPiece find(PPiece head\_ref, string position)**

Действие: **подпрограма за търсене на фигура по зададена позиция;**

Параметри: **PPiece head\_ref** *– указател към началото на списъка;*

**string position** – позицията;

Използвани глобални променливи: **няма;**

Извиквани функции: **няма*;***

**(5)**

Вид на функцията: **void move(PPiece &head\_ref, string fromPosition, string toPosition)**

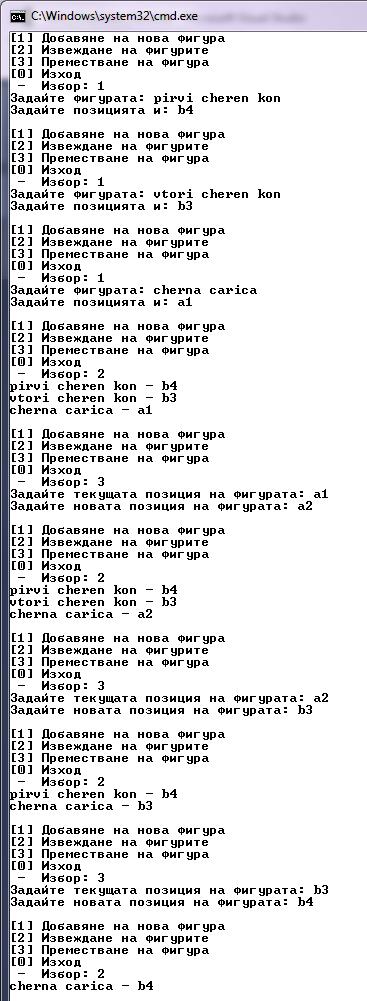
Действие: **подпрограма за преместване на фигура;**

Параметри: **PPiece &head\_ref** *– указател към началото на списъка;* **string fromPosition** – *текущата позиция на фигурата;* **string toPosition** – *позицията, на която трябва да се премести фигурата;*

Използвани глобални променливи: **няма;**

Извиквани функции:**find()***;*

# Тестов пример.



# Код на програмата.

#include <iostream>

#include <string>

#include <locale>

using namespace std;

typedef struct Piece

{

string name;

string position;

Piece \*next;

} \*PPiece;

void add(PPiece\* head\_ref, string name, string position)

{

PPiece current, new\_node;

new\_node = new Piece;

new\_node->name = name;

new\_node->position = position;

if (\*head\_ref == NULL)

{

new\_node->next = \*head\_ref;

\*head\_ref = new\_node;

}

else

{

current = \*head\_ref;

while (current->next != NULL)

current = current->next;

new\_node->next = current->next;

current->next = new\_node;

}

}

void print(PPiece head\_ref)

{

while (head\_ref)

{

cout << head\_ref->name << " - "

<< head\_ref->position << endl;

head\_ref = head\_ref->next;

}

}

PPiece find(PPiece head\_ref, string position)

{

while (head\_ref)

{

if (head\_ref->position == position)

{

break;

}

head\_ref = head\_ref->next;

}

return head\_ref;

}

void move(PPiece &head\_ref, string fromPosition, string toPosition)

{

PPiece p = head\_ref, pdel;

PPiece fromP = find(head\_ref, fromPosition);

PPiece toP = find(head\_ref, toPosition);

if (fromP)

{

fromP->position = toPosition;

if (toP)

{ // ако има друга фигура на новата позиция

if (head\_ref == toP)

{ // ако елементът за изтриване е първи

head\_ref = head\_ref->next;

delete toP;

}

else

{

while (p)

{

if (p->next == toP)

{

pdel = p->next;

p->next = p->next->next;

delete pdel;

break;

}

p = p->next;

}

}

}

}

else

{

cout << endl << "Фигурата не е намерена!";

}

}

int main()

{

setlocale(LC\_ALL, "bgr");

PPiece head = NULL;

string name, position, toPosition;

int choice;

do {

cout << "[1] Добавяне на нова фигура" << endl;

cout << "[2] Извеждане на фигурите" << endl;

cout << "[3] Преместване на фигура" << endl;

cout << "[0] Изход" << endl;

cout << " - Избор: ";

cin >> choice;

cin.ignore();

switch (choice) {

case 1:

cout << "Задайте фигурата: ";

getline(cin, name);

cout << "Задайте позицията и: ";

cin >> position;

add(&head, name, position);

break;

case 2:

print(head);

break;

case 3:

cout << "Задайте текущата позиция на фигурата: ";

cin >> position;

cout << "Задайте новата позиция на фигурата: ";

cin >> toPosition;

move(head, position, toPosition);

break;

}

cout << endl;

} while (choice != 0);

return 0;

}