Języki Programowania, Czwartek 28.11.2013
Zadanie 7

Zaimplementujemy prostą planszę szachową (załóżmy, że dowolnego rozmiaru podawanego jako argument uruchomienia programu) na której wstawimy pionki, które będą mogły poruszać się tylko w górę o 1 pole.
Załóżmy, że plansza ma postać macierzy kwadratowej NxN, gdzie pole 0 oznacza, że nie znajduje się na nim pionek, zaś 1, że na danym polu stoi pionek. Ruch pionka oznacza, że pole na którym stał pionek zmienia się z 1 na 0, zaś to na które pionek się przemieści zmienia się z 0 na 1. Załóżmy, że nasza domyślna plansza będzie następującej postaci (w funkci `main` przypisujemy 0 i 1 odpowiednim elementom tablicy):

```
Rozmiar planszy: 8
   0 1 2 3 4 5 6 7
   ---------------
0| 0 0 0 0 0 0 0 0
1| 0 0 0 0 0 0 0 0
2| 0 1 0 1 0 1 0 1
3| 1 0 1 0 1 0 1 0
4| 0 0 0 0 0 0 0 0
5| 0 0 0 0 0 0 0 0
6| 0 0 0 0 0 0 0 0
7| 0 0 0 0 0 0 0 0
```

Tworzymy klasę `ChessPiece` (figura szachowa).
Klasa powinna zawierać następujące pola:

```
int PieceCounter; //licznik ilosci figur (pole statyczne), domyslnie 0
int fX, fY; //pozycja figury na planszy
char *fName; //nazwa figury, dynamiczna tablica znakowa char
```

konstruktor:

- przyjmujący trzy parametry ustawiające pole `fName`, `fX` oraz `fY`. Zwiększa również licznik ilości figur.

destruktor:

- zmniejsza licznik pionków

oraz:

- `SetPosition(int x, int y); //ustawia pozycje figury`
- `GetPositionX(); //zwraca pozycje x (pole fX)`
- `GetPositionY(); //zwraca pozycje y (pole fY)`
- `GetName(); //zwraca nazwe figury (pole fName)`
- `Move(int **tab); //Przyjmuje plansze i jej rozmiar, przesuwa pionek w gore planszy o 1 (z wiersza o większym numerze na wiersz o mniejszym numerze)`
- `operator[]; //przeciążony operator [] - dla '1' zwraca fX, dla dowolnych pozostałych wartości zwraca fY; zdefiniowany tak by stał po obu stronach znaku =`

Poniżej znajduje się zawartość pliku **main.cpp**. W pliku tym będą znajdować się również 3 funkcje globalne.

```
void PrintTable(int **tab, int size)
{

     //Funkcja wypisuje tablice na ekran
}

void SetPiece(ChessPiece **pieceTab, int **tab, int size)
{
      /* Metoda SetPiece przyjmuje przyjmuje tablice 2D, jej rozmiar oraz tablice
wskaznikow na pionki. Jej zadaniem jest stworzenie obiektu typu ChessPiece w
miejscu gdzie na planszy znajduje się 1. */
}

int main(int argc, char** argv)
{

/*
      1) Tworzymy plansze-tablice 2D (podając '8' - rozmiar tablicy, jako pierwszy
```

```
parametr uruchomienia programu) 1 p.
     2) Nastepnie wypisujemy wczytana plansze na ekran, PrintTable 1.5 p.
     3) Tworzymy tablice wskaznikow na pionki, np. ChessPiece *pieceTab[100];
        a następnie ustawiamy pionki za pomoca funkcji SetPiece 3 p.
     4) Przesuwamy pionek pieceTab[1] oraz pieceTab[4] metoda Move 4 p.
     5) Ponownie wypisujemy plansze na ekran
     6) Wypisujemy pole X oraz Y elementu pieceTab[4] za pomoca operatora [] oraz
nazwę tego pionka metoda GetNazwa 4.5 p.
     7) Usuwamy plansze-tablice 2D oraz tablice wskaznikow na obiekt ChessPiece 5
p.

  */

  return 1;
}
```

Przykładowy wynik działania programu (**pogrubionym** zaznaczono pionki 1 oraz 4, które zostały przesunięte):

```
Rozmiar planszy: 8
   0 1 2 3 4 5 6 7
   ---------------
0| 0 0 0 0 0 0 0 0
1| 0 0 0 0 0 0 0 0
2| 0 1 0 1 0 1 0 1
3| 1 0 1 0 1 0 1 0
4| 0 0 0 0 0 0 0 0
5| 0 0 0 0 0 0 0 0
6| 0 0 0 0 0 0 0 0
7| 0 0 0 0 0 0 0 0

Wynik po ruszeniu dwoch pionkow
Rozmiar planszy: 8
   0 1 2 3 4 5 6 7
   ---------------
0| 0 0 0 0 0 0 0 0
1| 0 0 0 1 0 0 0 0
2| 1 1 0 0 0 1 0 1
3| 0 0 1 0 1 0 1 0
4| 0 0 0 0 0 0 0 0
5| 0 0 0 0 0 0 0 0
6| 0 0 0 0 0 0 0 0
7| 0 0 0 0 0 0 0 0
```

Uwaga!

1. Sposób alokowania pamięci tablicy dwuwymiarowej (dla tablicy typu `int`):

```
int rozmiar = 100;
int **tab = new int*[rozmiar];
for(int i=0;i<rozmiar;i++)
  tab[i] = new int[rozmiar];
```

2. Sposób zwalniania pamięci tablicy dwuwymiarowej:

```
for(int i=0;i<rozmiar;i++)
  delete[] tab[i];
delete[] tab;
```