

**Wydział Zarządzania**

**Projekt zaliczeniowy z programowanie komputerów**

**Przejście pionkiem po tablicy (16)**

**Autorzy:** **Tomasz Sałapatek, Łukasz Maczek**

**Kierunek studiów:** **Informatyka i Ekonometria**

**Kraków, 2023**

Dzień dobry, w naszym projekcie podjęliśmy się problemu poruszania się pionka po planszy oraz zliczania możliwych przez niego tras do przejścia. Projekt został wykonany podczas wspólnych spotkań, zatem ciężko jest określić dokładny podział pracy.

**Program do działania potrzebuje poniższych bibliotek:**

* **<stdio.h>** - kluczowa funkcja do działania programu, zawiera funkcje do operacji wejścia/wyjścia
* **<math.h>** - potrzebna do użycia funkcji z pierwiastkiem
* **<stdbool.h>** - potrzebna przy funkcji isValidmove, definiuje typ logiczny bool oraz dwie stałe, true i false
* **<conio.h>** - potrzebna do użycia funkcji ‘\_getch’

**Program zawiera następujące konieczne do działania funkcje:**

* **WYBOR** – Jedyna makrofunkcja w naszym programie, odpowiada
* **Odczyt** - Funkcja służąca do wczytania planszy z pliku oraz przypisania jej do tablicy,
* **printBoard** - Funkcja służąca do pokazywania planszy graczowi,
* **IsValidmove** – Funkcja sprawdzająca czy ruch gracza jest dozwolony,
* **movePlayer** – Funkcja przenosząca pionek w wybrane miejsc,
* **IloscTras** – Funkcja, która liczy ilość możliwych tras do przejścia,
* **spr\_dl** – Funkcja sprawdzająca ilość znaków w pliku z planszą.

**Poniżej dokładniej wyjaśnimy wybrane nietrywialne funkcje:**

**printBoard –** Funkcja oparta jest na konkretnej budowie planszy, która składa się ze znaków   
{0, X, S, G} które następnie podmienia na odpowiadające im pola na planszy. ‘X’ – czerwone niedostępne pole, ‘0’ – puste pole, ‘S’ – pole startowe oraz ‘G’ – pole mety. Pionek gracza zaakcentowaliśmy niebieskim kolorem oraz napisem ‘PL’ od słowa ‘player’. W celu czytelności po każdorazowym ruchu gracza, włączana jest komenda system(“cls”) służąca do odświeżania ekranu.

**MovePlayer –** Funkcja mająca na umożliwienie ruchu pionkiem. Zbudowana jest głównie na podstawie funkcji ‘switch’ oraz ‘\_getch’. Ruch gracza jest możliwy we wszystkich 8 kierunkach. Gracz wciska przycisk odpowiadający kierunkowi, w który chce się przemieścić, następnie do kolumny bądź wiersza jest dodawana konkretna wartość. Co ważne funkcja nie zawiera sprawdzenia czy dany ruch jest możliwy. Aby przemieszczenie było natychmiastowe, została użyta funkcja ‘\_getch’.

**IloscTras –** Funkcja ‘IloscTras’ działa na bazie algorytmu zliczania możliwych tras, który zakłada następujące warunki: 1. pole startowe znajduje się zawsze w lewym górnym rogu, a meta w prawym dolnym rogu planszy, 2. gracz porusza się zawsze w stronę mety (nie cofa się, nie wybiera trasy na około)  
Po spełnieniu powyższych warunków, algorytm działa w następujący sposób: na początku w polu startowym wpisuje liczbę 1, następnie sprawdza pierwszy wiersz i pierwszą kolumnę, jeśli w danym polu występuje ‘X’, który oznacza pole niedostępne, to wpisujemy w to pole 0, w innym wypadku, wpisujemy liczbę z pole wcześniejszego, co pozwala na wpisanie 1, jeśli do pola można dojść z pola startowego przy spełnieniu podanych warunków, lub 0, jeśli pole znajduje się po ‘X’, lub samo jest ‘X’. Kiedy już pola w pierwszym wierszu i pierwszej kolumnie są uzupełnione, pozostaje nam uzupełnienie pozostałych pól tablicy. Algorytm działa w taki sposób, że w dane pole wstawia sumę pól znajdujących się bezpośrednio na lewo, w górę i po skosie w lewym górnym rogu. Kiedy w taki sposób zapełnimy całą planszę, w polu mety powinniśmy otrzymać liczbę, która oznacza ilość możliwych tras do przejścia, liczba ta jest zwracana przez opisywaną funkcję ‘IloscTras’.

**spr\_dl** – Funkcja zlicza ilość znaków planszy z pliku. Następnie pierwiastkuje wynik i zwraca go. Warto się pochylić nad problem, który napotkaliśmy po drodze, W momencie pierwiastkowania zmienna jest przekonwertowywana na typ double. Na podstawie tej zmiennej jest później budowana tablica, więc musieliśmy zmienić jej typ na int. W związku z tym pojawia się błąd z możliwą utratą danych. Jest to jednak ostrzeżenie, które nie wiąże się z błędnym działaniem programu, ponieważ przy kwadratowych planszach, to jest takich jakie zbudowaliśmy, ilość znaków jest zawsze możliwa do spierwiastkowania bez liczb po przecinku, zatem przy konwersji nie utracimy żadnych znaczących danych.

**Program działa w następujący sposób:**

* Na początku jest wypisywana instrukcja dla gracza, dotycząca klawiszy sterowania, zasad oraz możliwy wybór jednej z czterech plansz po naciśnięciu odpowiedniego numeru.
* Następnie wybrana plansza jest wczytywana przy wykorzystaniu funkcji ‘odczyt’ oraz wyświetlana przy użyciu funkcji ‘printBoard’.
* Następnie przechodzimy do pętli ‘while’, w której jesteśmy, aż do momentu znalezienia się pionka na pozycji końcowej (meta). W podanej pętli wypisujemy legendę przypominającą graczowi oznaczenia pól. Potem wywołujemy funkcję ‘movePlayer’, a następnie sprawdzamy funkcją ‘isValidMove’ czy podany ruch jest odpowiedni. Jeśli tak, pozycja pionka zostaje zmieniona na podaną przez gracza, w innym wypadku pionek wraca na pole startowe.
* Program po wyjściu z pętli wypisuje komunikat o ukończeniu trasy oraz podaje ilość możliwych tras do przejścia, przy użyciu funkcji ‘IloscTras’, a następnie kończy działanie.