Zajęcia 11

Temat: Rekurencja. Metoda nawrotów

Czas trwania: 2x45 min

Cel zajęć:

projektuje i programuje proste problemy z różnych dziedzin, stosuje przy tym: instrukcje wejścia/wyjścia, wyrażenia arytmetyczne i logiczne, instrukcje warunkowe, instrukcje iteracyjne, tablice dwuwymiarowe, rekurencję, indukcję, pisze własne funkcje rekurencyjne, umie wyznaczyć liczby Fibonacciego, testuje poprawność programów dla różnych danych, posługuje się zintegrowanym środowiskiem programistycznym przy pisaniu, uruchamianiu i testowaniu programów;

Efekty:

- umie uruchomić potrzebne oprogramowanie,
- umie napisać program z wykorzystaniem funkcji rekurencyjnej, równaniami rekurencyjnymi,
- zna porządek leksykograficzny,
- zna przeszukiwanie drzew ukorzenionych,
- zna liczby Fibonacciego (potrafi obliczyć),

Formy i metody pracy: praca samodzielna, wykład, omówienie

Zadania do wykonania na zajęciach	Treści programowe
1. Roztargniony profesor	M.3, P.2.16, P.2.17, P.2.19, A.3.5, A.3.6
2. Mysz w labiryncie	M.3, P.2.16, P.2.17, P.2.19, A.3.5, A.3.6
3. Problem 8 hetmanów	M.3, P.2.16, P.2.17, P.2.19, A.3.5, A.3.6

Materiały do zajęć:

https://www.main2.edu.pl/main2/courses/show/7/8/

https://www.main2.edu.pl/main2/courses/show/7/18/

Zadania do wykonania w domu:

Czy koń doskoczy (gotowa paczka do SIO2)

ZADANIA I ROZWIĄZANIA:

Zadanie 1. Roztargniony profesor

Dostępna pamięć: 64MB

Pewien profesor, aby dostać się do swego gabinetu, musi pokonać N schodów. Profesor pokonuje zazwyczaj jeden lub dwa schody na raz. Oblicz, na ile różnych sposobów profesor może pokonać schody. Podaj te sposoby.

Wejście

Jedna liczba całkowita N (1 ≤ N ≤ 25) - liczba schodów.

Wyjście

Wyjście zawiera w pierwszym wierszu liczbę możliwych kombinacji, w kolejnych wierszach wszystkie możliwe kombinacje w porządku leksykograficznym.

Przykład

Wejście	Wyjście
3	3 1 1 1 1 2 2 1

Rozwiązanie

Profesor zgodnie z treścią zadania ma do pokonania N schodów. Napiszmy procedurę profesor, które będzie symulowała kolejne ruchy profesora. Jakie kroki może wykonać? Zgodnie z treścią zadania pokonuje jednocześnie jeden schodek (wówczas zostanie mu do pokonania jeszcze N-1 schodów) lub 2 schody (analogicznie pozostanie jeszcze N-1 schodów) – oczywiście profesor wykona ruch, jeżeli pozostała przed nim wystarczająca liczba schodów. Jak zapamiętać wszystkie sposoby, na które profesor się porusza? Dodajmy jako parametr funkcji listę dotychczas wykonanych ruchów i po każdym kroku dodawajmy kolejny. Po tym, jak dojdziemy na sam koniec schodów (pozostanie 0 schodów), wypiszmy uzyskaną kombinację oraz zliczmy kolejny sposób dojścia na koniec schodów.

```
ile_kombinacji = 0

profesor (N, ruchy)
     jeżeli (N = 1)
          ile_kombinacji ← ile_kombinacji + 1
          wypisz ruchy
     jeżeli (N ≥ 1) profesor (N-1, ruchy + ”1”)
     jeżeli (N ≥ 2) profesor (N-2, ruchy + ”2”)
```

Zadanie dla uczniów: Na ile sposobów profesor może przejść schody? Jaki znany ciąg opisuje tę liczbę?

Zadanie 2. Mysz w labiryncie

Dostępna pamięć: 32MB

Do labiryntu trafiła mysz. Pomóż jej odnaleźć wyjście!

Dane:

W pierwszej linii wejścia znajdują się dwie liczby całkowite n i m (2 ≤ n, m ≤ 1000) oznaczające rozmiar labiryntu: n kolumn i m wierszy. W kolejnych liniach znajdują się znaki oznaczające kolejno:

- `-` - możliwość przejścia (korytarz);
- `x` - brak przejścia (ściana);

- `o` - początkowe położenie myszy;
- `w` - wyjście (końcowe położenie myszy).

Wynik

Pierwszy i jedyny wiersz wyjścia powinien wypisać „**TAK**” – jeśli istnieje wyjście z labiryntu, lub „**NIE**” – jeżeli warunek nie będzie spełniony.

Wejście	Wyjście
8 8 w--x-x-- xx-x-x-- ---x---- --xxxx-- --x--x-- -----x-o xxxx---- ------x-	TAK

Rozwiązanie

Umieśćmy mysz i labirynt w kwadratowej tablicy `t[n+2][m+2]`. Miejsca, gdzie stoją ściany, oznaczmy wartością 1. Podobnie dodajmy ściany dookoła obszaru, po którym porusza się mysz (będzie to nasz wartownik – wartości 1 w pierwszym i ostatnim wierszu oraz kolumnie). Teraz wystarczy miejsce, gdzie mysz stoi (wejście), oznaczyć jako odwiedzone (wartość 1) i spróbować przejść rekurencyjnie w lewo, w prawo, w dół oraz w górę (jeżeli to miejsce było jeszcze nieoznaczone). Na koniec wystarczy sprawdzić, czy kiedykolwiek odwiedziliśmy miejsce o współrzędnych wyjścia (w algorytmie poniżej pominęliśmy wczytywanie).

```
ruch_myszy (X, Y)
     t[X][Y] ← 1
     jeżeli (t[X-1][Y] = 0) ruch_myszy(X-1, Y)
     jeżeli (t[X+1][Y] = 0) ruch_myszy(X+1, Y)
     jeżeli (t[X][Y-1] = 0) ruch_myszy(X, Y-1)
     jeżeli (t[X][Y+1] = 0) ruch_myszy(X, Y+1)
```

Główna część programu:

```
ruch_myszy (X_wejścia, Y_wejścia)
jeżeli (t[Xwyjścia][Ywyjścia] = 1) wypisz ”TAK”
w przeciwnym wypadku wypisz ”NIE”
```

Zadanie 3. Problem 8 hetmanów

Janek uczy się grać w szachy. Poznaje sposób poruszania się różnych figur. Najśmieszniejszy wydaje mu się styl skoków konika. Zastanawia się również, co byłoby, gdyby na szachownicy znalazło się kilka figur hetmanów. Analizuje, czy na planszy o wymiarach **n** x **n** możliwe jest takie rozstawienie **n** tych figur, aby żaden z nich się nie szachował, a jeśli tak, to na ile sposobów można to zrobić?

Wejście

n - rozmiar szachownicy, liczba naturalna mniejsza niż 15.

Wyjście

Liczba sposobów, na jakie można rozmieścić figury hetmanów na szachownicy tak, by się wzajemnie nie szachowały.

Przykład

Wejście	Wyjście
8	92

Rozwiązanie

To klasyczny problem, dla którego wynik liczbowy bardzo łatwo odnaleźć w Internecie. Warto jednak go przeanalizować jako klasyczny algorytm z nawrotami.

Dysponujemy szachownicą o rozmiarze n na n pól. Chcemy na niej rozmieścić w taki sposób n królowych, aby żadna z nich się wzajemnie nie szachowała. W tym celu ustawiamy pierwszą królową w pierwszej kolumnie w pierwszym wierszu. Przechodzimy (rekurencyjnie) do kolejnej kolumny i szukamy wiersza, który nie jest szachowany. I ponownie przechodzimy rekurencyjnie do kolejnej kolumny. Jeżeli uda się nam w ten sposób ustawić królową w n-tym wierszu, zliczamy tę sytuację jako kolejny sposób. Następnie usuwamy królową z szachownicy i próbujemy ustawić w kolejnym wierszu. Po przeanalizowaniu wszystkich wierszy w kolumnie wracamy do poprzedniej kolumny.

W jaki prosty sposób zapamiętać już zajęte pola? Dość łatwo zauważyć, że tablica `zajęty_wiersz[]` może spamiętywać wiersze, w których ustawiliśmy już hetmana. Podobna obserwacja pozawala zapamiętać zajęte przekątne.

Niech w oznacza numer badanego wiersza, zaś k – kolumny.

```
funkcja możliwe (w, k)
   zwróć zajęty_wiersz[w]=0 i przekątna1[w+k-1]=0 i przekątna2[k-
w+n]

procedura ustaw(w, k)
     zajęty_wiersz[w]←1, przekątna1[w+k-1]←1, przekątna2[k-w+n]←1

procedura zdejmij(w, k)
     zajęty_wiersz[w]←0, przekątna1[w+k-1]←0, przekątna2[k-w+n]←0

procedura próbuj (k)
     dla i=1,2,3,…,n wykonaj
          jeżeli możliwe (w, k)
               ustaw(w, k)
               jeżeli k < n
                    próbuj (k+1)
               przeciwnie
                    ilość_sposobów ← ilość_sposobów + 1
               zdejmij(w, k)
```

Główna część programu:

```
wczytaj n
próbuj(1)
```

```
wypisz ilość_sposobów
```