Zajęcia 9

Temat: Funkcje 2

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, rekurencję, pisze własne funkcje rekurencyjne, algorytm Euklidesa, 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 algorytm Euklidesa,

Formy i metody pracy: praca samodzielna, omówienie, wykład

Zadania do wykonania na zajęciach	Treści programowe
1. Struś pędziwiatr	M.2, P.2.16, P.2.17, A.3.1
2. Liczby parzystocyfrowe	M.2, P.2.16, P.2.17, A.3.2
3. Drzewo Sterna-Brocota	M.2, P.2.16, P.2.17, A.3.2

Materiały do zajęć:

https://www.main2.edu.pl/main2/courses/show/6/21/

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

Zadania do wykonania w domu:

Szybkie potęgowanie (przygotowana paczka do SIO2)

ZADANIA I ROZWIĄZANIA

Zadanie 1. Struś pędziwiatr

Limit pamięci: 64MB

Struś Pędziwiatr porusza się z niesamowitą prędkością. Jest w stanie całe miasto przemierzyć w mniej niż jedną sekundę. Na przeszkodzie stoją mu jednak sygnalizatory świetlne. Nie są ze sobą zsynchronizowane, przez co Struś wciąż rusza i wciąż się zatrzymuje. Każde skrzyżowanie w mieście działa w ten sam sposób: przez x_i sekund na i-tym skrzyżowaniu zapalone jest czerwone, a następnie przez jedną sekundę zielone światło. Struś zauważył, że zdarzają się takie momenty, kiedy wszystkie światła w mieście świecą na zielono. Ile sekund w najgorszym przypadku musi czekać na taki moment Struś?

Wejście

W pierwszym wierszu wejścia znajduje się liczba całkowita n ($1 < n < 10$) – liczba skrzyżowań w mieście. W kolejnych n liniach znajduje się po jednej liczbie całkowitej x_i – czas trwania zmiany czerwonych świateł na i-tym skrzyżowaniu ($1 < x_i < 50$). Możesz założyć, że każde ze skrzyżowań działa z inną częstotliwością zmian świateł.

Wyjście

Jedna liczba całkowita – najmniejszy czas, jaki upłynie pomiędzy dwoma momentami, kiedy zielone światła zapalą się jednocześnie na wszystkich skrzyżowaniach.

Przykład

Wejście	Wejście
2 3 5	11

Rozwiązanie

Rozwiązanie zadania wymaga omówienia podstawowych informacji związanych z algorytmem Euklidesa, funkcjami rekurencyjnymi oraz definicjami i równaniami rekurencyjnymi.

Warto zauważyć, że pełna zmiana świateł na i-tym skrzyżowaniu trwa x_i+1 sekund. Skoro światła się zmieniają cyklicznie i zają się takie momenty, kiedy wszystkie światła w mieście świecą na zielono, to znaczy że pełny cykl musi trwać NWW ze wszystkich czasów x_i+1. Przez ostatnią sekundę wszystkie światła będą świeciły się na zielono.

Spróbujmy w naszym zadaniu zdefiniować funkcję NWD (wykorzystywaną do wyznaczenia NWW) rekurencyjnie. Skorzystajmy ze wzoru:

$$nwd(a,b) = \begin{cases} a \; dla \; b = 0 \\ nwd(b, a \bmod b) \; dla \; b \neq 0 \end{cases}$$

Funkcja NWD rekurencyjnie:

```
funkcja nwd (a, b)
     jeżeli b = 0
          zwróć a
```

```
    nwd (b, a mod b)
```

Teraz wystarczy obliczyć NWW dla wszystkich wartości z zadania.

```
wczytaj n
wczytaj x
wynik ← x + 1
dla i=1 do n-1 wykonaj
    wczytaj x
    wynik ← (x+1)·wynik / nwd(x+a,wynik)
zwróć wynik - 1
```

Zadanie 2. Liczby parzystocyfrowe

Limit pamięci: 256MB. Źródło: OIJ (szkopuł.edu.pl)

Dodatnią liczbę całkowitą nazywamy parzystocyfrową, jeśli wszystkie jej cyfry są parzyste. Na przykład: liczby 6, 42, 2020 są parzystocyfrowe, zaś 7, 34, 2019 lub 13 579 nie są. Gdyby wszystkie liczby parzystocyfrowe ustawić w kolejności rosnącej, która liczba byłaby N-ta w tym porządku?

Napisz program, który: wczyta liczbę naturalną N, wyznaczy N-tą liczbę parzystocyfrową i wypisze wynik na standardowe wyjście.

Wejście

W pierwszym (jedynym) wierszu wejścia znajduje się jedna liczba naturalna N. Liczba będzie równa co najmniej 1 i co najwyżej 10^{18}.

Wyjście

W pierwszym (jedynym) wierszu wyjścia powinna się znaleźć jedna liczba naturalna – N-ta liczba parzystocyfrowa.

Przykład

Wejście 12 Wejście 44	Wejście 7921 Wejście 446282

Rozwiązanie

Omówienie rozwiązania: https://oij.edu.pl/oij14/zadania/par/parroz.pdf

Zadanie można w prosty sposób rozwiązać rekurencyjnie:

```
procedura dziesiętny_na_piątkowy (x)
    jeżeli x ≥ 5
        dziesiętny_na_piątkowy (x div 2)
wypisz (x mod 5) · 2
```

Zadanie dla uczniów: Jakie znaczenie ma rekurencyjne wywołanie funkcji przed lub po wypisaniu kolejnej cyfry?

Główna część programu wygląda więc następująco:

```
wczytaj n
procedura dziesiętny_na_piątkowy (n)
```

Zadanie 3. Drzewo Sterna-Brocota

Limit pamięci: 128MB

Drzewo Sterna-Brocota to drzewo binarne zawierające wszystkie dodatnie ułamki nieskracalne. Własności (według Wikipedii):

- W drzewie występują wszystkie dodatnie liczby wymierne zapisane jako ułamki nieskracalne.
- Jeśli liczby a oraz b są względnie pierwsze, to ułamek $\frac{a}{b}$ występuje w drzewie dokładnie raz.

Zaczynamy od $\frac{0}{1}$ - symbolizującego zero i $\frac{1}{0}$ symbolizującego nieskończoność. Następnie na kolejnych piętrach drzewa wpisujemy „pomiędzy" wartości $\frac{a}{b}$ oraz $\frac{c}{d}$ wartość $\frac{a+c}{b+d}$.

Zatem w pierwszym kroku mamy: $\frac{0}{1}\ \frac{1}{1}\ \frac{1}{0}$

W drugim kroku: $\frac{0}{1}\ \frac{1}{2}\ \frac{1}{1}\ \frac{2}{1}\ \frac{1}{0}$

Zaś w trzecim: $\frac{0}{1}\ \frac{1}{3}\ \frac{1}{2}\ \frac{2}{3}\ \frac{1}{1}\ \frac{3}{2}\ \frac{2}{1}\ \frac{3}{1}\ \frac{1}{0}$

Napisz program, który czyta liczbę naturalną n i wypisuje sekwencję ułamków odpowiadającą temu numerowi iteracji.

Wejście

Jedyny wiersz danych zawiera liczbę całkowitą n (0 ≤ n ≤ 20) – numer iteracji.

Wyjście

Program powinien wypisać wiersz tekstu zawierający sekwencję ułamków (zapisanych z użyciem znaku „/") oddzielonych pojedynczymi odstępami.

Przykład

Wejście
4
Wejście
0/1 1/4 1/3 2/5 1/2 3/5 2/3 3/4 1/1 4/3 3/2 5/3 2/1 5/2 3/1 4/1 1/0

Rozwiązanie

Rozwiązanie zadania wymaga omówienia podstawowych informacji związanych z funkcjami rekurencyjnymi, parametrami funkcji oraz zasięgiem zmiennych.

Wyznaczmy rozmiar drzewa w n-tej iteracji. Zauważmy, że przed pierwsza iteracją mamy 2 wyrazy. W każdym kolejnym kroku pomiędzy wszystkie dotychczasowe wyrazy wstawiamy

o jeden mniej wyraz: 0 – 2 wyrazy. 1 – 2+1 wyrazy, 2 – 3+2 wyrazy, 3 – 5+4 wyrazy, 4 – 9+8 wyrazów. Łatwo zauważyć, że dla n>0 mamy 2^n+1 wyrazów.

Zauważmy, że $a^n=a^{n-1}\cdot a$, możemy więc obliczyć potęgę ze wzoru rekurekurencyjnego:

$$potęga(a,n)=\begin{cases}1\ dla\ n=0\\ potęga(a,n-1)\cdot a\ dla\ n\neq 0\end{cases}$$

Funkcja potęga rekurencyjnie:

```
funkcja potęga (a, n)
     jeżeli n = 0
          zwróć 1
     zwróć potęga(a, n-1)·a
```

Jak wyznaczyć wartość „środka” pomiędzy dwoma elementami tablicy?

```
środek ← (lewy+prawy)/2
licznik[środek] ← licznik[lewy]+licznik[prawy]
mianownik[środek] ← mianownik[lewy]+ mianownik[prawy]
```

Teraz wystarczy wyznaczyć rekurencyjnie wszystkie „środki” dla wszystkich zadanych wyrazów pomiędzy wskazanym lewym i prawym elementem. Kolejne „środki” będziemy wyznaczać tak długo, dopóki pomiędzy lewym i prawym elementem pozostanie jeszcze co najmniej jeden wyraz wolny. Dla uproszczenia zakładamy, że tablice licznik i mianownik zostały zdefiniowane jako zmienne globalne.

```
procedura dsb (lewy, prawy)
środek ← (lewy+prawy)/2
licznik[środek] ← licznik[lewy]+licznik[prawy]
mianownik[środek] ← mianownik[lewy]+ mianownik[prawy]
jeżeli prawy-lewy>1
     dsb(lewy, środek)
     dsb(środek, prawy)
```

Główna część programu:

```
wczytaj n
rozmiar ← potęga(2,n)
licznik[0] ← mianownik[rozmiar] ← 0
licznik[rozmiar] ← mianownik[0] ← 1
dsb(0,rozmiar)
dla i=0 do n wykonuj
     wypisz licznik[i], '/', mianownik[i]
```

Dla wnikliwego ucznia: czym różnią się tablice licznik i mianownik? Jak znacząco można oszczędzić pamięć?