ZMP - Lista 2

Funkcje rekurencyjne, wskaźniki i tablice

Marcin Michalski, WMAT PWr

Marzec 2024

Ćwiczenia są do przećwiczenia na laboratoriach/samodzielnie. Jeśli na wykładzie nie doszliśmy do dynamicznych struktur, to załóż, że ograniczamy się do tablic arbitralnych rozmiarów.
Zadania należy rozwiązać zgodnie ze specyfikacją i udostępnić prowadzącym laboratoria.

Deadline: 23.03.2024, 23:59.

Ćwiczenie 1. Napisz funkcję, która zapisze do tablicy pierwsze 20 liczb naturalnych, a następnie wypisze sumę ich kwadratów. Zrób to trawersując tablicę na 2 sposoby - wykorzystując indeksy i wskaźnik.

Ćwiczenie 2. Zaimplementuj sito Eratostenesa.

Ćwiczenie 3. Zaimplementuj funkcję obliczającą NWD dwóch liczb na 2 sposoby - za pomocą pętli i rekurencyjnie.

Ćwiczenie 4. Napisz funkcję, która oblicza NWW podanego (skończonego) ciągu liczb.

Ćwiczenie 5. Napisz funkcję obliczającą symbol Newtona $\binom{n}{k}$.

Ćwiczenie 6. Zaimplementuj funkcję `swap`, która zamieni wartości dwóch zmiennych całkowitych. Zrób to na 2 sposoby - przekazując parametry przez referencję i za pomocą wskaźników.

Ćwiczenie 7. Napisz funkcję, która dla danej tablicy `tab` liczb całkowitych o znanym rozmiarze i liczby całkowitej `p` poprzestawia zawartość `tab` w taki sposób, że najpierw w tablicy będą liczby mniejsze[1] od `p`, potem równe[1] `p`, a potem większe[1] od `p`.
Postaraj się to zrobić "w miejscu", tzn. bez alokowania dodatkowej pamięci na nową tablicę.

Ćwiczenie 8. Napisz funkcję `subsets(int tab[], int size)`, która dla zadanej tablicy liczb całkowitych o znanym wymiarze wypisze wszystkie "podzbiory" tej tablicy. Możemy założyć, że wartości w tablicy się nie dublują. Np. wywołanie `subsets(tab, 3)` dla `int tab[]={1,2,3}` powinno zwrócić coś izomorficznego z `{} {1} {2} {3} {1,2} {1,3} {2,3} {1,2,3}`.

Ćwiczenie 9. Przypomnijmy, że liniowy porządek $(X, \preceq)$ jest dobry, jeśli każdy niepusty podzbiór $A \subseteq X$ ma element najmniejszy (w sensie $\preceq$). Pokaż, że porządek liniowy jest dobry wtedy, i tylko wtedy, gdy nie ma w nim nieskończonych ciągów ściśle malejących.

[1] O ile takie są.

Ćwiczenie 10. Funkcja Ackermana $A : \mathbb{N}^2 \to \mathbb{N}$ jest zadana następująco

$$A(m,n) = \begin{cases} n+1, & \text{gdy } m = 0, \\ A(m-1,1), & \text{gdy } m > 0, n = 0, \\ A(m-1, A(m,n-1)), & \text{otherwise.} \end{cases}$$

Zaimplementuj funkcję Ackermana i oblicz/wyznacz jej wartości dla początkowych m. Następnie udowodnij, że jest dobrze zdefiniowana.

Rozwiązania zadań umieść w `lista_2/z_i`, gdzie `i` to numer zadania, w swoim repozytorium. Pliki nazwij `nr_indeksu_zic.cpp`, gdzie `c`, to odpowiedni podpunkt, o ile zadanie taki ma (jeśli nie ma, to `c` należy pominąć). Przykład: `123456_z1a.cpp`.

Zadanie 1 (4p.)**.** Napisz programy, które będą obliczać n-ty wyraz ciągu Fibonacciego[2]

(a) za pomocą pętli;

(b) rekurencyjnie.

Po uruchomieniu każdy z nich powinien oczekiwać liczby naturalnej ze standardowego wejścia i na wejściu oddać wynik.
Oba programy powinny działać rozsądnie szybko, tzn. powinniśmy się doczekać obliczenia np. 40-stego wyrazu tego ciągu.

Zadanie 2 (4p.)**.** (a) Napisz program, który przyjmie ze standardowego wejścia 1001 liczb całkowitych, sprawdzi, czy ostatnia z nich jest wśród tych 1000 poprzednich, i na wyjściu wypisze TAK, jeśli tak jest, i NIE w przeciwnym przypadku.

(b) To samo, tylko przy założeniu, że dane (oprócz ostatniej liczby) przychodzą posortowane rosnąco, a program działa *znacznie* szybciej. Ile porównań wystarczy w tym przypadku?

Zadanie 3 (3p.)**.** Napisz program, który po wywołaniu z dwoma ciągami znaków jako argumentami[3] stwierdzi, czy te ciągi są swoimi anagramami, i zwróci w standardowym wyjściu TAK, jeśli tak jest, i NIE w przeciwnym razie.
Postaraj się nie korzystać z gotowych funkcji na obiektach `string` ani z dynamicznych struktur danych, jeśli nie mówiliśmy o nich jeszcze na wykładzie.

[2]Przyjmijmy, że ciąg Fibonacciego to $0, 1, 1, 2, 3, 5, 8, \ldots$ i jego numeracja zaczyna się, oczywiście, od 0.

[3]Powinny być obsłużone przez drugi parametr funkcji `main`.