Politechnika Wrocławska

Dorota Aniszewska, Wojciech Myszka

PROGRAMOWANIE W JĘZYKU C
Lista nr 4. Tablice. Algorytmy: sortowanie, wyszukiwanie binarne.

Tablica jednowymiarowa może mieć stały rozmiar; najlepiej wówczas jest określić go używając *#define*. Rozmiar tablicy może być również dynamiczny, czyli być ustalany w trakcie działania programu - może być on wylosowany lub nadany przez użytkownika. Należy wówczas zarezerwować odpowiednią ilość pamięci używając funkcji *malloc()*, a na końcu programu ją zwolnić używając funkcji *free()*.

Program tablice.c zawiera przykładową deklarację i funkcje dla tablicy o stałym rozmiarze. Program tablicedynamicznie.c zawiera przykładową deklarację i funkcje dla tablicy o rozmiarze ustalanym w trakcie działania programu.

1. Napisz program, który dla tablicy liczb całkowitych o stałym rozmiarze:
   a) znajduje wartość największą i najmniejszą,
   b) oblicza średnią arytmetyczną,
   c) oblicza standardowe odchylenie według wzoru: $\sigma = \sqrt{\frac{\sum_{i=1}^{n}(a_i - s)^2}{n}}$, gdzie a_i to i-ty element tablicy, n - liczba elementów tablicy, s - średnia arytmetyczna.

2. Napisz funkcję, która nadaje wartości losowe wszystkim elementom tablicy typu *int*:
   void LosujTablice(int, int, int, int)* - pierwszy parametr to adres tablicy, drugi liczba elementów tablicy, trzeci to najmniejsza możliwa wartość, jaką może mieć element tablicy, czwarty parametr to największa możliwa wartość, jaką może mieć element tablicy. Wywołaj ją w funkcji *main*.

3. Napisz funkcję wyświetlającą wszystkie elementy tablicy na ekranie:
   *void WyswietlTablice(int *, int, int)* - pierwszy parametr to adres tablicy, drugi to liczba elementów tablicy, trzeci parametr to liczba elementów tablicy w jednym wierszu, Wywołaj ją w funkcji *main*.

4. Napisz funkcję obliczającą wartość średniej (nie wyświetlająca jej na ekranie!). Wywołaj ją w funkcji *main* i wyświetl wartość średniej.

5. Napisz funkcję, która „odwróci" tablicę, czyli pierwszy element zostanie ostatnim, a ostatni pierwszym.

6. Napisz funkcję, która zamieni miejscami wartości dwóch wskazanych elementów tablicy.

7. Napisz program, który podaje, na którym miejscu w tablicy liczb całkowitych znajduje się element o najmniejszej wartości. Jeśli jest więcej niż jeden elementów tablicy o tej wartości, program ma podać liczbę tych elementów i wszystkie ich pozycje w tablicy.

8. Napisz funkcję, który "przesuwa" elementy tablicy o jedno miejsce w prawo albo w lewo (wybiera użytkownik) wolne miejsca uzupełniając zerami, np. tablica (5,4,3,2,1) przesunięta w prawo to (0,5,4,3,2).

9. Napisz funkcję rekurencyjną, która oblicza sumę elementów tablicy.

10. Zaprogramuj następujący algorytm: Utwórz tablicę jednowymiarową T1 o podanym przez użytkownika rozmiarze i nadaj jej losowe wartości z przedziału do 100.
krok 1. Oblicz średnią wartość elementów tablicy T1 i utwórz nową tablicę T2, która będzie zawierała tylko większe od średniej elementy z tablicy T1,
krok 2. Przepisz wartości z tablicy T2 do T1 oraz usuń T2,
krok 3. Jeśli liczba elementów tablicy T1 $>$ 1, przejdź do kroku 1, w przeciwnym razie zakończ program.

11. Napisz funkcję, który zrealizuje dodawanie wylosowanych liczb binarnych. Liczby binarne składają się z wartości 1 i 0 zapisanych do tablicy liczb całkowitych typu unsigned char o stałym rozmiarze 8 lub parzystej wielokrotności liczby 8. Wynik ma być zapisywany w tablicy o tym samym rozmiarze. Program ma obsługiwać overflow. Na ekranie dwie liczby oraz wynik wyświetlane są użytkownikowi jako ciąg 1 i 0. Dodatkowo można zaprogramować konwersję do systemu dziesiętnego, tylko i wyłącznie w celu prezentacji liczb na ekranie.

12. Napisz funkcję, który zrealizuje mnożenie wylosowanych liczb binarnych. Liczby binarne składają się z wartości 1 i 0 zapisanych do tablicy liczb całkowitych typu unsigned char o stałym rozmiarze 8 lub parzystej wielokrotności liczby 8. Wynik zapisywany jest w tablicy o dwa razy większym rozmiarze. Na ekranie dwie liczby oraz wynik wyświetlane są użytkownikowi w systemie binarnym. Dodatkowo można zaprogramować konwersję do systemu dziesiętnego, tylko i wyłącznie w celu prezentacji liczb na ekranie.

13. **Sortowanie bąbelkowe**. Napisz funkcję, który posortuje tablicę liczb losowych rosnąco według podanego algorytmu: sortowanie bąbelkowe polega na porównywaniu dwóch kolejnych elementów i zamianie ich kolejności, jeżeli zaburza ona porządek, w jakim się sortuje tablicę. Sortowanie kończy się, gdy podczas kolejnego przejścia (porównania każdego elementu tablicy z następnym) nie dokonano żadnej zmiany.
Przykład działania dla tablicy (4,2,1,7,5):
1 przejście tablicy:
(**4**,2,1,7,5)→zamiana miejscami 2 i 4 → (2,**4**,1,7,5) → zamiana miejscami 4 i 1 → (2,1,**4**,7,5) → (2,1,4,**7**,5) → zamiana miejscami 7 i 5 → (2,1,4,5,7)
2 przejście tablicy:
(**2**,1,4,5,7) → zamiana miejscami 2 i 1 → (1,**2**,4,5,7) → (1,2,**4**,5,7) → (1,2,4,**5**,7)
3 przejście tablicy:
(**1**,2,4,5,7) → (1,**2**,4,5,7) → (1,2,**4**,5,7) → (1,2,4,**5**,7)
Koniec działania algorytmu, ponieważ tablica jest posortowana.

14. **Sortowanie przez wybieranie**. Napisz funkcję, który posortuje tablicę liczb losowych rosnąco według podanego algorytmu: jeśli sortujemy tablicę n-elementową rosnąco, to na i-tym miejscu ma znaleźć się najmniejszy element znaleziony wśród elementów od i do n tablicy, co oznacza, że porównujemy i-ty element z minimum znalezionym wśród elementów od i+1 do n i zamieniamy miejscami te wartości, jeśli i-ty element jest większy niż znalezione minimum.
Przykład działania dla tablicy (4,5,3,7,2) (n=5, algorytm będzie miał 4 kroki);
1 krok:
(**4**,5,3,7,2) → zamieniamy wartość 4 z wartością 2 → (2,5,3,7,4)
2 krok:
(2,**5**,3,7,4) → zamieniamy wartość 5 z wartością 3 → (2,3,5,7,4)
3 krok:
(2,3,**5**,7,4) → zamieniamy wartość 5 z wartością 4 → (2,3,4,7,5)
4 krok:
(2,3,4,**7**,5) → zamieniamy wartość 7 z wartością 5 → (2,3,4,5,7).

15. **Sortowanie przez wstawianie**. Napisz funkcję, który posortuje tablicę liczb losowych rosnąco według podanego algorytmu: wybieramy po kolei elementy (zaczynamy od drugiego), porównujemy go z kolejnymi elementami zbioru już posortowanego (w pierwszym kroku zbiór posortowany zawiera tylko jeden element), póki nie napotkamy elementu równego lub elementu większego (jeśli chcemy otrzymać ciąg rosnący) lub nie znajdziemy się na początku/końcu zbioru uporządkowanego. Wyciągnięty element wstawiamy w miejsce gdzie skończyliśmy porównywanie.
Przykład działania dla tablicy (4,10,3,7,1) (5 elementów, więc 4 kroki są potrzebne):
1 krok:
(4,**10**,3,7,1) → wartość 10 jest na właściwym miejscu, bo jest większa niż 4
2 krok:
(4,10,**3**,7,1) → wartość 3 trzeba wstawić w odpowiednie miejsce, porównujemy tą wartość z elementami ją poprzedzającymi, ma być na pierwszym miejscu, więc elementy drugi i trzeci przesuwamy w prawo i na miejsce pierwsze wstawiamy wartość 3 → (3,4,10,7,1)
3 krok:
(3,4,10,**7**,1) → wartość 7 trzeba wstawić w odpowiednie miejsce, porównujemy tą wartość z elementami ją poprzedzającymi, ma być na trzecim miejscu, więc element czwarty przesuwamy w prawo i na miejsce trzecie wstawiamy wartość 7 → (3,4,7,10,1)
4 krok:
(3,4,7,10,1) → wartość 1 trzeba wstawić w odpowiednie miejsce, porównujemy tą wartość z elementami ją poprzedzającymi, ma być na pierwszym miejscu, więc pozostałe elementy przesuwamy w prawo i na miejsce pierwsze wstawiamy wartość 1 → (1,3,4,7,10).

16. **Sortowanie szybkie (quicksort)**: algorytm jest rekurencyjny. W każdym kroku w tablicy wybiera się element rozdzielający, po czym do lewej przenoszone są wszystkie elementy nie większe od wartości elementu rozdzielającego (również element rozdzielający, na końcu lewej części), w prawej części tablicy zostają wszystkie elementy większe. Powtórzyć należy te czynności dla dwóch części tablicy. Rekurencja kończy się, gdy kolejny fragment uzyskany z podziału zawiera pojedynczy element, ponieważ tablica jednoelementowa nie wymaga sortowania.
Uwaga: najlepszym elementem rozdzielającym jest mediana, ale prościej wybrać "środkowy z trzech" (wybieramy losowo trzy elementy i rozdzielającym będzie element, którego wartość leży pomiędzy wartościami pozostałych dwu), można też wartość elementu rozdzielającego wylosować.

17. Napisz funkcję, która realizuje wyszukiwanie binarne zadanej wartości w posortowanej tablicy liczb całkowitych i jeśli się ona w tablicy znajduje, zwraca numer elementu o tej wartości. Wyszukiwanie binarne polega na dzieleniu tablicy na coraz mniejsze przedziały do momentu, gdy szukany element zostanie znaleziony, bądź przedział osiągnie długość zero, co oznacza brak szukanego elementu. W każdym kroku wyznaczany jest środek przedziału i sprawdzane jest, czy element zapisany na środku przedziału jest poszukiwanym elementem (wówczas algorytm kończy działanie), czy jest on niego mniejszy (albo większy): wówczas przedział jest odpowiednio zawężany.

18. Napisz funkcję, która oblicza pochodną wielomianu. Stopień wielomianu oraz współczynniki mogą być wylosowane. Wynik działania, czyli pochodną, funkcja powinna zapisać do tablicy. Wywołaj tą funkcję w funkcji *main* i wyświetl wielomian i jego pochodną.

19. Napisz program (funkcję), która oblicza iloczyn dwóch wielomianów. Stopnie obu wielomianów oraz współczynniki mogą być wylosowane. Iloczyn powinien być zapisany do tablicy o odpowiedniej wielkości. Wywołaj tą funkcję w funkcji *main* i wyświetl wielomiany i ich iloczyn.