Lista zadań nr 5

Tablice jednowymiarowe. Funkcje operujące na tablicach jednowymiarowych. Arytmetyka wskaźnikowa. Dynamiczna alokacja pamięci dla tablic jednowymiarowych

Zadania podstawowe:

Zadanie 1 Napisz program, w którym deklarowana jest 10-elementowa tablica typu `int`. Tablica powinna być inicjowana dowolnymi liczbami całkowitymi. Program powinien wyświetlać elementy tablicy używając pętli. Użyj dyrektywy `#define` do określenia rozmiaru tablicy.

Zadanie 2 Napisz program, który wczytuje osiem liczb zmiennoprzecinkowych (typu `double`) do tablicy, a następnie wyświetla je w odwrotnej kolejności.

Zadanie 3 Napisz program, w którym deklarowana jest 10-elementowa tablica liczb typu `double`. Zaprojektuj i wykorzystaj funkcję `complete_array()`, której zadaniem będzie pobranie od użytkownika kolejnych liczb i zapisanie ich w kolejnych komórkach tablicy. Zaprojektuj funkcję `print_array()`, która zajmie się wyświetleniem tablicy.

Zadanie 4 Zaprojektuj i napisz funkcję, która policzy ile jest w danej tablicy liczb podzielnych przez k (k - liczba podana przez użytkownika). Przetestuj funkcję w prostym programie.

Zadanie 5 Zaprojektuj i napisz funkcję, która znajdzie największy i najmniejszy element w tablicy i zamieni je miejscami. Przetestuj funkcję w prostym programie.

Zadanie 6 Zaprojektuj i napisz funkcję, która policzy ile jest w tablicy liczb dodatnich, ujemnych i równych zero. Przetestuj funkcję w prostym programie.

Zadanie 7 Zaprojektuj i napisz funkcję, która przesunie liczby w tablicy o jedno miejsce tzn. pierwszy element przestawi na ostatnią pozycję, drugi na pierwszą, trzeci na drugą pozycję itd.. Przetestuj funkcję w prostym programie.

Zadanie 8 Zaprojektuj i napisz funkcję, która odwróci elementy (pierwszy zamieni z ostatnim, drugi z przedostatnim itd.) tablicy przekazanej jako argument funkcji. Przetestuj funkcję w prostym programie.

Zadanie 9 Zaprojektuj i napisz funkcję, która wymnoży wszystkie elementy tablicy przez daną liczbę. Tablica (wskaźnik do pierwszego elementu) i liczba powinny być przekazane jako parametry funkcji. Przetestuj funkcję w prostym programie.

Zadanie 10 Zaprojektuj i napisz funkcję szukającą w danej n-elementowej tablicy typu `int` pary sąsiednich elementów, których suma jest największa. Funkcja jest typu `void`, a indeks pierwszego z elementów spełniających warunek ma być dostępny w miejscu wywołania funkcji.

Zadanie 11 Zaprojektuj i zdefiniuj funkcję, która posortuje dane w tablicy (metodą bąbelkową lub przez wybór elementu minimalnego). Przetestuj funkcję na tablicy o stu elementach generowanych losowo (losowe liczby od -1000 do 1000) - za wypełnianie tablicy odpowiednimi wartościami powinna odpowiadać osobna funkcja.

Zadanie 12 Zaprojektuj i napisz funkcję:

- wyznaczającą sumę dwóch n-elementowych wektorów (tablic n-elementowych typu double);
- wyznaczającą iloczyn skalarny dwóch n-elementowych wektorów (suma iloczynów odpowiadających sobie współrzędnych).

Przetestuj funkcje w prostym programie.

Zadanie 13 Napisz program, w którym deklarowana i inicjowana jest 3-elementowa tablica liczb typu `int`. Zadeklaruj trzy zmienne wskaźnikowe typu `int` i „ustaw" je, tak aby wskazywały na adresy kolejnych komórek tablicy. Wykorzystując zmienne wskaźnikowe zmodyfikuj elementy tablicy trzykrotnie zwiększając wartości, które się tam znajdują. Wyświetl zmodyfikowaną tablicę.

Zadanie 14 Pierwsze dwa argumenty funkcji `licz()` wskazują dwa elementy tej samej tablicy typu `double`. Uzupełnić definicję funkcji w taki sposób, by zwracała informację o liczbie elementów posiadających taką samą wartość jak trzeci argument i zawartych między elementami wskazywanymi przez pierwsze dwa argumenty włączając w to element wskazywany przez pierwszy argument oraz wyłączając element wskazywany przez drugi argument. Przetestuj funkcję w poniższym programie.

```
#include <stdio.h>
int licz(double *p1, double *p2, double x);
int main(void){
 double t[]={8.0, 2.0, 1.0, 6.0, 2.0, 7.0, 5.0, 2.0, 9.0};
 int n= licz(t+1, t+8, 2.0);
 printf("%d\n", n);
 return 0;
}
```

Zadanie 15 Wskaźnik `wsk` dzieli N elementową tablicę liczb typu `int` na dwie części. Napisz definicję funkcji, której parametrami będą wskaźnik na początek tablicy, wskaźnik `wsk` i rozmiar tablicy. Funkcja powinna obliczyć dwie sumy elementów tablicy: umieszczonych przed elementem wskazanym przez `wsk` (łącznie z nim) oraz sumę elementów położonych za nim. Jeśli suma pierwszej części tablicy jest większa, to funkcja powinna zwrócić `0`, w przeciwnym wypadku `1`.
Prototyp funkcji: `int fsuma(int *t, int *wsk, int roz);`.

Zadanie 16 Napisz funkcję, która jako argumenty przyjmuje tablicę o elementach typu `double` oraz rozmiar tej tablicy. Funkcja powinna zwracać wskaźnik do elementu maksymalnego tej tablicy. W definicji funkcji nie możesz używać żadnych zmiennych typu `double` oraz korzystać z indeksowania. Przetestuj funkcję w prostym programie.

Zadanie 17 Napisz funkcję, która jako argumenty przyjmuje tablicę o elementach typu `int`, rozmiar tej tablicy oraz pewną liczbę całkowitą `k`. Funkcja powinna zwracać informację o liczbie elementów w tablicy, które są większej od `k`. W definicji funkcji możesz użyć tylko jednej zmiennej całkowitej i nie możesz używać indeksowania. Przetestuj funkcję w prostym programie.

Zadanie 18 Napisz funkcję `create_array()` przydzielającą pamięć dla tablicy o k elementach typu `double`. Liczba elementów tablicy jest parametrem tej funkcji, a funkcja zwraca wskaźnik do odpowiedniego bloku pamięci zaalokowanego przy pomocy funkcji `malloc()`.

W programie, korzystając z funkcji `create_array()`, utwórz dwie dynamiczne tablice typu `double`: `t1` o rozmiarze n oraz `t2` o rozmiarze m (n i m pobierz w funkcji `scanf()`). W funkcji `complete_array()` wypełnij tablice `t1` i `t2` losowymi liczbami, a następnie przekopiuj ich zawartość do tablicy o identyfikatorze `t3` i rozmiarze $n + m$, również utworzonej dynamicznie.

Wydrukuj tablice `t1`, `t2` i `t3`, korzystając z funkcji `print_array()`. Pamiętaj o zwolnieniu przydzielonej pamięci z wykorzystaniem funkcji `free()`.

Zadanie 19 Napisz funkcję `reverse()`, która jako argumenty przyjmuje dwa wskaźniki. Pierwszy powinien wskazywać na początkowy element pewnej tablicy liczb całkowitych (`int`), a drugi na ostatni element tej tablicy. Funkcja powinna zamieniać kolejno pierwszy element tablicy z ostatnim, drugi z przedostatnim itd., aż do elementów środkowych. Funkcja nie może korzystać z indeksowania tablicy, powinna zmieniać między sobą wartości wskazywane przez dwa wskaźniki.

Przetestuj funkcję `reverse()` w programie na 5 przykładowych tablicach tworzonych dynamicznie (skorzystaj z pętli). Rozmiary kolejnych tablic powinny być podawane przez użytkownika.

W celu utworzenia tablicy program powinien korzystać z funkcji `create_array()` zwracającej wskaźnik do odpowiedniego bloku pamięci zaalokowanego przy pomocy funkcji `malloc()`. Funkcja `create_array()` powinna również wypełniać utworzoną tablicę losowymi liczbami z zakresu od -50 do 50.

W celu wyświetlenia tablicy program powinien korzystać z funkcji `print_array()`, która powinna drukować zawartość tablicy po 20 elementów w wierszu. Pamiętaj o zwalnianiu przydzielonej pamięci z wykorzystaniem funkcji `free()`.

Zadanie 20 Napisz funkcję `bubble_sort()`, która jako argumenty przyjmuje dwa wskaźniki. Pierwszy powinien wskazywać na początkowy element pewnej tablicy liczb całkowitych (`int`), a drugi na ostatni element tej tablicy. Funkcja powinna sortować tablicę niemalejąco, wykorzystując algorytm sortowania bąbelkowego. Funkcja sortująca nie może korzystać z indeksowania. Wykorzystaj funkcję pomocniczą `swap()`, która dokonywać będzie zamiany wartości sąsiednich elementów tablicy. Jako jej parametrów użyj również dwóch wskaźników.

Przetestuj funkcję `bubble_sort()` w programie na pięciu przykładowych tablicach tworzonych dynamicznie (skorzystaj z pętli). Rozmiary kolejnych tablic powinny być podawane przez użytkownika.

W celu utworzenia tablicy program powinien korzystać z funkcji `create_array()` zwracającej wskaźnik do odpowiedniego bloku pamięci zalokowanego przy pomocy funkcji `malloc()`. Funkcja `create_array()` powinna również wypełniać utworzoną tablicę losowymi liczbami z zakresu od -100 do 100.

W celu wyświetlenia tablicy program powinien korzystać z funkcji `print_array()`, która powinna wyświetlać zawartość tablicy po 20 elementów w wierszu. Pamiętaj o zwalnianiu przydzielonej pamięci z wykorzystaniem funkcji `free()`.

Zadania dodatkowe:

Zadanie 1 Zaprojektuj i napisz funkcję, która wyzeruje wszystkie elementy większe od danej liczby. Tablica (wskaźnik do pierwszego elementu) i liczba powinny być przekazane jako parametry funkcji. Przetestuj funkcję w prostym programie.

Zadanie 2 Napisz program, w którym deklarowana jest 100-elementowa tablica typu `long int`. Zdefiniuj i wykorzystaj funkcję `complete_array()` do wypełnienia tablicy kolejnymi kwadratami liczb naturalnych (zaczynając od 1). Zaprojektuj, a następnie zdefiniuj funkcję `print_array()`, która zajmie się wyświetleniem tablicy – funkcja powinna wyświetlać po 10 liczb oddzielonych spacją w 10 wierszach a każda liczba powinna być wyświetlona na pięciu polach.

Zadanie 3 Napisz funkcję szukającą w danej n-elementowej tablicy typu `double` pierwszych dwóch jednakowych elementów. Funkcja zwraca wartość 0, gdy brak jest dwóch jednakowych elementów i 1 w przeciwnym przypadku. Indeksy znalezionych elementów mają być dostępne w miejscu wywołania funkcji. Wykorzystać tę funkcję w programie.

Zadanie 4 Zaprojektuj i napisz funkcję, która posortuje elementy tablicy stosując sortowanie Shella. Przetestuj funkcję w prostym programie.

Zadanie 5 Mamy ciąg $a_0, a_1, \ldots, a_{n-1}$ liczb całkowitych takich, że $a_i \in [-10000, 10000]$ dla $i = 0, 1, 2, \ldots, n-1$. Zakładamy, że $2 \leqslant n \leqslant 100000$. Znajdź takie $i \in 0, 1, 2, \ldots, n-1$ aby różnica miedzy sumą liczb: $\sum_{j=0}^{i} a_j$, a sumą $\sum_{j=i+1}^{n-1} a_j$ była jak najmniejsza. Interesuje nas wartość bezwzględna tej różnicy.

Zadanie 6 Niech m będzie liczbą całkowitą, taką że $1 \leqslant m \leqslant 1000000$. Rozważmy dwie tablice A i B zawierające po $n > 1$ liczb całkowitych odpowiednio $a_0, a_1, \ldots, a_{n-1}$ i $b_0, b_1, \ldots, b_{n-1}$ ($\leqslant a_i, b_i \leqslant m$ dla $i = 0, 1, 2, \ldots, n-1$). Napisz program, który sprawdza czy istnieje operacja zamiany, która może być wykonana na tablicach A i B w taki sposób, aby po dokonaniu zamiany suma elementów tablicy A była równa sumie elementów tablicy B. Mówiąc o operacji zamiany mamy na myśli wybranie jednego elementu z tablicy A oraz jednego z tablicy B i wymienienie ich wartości.

Zadanie 7 Rozważmy ciąg składający się z n ($1 \leqslant n \leqslant 1000000$) liczb całkowitych $a_0, a_1, a_2, \ldots, a_{n-1}$ ($1 \leqslant a_i \leqslant 10^9$ dla $i = 0, 1, 2, \ldots, n-1$). Napisz program, który sprawdzi, czy podany ciąg jest permutacją liczb od 1 do n.

Zadanie 8 Losujemy kolejno m ($1 \leqslant m \leqslant 100000$) liczb z zakresu od 1 do n ($n \leqslant m$). Napisz program, który sprawdzi ile liczb wystarczy wylosować aby skompletować

wszystkie liczby od 1 do n i zwróci tą liczbę - w przypadku niepowodzenie program powinien wyświetlić odpowiednią informację (np. zwrócić liczbę -1).

Zadanie 9 Dysponujemy pewną zabawką posiadającą $n+1$ przycisków. Nad każdym z nich poza ostatnim znajduje się licznik, początkowo wyzerowany. Naciskając dowolny z n przycisków, wartość licznika, który mu odpowiada zwiększa się o 1. Ostatni przycisk o numerze $n+1$ jest inny, jego naciśnięcie powoduje, że wszytki liczniki otrzymują wartość maksymalną spośród dotychczas wskazywanych. Dysponujemy listą przycisków, które były kolejno wciskane. Napisz program, który zwraca stan liczników po wykonaniu serii wciśnięć przycisków.