Lista zadań nr 1

Wskaźniki, dynamiczna alokacja pamięci.

Zadania podstawowe:

Zadanie 1 Zaprojektuj i napisz funkcję, która jako argumenty przyjmuje tablice o elementach typu `int`, rozmiar tej tablicy oraz pewną liczbę całkowitą `k`. Funkcja powinna zwracać liczbę elementów w tablicy, które są większej od `k`. W definicji funkcji może użyć tylko jednej zmiennej całkowitej i nie możesz używać indeksowania. Przetestuj funkcję w prostym programie.

Zadanie 2 Zaprojektuj i napisz funkcję, która jako argumenty przyjmuje tablice 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 3 Zaprojektuj i napisz funkcję, która jako argumenty przyjmuje dwa wskaźniki. Pierwszy powinien wskazywać na pierwszy element pewnej tablicy liczb całkowitych (`int`), a drugi na ostatni element tej tablicy. Funkcja powinna sortować tablicę wykorzystując algorytm sortowania bąbelkowego. Funkcja sortującą nie może korzystać z indeksowania. Zaprojektuj i wykorzystaj funkcję pomocniczą, która zamienia między sobą wartości wskazywane przez dwa wskaźniki będące argumentami wywołania tej funkcji. Przetestuj funkcję sortującą w programie na pięciu przykładowych tablicach tworzonych dynamicznie. 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 10 elementów w wierszu. Pamiętaj o zwalnianiu przydzielonej pamięci korzystając z funkcji `free()`.

Zadanie 4 Napisz funkcję, która dostaje jako argumenty dwuwymiarową prostokątną tablicę tablic `tab1` o wymiarach $n \times m$ i elementach typu `int` oraz jej wymiary, i zwraca jako wartość wskaźnik do nowo utworzonej dwuwymiarowej tablicy tablic `tab2` o wymiarach $m \times n$ zawierającej transponowaną macierz przechowywaną w tablicy `tab1` (czyli dla dowolnych `k` i `j` zachodzi `tab1[k][j] = tab2[j][k]`). Przetestuj funkcję w prostym programie.

Zadanie 5 Napisz funkcję, która otrzymuje w argumentach dwie kwadratowe tablice dwuwymiarowe elementów typu `int` oraz ich wspólny wymiar, i zwraca jako wartość wynik dodawania macierzy przechowywanych w przekazanych argumentach. Wynik powinien zostać zwrócony w nowo utworzonej tablicy dwuwymiarowej.

Zadanie 6 Napisz funkcję, która otrzymuje w argumentach dwie kwadratowe tablice dwuwymiarowe elementów typu `int` oraz ich wspólny wymiar, i zwraca jako wartość wynik mnożenia macierzy przechowywanych w przekazanych argumentach. Wynik powinien zostać zwrócony w nowo utworzonej tablicy dwuwymiarowej. Przetestuj funkcję w prostym programie.

Zadanie 7 Napisz program w którym osobna funkcja alokuje pamięć dla dwuwymiarowej tablicy typu `int` o rozmiarze $n \times m$. Funkcja powinna działać dla dowolnych $(m, n > 0)$ i wypełniać tablicę liczbami pseudolosowymi z zakresu od 2 do 100. Wartości n oraz m są podane przez użytkownika, ale takie że $n \cdot m = 1000$. Napisz kolejną funkcję która, jako argumenty przyjmuje tablicę (`tab`) i jej wymiary alokowaną przez pierwszą funkcję oraz liczbę całkowitą $2 \leqslant k \leqslant 100$ i wskaźnik do `int`. Funkcja powinna sprawdzać ile jest liczb w tablicy `tab` podzielnych przez k, tworzyć dynamicznie tablicę jednowymiarową o takim rozmiarze i zapisywać tam te liczby. Funkcja powinna zwracać wskaźnik do tak zaalokowanej tablicy oraz zapisywać jej rozmiar w miejscu wskazywanym przez przekazany jej wskaźnik. Jeżeli w tablicy `tab` nie ma liczb podzielnych przez k funkcja powinna zwracać `Null`. Napisz osobną funkcję do wyświetlania liczb zapisanych w takiej tablicy (tj. tych podzielnych przez k) - pamiętaj, że jej rozmiar będzie zapisany w zmiennej do której wskazuje argument wskaźnikowy poprzedniej funkcji. Przetestuj działanie wszystkich funkcji w programie.

Zadanie 8 Napisz program, w którym definiujesz funkcję, która dynamicznie alokuje pamięć dla tablicy 200 liczb całkowitych. Funkcja powinna wypełniać tablicę losowymi liczbami z zakresu 0 do 5000 i zwracać wskaźnik do tak stworzonej tablicy (`tab`). Następnie, zdefiniuj funkcję która jako argument będzie przyjmowała tak stworzoną tablicę i jej rozmiar oraz zwykłą tablicę typu `int` o rozmiarze 10. Wspomina funkcja powinna stworzyć tablicę postrzępioną o 10 wierszach - wierszu pierwszym powinny być zapisane wszystkie liczby z tablicy `tab`, których ostatnią cyfrą jest `0`, w wierszu drugim - te których ostatnią cyfrą jest 1 itd. Liczność poszczególnych wierszy (tablic alokowanych dynamicznie) powinna być zapisywana w kolejnych elementach tablicy, przekazanej funkcji - tej o rozmiarze 10. Napisz osobną funkcję, która wyświetli taką tablicę postrzępiona - pamiętaj, że rozmiar każdego wiersza jest jest zapisany w osobnej tablicy! Przetestuj działanie wszystkich funkcji w programie.

Zadanie 9 Napisz program w którym osobna funkcja alokuje pamięć dla dwuwymiarowej tablicy typu `double` o rozmiarze $n \times m$ $(5 \leqslant n, m \leqslant 20)$. Funkcja powinna wypełniać tablice liczbami losowymi np. z zakresu od -100.0 do 100. Napisz funkcję która, stosując np. sortowanie bąbelkowe albo sortowanie przez wybieranie posortuje tę tablicę względem kolumn - w posortowanej tablicy jako pierwsza (z lewej strony) powinna być kolumna o najmniejszej sumie elementów, a jako ostatnia (z prawej strony) ta o największej sumie elementów. W celu zamiany danych kolumn w tablicy skorzystaj z osobnej funkcji. Również do sumowania elementów we wskazanej kolumnie wykorzystaj osobną funkcję.

Zadania dodatkowe:

Zadanie 1 Napisz program, który symuluje grę w kości. Aby wygrać, w pierwszym rzucie gracz musi wyrzucić na dwóch kościach sumę oczek równą 7 albo 11. Jeżeli wyrzuci sumę 2, 3 lub 12, przegrywa. Każdy inny wynik to tzw. "punkt" zezwalający na kontynuację gry. W kolejnych rzutach gracz odnosi zwycięstwo, jeżeli ponownie wyrzuci "punkt", a przegrywa przez wyrzucenie sumy 7. Na końcu każdej gry program ma zapytać użytkownika, czy gra jeszcze raz. Gdy użytkownik zdecyduje o zakończeniu rozgrywki, program ma wypisać liczbę przegranych i wygranych gier i zakończyć działanie. Program powinien wykorzystywać następujące funkcje: funkcja `throw()` – zwracająca wyniki rzut dwoma kościami, funkcja `single_game())` – powinna obsługiwać pojedynczą grę poprzez wywołanie funkcji `throw()` dla określenia wyników kolejnych rzutów, a także wyświetlać przebieg gry i zwracać `true` gdy gra zakończy się wygraną gracza lub `false` gdy gracz przegra, funkcja `get_answer()` powinna pobierać prawidłową odpowiedź (1 (tak) lub 0 (nie)) i ją zwracać, funkcja `game()` powinna realizować całą rozgrywkę wywołując odpowiednio funkcje `single_game()` i `get_answer()` oraz zliczać liczbę wygranych i przegranych gracza.