Lista zadań nr 1

Arytmetyka wskaźnikowa, dynamiczna alokacja pamięci, pliki tekstowe (`fscanf()` i `fprintf()`).

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 funkij 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 program, który tworzy plik tekstowy i zapisuje do niego ciąg tekstowy: "Programowanie Proceduralne 2021". Pamiętaj o zamknięciu pliku.

Zadanie 5 Napisz program, który tworzy plik tekstowy *numbers.txt* i zapisuje do niego 500 losowych liczb z przedziału od 1 do 1000. Liczby powinny być zapisywane w osobnych wierszach. Pamiętaj o zamknięciu pliku.

Zadanie 6 Napisz program, który odczytuje kolejne liczby z pliku *numbers.txt* utworzonego przez program z zadania 5, wyświetla je i zapisuje do innego pliku *numbers2.txt* tylko liczby, które są podzielne przez 3. Pamiętaj o zamknięciu obu plików.

Zadanie 7 Napisz program, który wczyta liczby całkowite z pliku *numbers2.txt* (stworzonego w zadaniu 6) i umieści je w tablicy alokowanej dynamicznie. Program powinien korzystać z następujących funkcji:

- funkcja `count_numbers()` - liczy i zwraca liczbę liczb zapisanych w pliku przekazanym jej jako argument i zamyka ten plik;
- funkcja `create_array()` - tworzy dynamicznie tablicę liczb typu `int` o rozmiarze przekazanym przez argument wywołania i zwraca wskaźnik do zaalokowanego bloku pamięci;
- funkcja `complete_array()` - wypełnia tablicę przekazaną jej jako argument liczbami z pliku, który jest jej drugim argumentem wywołania, trzecim argumentem wywołania funkcji `complete_array()` powinien być rozmiar tablicy; funkcja powinna zamykać plik;
- funkcja `print_array()` - wyświetla elementy tablicy, będącej argumentem wywołania, drugim argumentem tej funkcji powinien być rozmiar tablicy.

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.