Lista 1

Utworzenie tablicy jest równoznaczne z alokacją pamięci na elementy tablicy (utworzeniem dynamicznej tablicy). W zadaniach należy pamiętać o zwolnieniu zasobów przydzielonych na stercie.

Zabronione jest korzystanie z bibliotek, włączając w to `stl` czy `boost`.

Zadanie 1.
Zaprojektować i zaimplementować funkcję w języku C++ realizującą następujące zadanie. Wyświetlanie na standardowym wyjściu wartości maksymalnej z trzech liczb a, b, c pobranych ze standardowego strumienia wejściowego (wyszukiwanie liniowe).

Zadanie 2.
Zaprojektować i zaimplementować algorytm realizujący następujące zadanie. Wyświetlanie na standardowym wyjściu informacji o punkcie przecięcia prostej $y = ax + b$ z osić OX, dla współczynników a, b pobranych ze standardowego strumienia wejściowego.

Zadanie 3.
Zaprojektować i zaimplementować algorytm realizujący następujące zadanie. Wyświetlanie na standardowym wyjściu pierwiastków równania kwadratowego $ax^2 + bx + c = 0$, dla współczynnikćw a, b, c pobranych ze standardowego strumienia wejściowego. Zastanowić się nad poprawnością rozwiązania z punktu widzenia matematyki i informatyki (wzory Viete'a, kodowanie FP2 a porównywanie liczb zmiennoprzecinkowych, itp.).

Przykładowe wywołanie funkcji `rozw_rown(a, b, c, x1, x2);`

Zadanie 4.
Zaprojektować i zaimplementować funkcję w języku C++ zwracającą n-ty wyrazu ciągu Fibonacciego z n przekazanego, jako parametr funkcji. Zadanie wykonać w sposób rekurencyjny.
`unsigned long fib_r(const unsigned int n);`

Zadanie 5.
Zaprojektować i zaimplementować funkcję w języku C++ zwracającą n-ty wyraz ciągu Fibonacciego z n przekazanego, jako parametr funkcji. Zadanie wykonać w sposób iteracyjny.
`unsigned long fib_i(const unsigned int n);`

Zadanie 6.(zadanie wspierające)

Zaprojektować i zaimplementować funkcję w języku C++ wypisującą na standardowym wyjściu elementy tablicy przekazanej do tej funkcji poprzez wskaźnik.
`void wypisz(int * tab, const unsigned int n);`

Zadanie 7.(zadanie wspierające)
Zaprojektować i zaimplementować funkcję w języku C++ dynamicznie przydzielającą pamięć dla tablicy liczb (np. całkowitych).
`int * utworz(const unsigned int n);` `void utworz(int * tab, const unsigned int n);`

Zadanie 8.
Zaprojektować i zaimplementować funkcję w języku C++ zwracającą utworzoną wewnątrz i wypełnioną kolejnymi wyrazami ciągu Fibonacciego tablicę T o rozmiarze n.
`int* wypelnij (unsigned int n);`

Zadanie 9.
Zaprojektować i zaimplementować funkcję w języku C++ zwracającą wartość silni z n przekazanego, jako parametr funkcji. Zadanie wykonać w sposób rekurencyjny.
`unsigned long sil_r(const unsigned int n);`

Zadanie 10.
Zaprojektować i zaimplementować funkcję w języku C++ zwracającą wartość silni z n przekazanego, jako parametr funkcji. Zadanie wykonać w sposób iteracyjny.
`unsigned long sil_i(const unsigned int n);`

Zadanie 11.
Zaprojektować i zaimplementować funkcję w języku C++ zwracającą sumę N-pierwszych wyrazów ciągu Fibonacciego.
Przykładowe wywołanie funkcji `suma_fib(n);`

Zadanie 12.
Zaprojektować i zaimplementować funkcję w języku C++ zwracającą liczbę nieparzystych wyrazów wśród N-pierwszych wyrazów ciągu Fibonacciego.
Przykładowe wywołanie funkcji `nieparzyste_fib(n);`

Zadanie 13.
Zaprojektować i zaimplementować funkcję w języku C++ wypełniającą tablicę T o rozmiarze n sześcianami kolejnych liczb naturalnych dodatnich. Tablicę utworzyć wewnątrz funkcji i odebrać jako argument od funkcji poprzez wskaźnik (wykorzystać funkcję `utworz`).

Zadanie 14.
Zaprojektować i zaimplementować funkcję w języku C++ wypełniającą tablicę T o rozmiarze n kolejnymi liczbami naturalnymi zaczynając od wartości 5. Tablicę utworzyć wewnątrz funkcji i odebrać jako argument od funkcji poprzez wskaźnik (wykorzystać funkcję `utworz`).

Zadanie 15.
Zaprojektować i zaimplementować funkcję w języku C++ wypełniającą tablicę T o rozmiarze n w następujący sposób: jeśli indeks jest liczbą parzystą, to do tablicy o tym indeksie przypisz podwojoną wartość indeksu, w przeciwnym razie przypisz wartość indeksu pomniejszoną o 1. Tablicę utworzyć wewnątrz funkcji i odebrać jako argument od funkcji poprzez wskaźnik (wykorzystać funkcję `utworz`).

Zadanie 16.
Zaprojektować i zaimplementować funkcję w języku C++ zwracającą wartość maksymalną z elementów tablicy T o rozmiarze n. Tablicę należy przekazać do funkcji poprzez wskaźnik.

Zadanie 17.
Zaprojektować i zaimplementować funkcję w języku C++ zwracającą, na której pozycji w tablicy T o rozmiarze n znajduje się wartość maksymalna. Tablicę należy przekazać do funkcji poprzez wskaźnik.

Zadanie 18.
Zaprojektować i zaimplementować funkcję w języku C++ zwracającą wartość średniej arytmetycznej z elementów tablicy T o rozmiarze n. Tablicę należy przekazać do funkcji poprzez wskaźnik.

Zadanie 19.
Zaprojektować i zaimplementować funkcję w języku C++ wypełniającą od końca tablicę T o rozmiarze n wartościami indeksu. Tablicę utworzyć wewnątrz funkcji i odebrać od funkcji jako argument poprzez wskaźnik. Licznik pętli należy zainicjować wartością 0.

Zadanie 20.
Zaprojektować i zaimplementować funkcję w języku C++ wypełniającą od końca tablicę T o rozmiarze n na przemian wartościami 0 i 2. Tablicę utworzyć wewnątrz funkcji i odebrać od funkcji jako argument poprzez wskaźnik.

Zadanie 21.
Zaprojektować i zaimplementować funkcję `polowki`, która przyjmuje dwa argumenty:

- tablicę (jako wskaźnik na pierwszy element) `tab`
- rozmiar tablicy `rozmiar`

i zwraca liczbę rzeczywistą.

Działanie funkcji powinno być następujące:

- od połowy tablicy (bez elementu, leżącego w środku tablicy) do początku (w tej kolejności) elementy powinny zostać wyzerowane;
- od połowy tablicy (włącznie) do końca (w tej kolejności) elementy powinny przyjmować wartości indeksu, powiększone o 1.

Dla tablic o parzystej liczbie elementów wybór elementu środkowego jest dowolny. Funkcja powinna zwrócić stosunek liczby elementów różnych od zera do liczby elementów równych 0.

Zadanie 22.
Zaprojektować i zaimplementować funkcję `kasuj_element`, która przyjmuje trzy argumenty:

- tablicę (jako wskaźnik na pierwszy element) `tab`
- rozmiar tablicy `rozmiar`
- `ktory`, będący liczbą całkowitą

i zwraca tablicę (jako wskaźnik na pierwszy element).

Działanie funkcji powinno przypominać usuwanie elementu z wektora, tj. funkcja powinna:

- utworzyć nową tablicę o rozmiarze `rozmiar-1`;
- skopiować wszystkie elementy z `tab` do nowej tablicy, pomijając element o indeksie `ktory`;

W ten sposób zostanie utworzona tablica, która zawiera wszystkie elementy ze źródłowej tablicy z pominięciem jednego elementu.

Funkcja powinna zwracać wskaźnik na utworzoną w tej funkcji tablicę.

Zadanie 23.
Zaprojektować i zaimplementować funkcję `dodaj_1`, która przyjmuje dwa argumenty:

- tablicę (jako wskaźnik na pierwszy element) `tab`;
- rozmiar tablicy `rozmiar`;

i zwraca tablicę (jako wskaźnik na pierwszy element.

Funkcja powinna utworzyć (i zwrócić) nową tablicę, która będzie zawierać elementy z tablicy `tab` powiększone o jeden.

Zadanie 24.
Zaprojektować i zaimplementować funkcję `tabliczka_mnozenia`, która przyjmuje jeden argument - `rozmiar`, będący liczbą całkowitą. Funkcja zwraca tablicę dwuwymiarową (jako wskaźnik na wskaźnik na element `[0][0]`).

Działanie funkcji powinno być następujące:

- powinna zostać dynamicznie utworzona tablica tablic liczb całkowitych (tablica dwuwymiarowa);
- każdy element tablicy powinien posiadać wartość równą `numer kolumny * numer wiersza`.

Zwrócony powinien zostać wskaźnik na tak wypełnioną tablicę.

Zadanie 25.
Zaprojektować i zaimplementować funkcję `statystyki`, która przyjmuje trzy argumenty:

- tablicę (jako wskaźnik na pierwszy element) `tab`;
- rozmiar tablicy `rozmiar`;
- maksymalna (największa) liczba całkowita `maksymalna`, która może wystąpić w tablicy `tab`.

i zwraca tablicę (jako wskaźnik na pierwszy element). Założenie: `maksymalna` jest liczbą naturalną, mniejszą od 10 000. Elementy tablicy `tab` są liczbami całkowitymi, większymi od -1.

Działanie funkcji powinno być następujące:

- powinna zostać utworzona tablica o rozmiarze `maksymalna`;
- w nowo utworzonej tablicy element o indeksie `i` przechowuję liczbę wystąpień wartości `i` w tablicy `tab`. Czyli `nowa[0] =` liczba zer w `tab`, `nowa[1] =` liczba wartości 1 w `tab` itd.

Zwracany jest wskaźnik do wypełnionej w ten sposób tablicy.

Zadanie 26.
Zaprojektować i zaimplementować funkcję `wytnij`, która przyjmuje cztery argumenty:

- tablicę (jako wskaźnik na pierwszy element) `tab`;
- rozmiar tablicy `rozmiar`;
- liczbę całkowitą `od`;
- liczbę całkowitą `az_do`;

i zwraca tablicę (jako wskaźnik na pierwszy element).

Funkcja powinna zwrócić nową tablicę, która będzie zawierać wartości z tablicy `tab`, zaczynając na elemencie o indeksie `od`, kończąc na elemencie o indeksie `az_do`. Na przykład, dla `tab = {1, 2, 3, 4}`, `rozmiar = 4`, `od = 2`, `az_do = 3` funkcja powinna zwrócić tablicę `{3, 4}`.

Zadanie 27.
Zaprojektować i zaimplementować funkcję `odwroc`, która przyjmuje dwa argumenty:

- tablicę (jako wskaźnik na pierwszy element) `tab`;
- rozmiar tablicy `rozmiar`;

i zwraca tablicę (jako wskaźnik na pierwszy element).

Działanie funkcji polega na utworzeniu (i zwróceniu) nowej tablicy i uzupełnieniu jej wartościami z tablicy `tab` w odwróconej kolejności (pierwszym elementem jest ostatni element `tab`, drugim - przedostatni element tablicy `tab` itd.).

Zadanie 28.
Zaprojektować i zaimplementować funkcję `skopiuj_do`, która przyjmuje trzy argumenty:

- tablicę (jako wskaźnik na pierwszy element) `tab`;
- rozmiar tablicy `rozmiar`;
- tablicę (jako wskaźnik na pierwszy element) `docelowa`

i nie zwraca żadnej wartości.

Działanie funkcji polega na skopiowaniu wartości z tablicy `tab` do tablicy `docelowa`.

Zadanie 29.
Zaprojektować i zaimplementować funkcję `roznica`, która przyjmuje trzy argumenty:

- wskaźnik `wsk_f` na funkcję, która przyjmuje jako argument wartość rzeczywistą i która zwraca wartość rzeczywistą;
- dwie liczby rzeczywiste `a` i `b`.

i zwraca wartość rzeczywistą.

Funkcja powinna:

- obliczyć wartości funkcji `wsk_f` dla argumentów `a` i `b`;
- zwrócić wartość bezwzględną różnicy wartości funkcji dla argumentów `a` i `b`.

Zadanie 30.
Zaprojektować i zaimplementować funkcję obliczającą sumę dwóch liczb przekazanych jako parametry oraz zwracającą wynik tej operacji.

Zadanie 31.
Zaprojektować i zaimplementować funkcję obliczającą rónicę dwóch liczb przekazanych jako parametry oraz zwracającą wynik tej operacji.

Zadanie 32.
Zaprojektować i zaimplementować funkcję obliczającą iloczyn dwóch liczb przekazanych jako parametry oraz zwracającą wynik tej operacji.

Zadanie 33.
Zaprojektować i zaimplementować funkcję obliczającą iloraz dwóch liczb przekazanych jako parametry oraz zwracającą wynik tej operacji.

Zadanie 34.
Zaprojektować i zaimplementować funkcję wyznaczającą wartość x^2

Zadanie 35.
Zaprojektować i zaimplementować funkcję wyznaczającą wartość x^3

Zadanie 36.
Zaprojektować i zaimplementować funkcję obliczającą wartość poniższego równania z wykorzytsaniem wcześniej opracowanych funkcji (nie korzystać

jawnie z operatorów $+, -, *, /$)

$$f(x) = (10 \cdot x^3 + 3.14 \cdot x^2) \cdot (\frac{x}{3} - \frac{1}{x^2})$$

Zadanie 37.
Zaprojektować i zaimplementować funkcję obliczającą obwód trójkąta.

Zadanie 38.
Zaprojektować i zaimplementować funkcję obliczającą pole powierzchni trójkąta (wzór Herona).

Zadanie 39.
Zaprojektować i zaimplementować program symulujący działanie kalkulatora wykorzystając poprzednio zaimplementowane funckje. Program powinien posiadać menu umożliwiające wybór operacji do wykonania (dodawanie, odejmowanie, mnożenie, dzielenie, pierwiastkowanie, potęgwanie, zakończenie działania programu). Mechanizm menu kalkulatora należy zaimplementować z wykorzysatniem instrukcji `switch` i przetestować poniższym fragmentem kodu źródłowego:

```
int main()
{
 while(kalkulator());

 return 0;
}
```