Zajęcia 4 – procedury i funkcje

1. Napisz funkcję `double min(double tab[], int rozmiar)`, która znajduje i zwraca minimalną wartość w podanej tablicy.

   Następnie napisz funkcję
   `void max_ref(double tab[], int rozmiar, double & wynik)`, która znajduje maksymalną wartość w podanej tablicy, ale wynik zwraca przez swój trzeci parametr, tj. `wynik`.

2. Napisz funkcję, która zwraca wartość silni dla podanej liczby n. Funkcja powinna być napisana w dwóch wersjach: iteracyjnej i rekurencyjnej.

   Przykład deklaracji funkcji:

   ```
   long silnia(long n);
   long silniaRekunrencja(long n);
   ```

3. Napisz funkcję, która zwraca wartość n-tego wyrazu ciągu Fibonacciego. Funkcja powinna być napisana w dwóch wersjach: iteracyjnej i rekurencyjnej.

   Przykład deklaracji funkcji:

   ```
   unsigned fib(unsigned n);
   unsigned fibRekurencyjna(unsigned n);
   ```

4. Napisz program, który umożliwia szyfrowanie i deszyfrowanie podanego ciągu znaków przy użyciu szyfru Cezara (który jest szczególnym przypadkiem szyfru podstawieniowego monoalfabetycznego). Szyfrowanie ma realizować funkcja szyfruj, która przyjmuje 4 parametry:

   - łańcuch znaków do zaszyfrowania, zakończony znakiem '\0'
   - przesunięcie alfabetu,
   - tablicę znaków, w której należy zapisać wynik.

   Zamianie mają podlegać jedynie litery alfabetu angielskiego. Wynikiem funkcji ma być zaszyfrowany tekst.

   Przykład deklaracji funkcji:

```
void szyfruj(char tekst[], int przesuniecie, char wynik[]);
void deszyfruj(char tekst[], int przesuniecie, char wynik[]);
```

Przykład:

```
Szyfr Cezara z przesunięciem 1 dokonuje zamiany wg. schematu:
  a -> b
  b -> c
  ...
  z -> a
Szyfr Cezara z przesunięciem 3 dokonuje zamiany wg. schematu:
  a -> d
  b -> e
  ...
  z -> d
```

5. Napisz funkcję `strfind`, która szuka w tekście (pierwszy parametr) podanej frazy (drugi parametr). Wynikiem funkcji ma być indeks znaku, od którego podana fraza zaczyna się w tekście lub -1, jeżeli tekst nie zawiera szukanej frazy. Wielkość liter w podanych ciągach nie ma znaczenia.

   Przykład:

   Dla podanego fragmentu programu:

```
char zdanie[] = "Jutro jest egzamin z programowania.";
char fraza[] = "Program";
cout << "Szukam w \"" << zdanie << "\"" << endl;
cout << "\t\"" << fraza << "\" : " << strfind(zdanie, fraza) << endl;
cout << "\t\"jutro JEST\" : " << strfind(zdanie, "jutro JEST") << endl;
cout << "\t\"WDI\" : " << strfind(zdanie, "WDI") << endl;
cout << "Szukam w \"" << fraza << "\"" << endl;
cout << "\t\"" << zdanie << "\" : " << strfind(fraza, zdanie) << endl;
cout << "\t\"" << fraza << "\" : " << strfind(fraza, fraza) << endl;
```

   na ekranie powinno zostać wyświetlone:

```
Szukam w "Jutro jest egzamin z programowania."
    "Program" : 21
    "jutro JEST" : 0
    "WDI" : -1
Szukam w "Program"
    "Jutro jest egzamin z programowania." : -1
    "Program" : 0
```

6. Napisz program obliczający wartość wielomianu stopnia n. Przykład deklaracji funkcji:

```
double obl_wiel(double x, int n, ...)
```

W związku ze zmienną liczbą parametrów przydatne będą makra z biblioteki `cstdarg`:

```
va_list ...
va_start(..., ...)
wsp = va_arg(..., ...)
va_end(...)
```

7. Dywan Sierpińskiego to fraktal otrzymany z kwadratu za pomocą podzielenia go na dziewięć (3x3) mniejszych kwadratów, usunięcia środkowego kwadratu i ponownego rekurencyjnego zastosowania tej samej procedury do każdego z pozostałych ośmiu kwadratów.

   Napisz *rekurencyjną* funkcję, która „rysuje" dywan Sierpińskiego w konsoli za pomocą znaków '#' oraz spacji. Funkcja przyjmuje parametr n określający stopień szczegółowości rysowanego dywanu. Rysowanie odbywa się w dwuwymiarowej tablicy znaków, przy czym zaczerniony kwadrat oznaczany jest znakiem '#', natomiast pusty – spacją. Długość boku dywanu dla stopnia n wynosi 3^{n-1} znaków.
   Przykładowe wydruki dla kilku początkowych wartości n przedstawiono poniżej.

```
n=1   n=2    n=3              n=4

#     ###    #########        ###########################
      # #    # ## ## #        # ## ## ## ## ## ## ## ## #
      ###    #########        ###########################
             ###   ###        ###   ######   ######   ###
             # #   # #        # #   # ## #   # ## #   # #
             ###   ###        ###   ######   ######   ###
             #########        ###########################
             # ## ## #        # ## ## ## ## ## ## ## ## #
             #########        ###########################
                              #########        #########
                              # ## ## #        # ## ## #
                              #########        #########
                              ###   ###        ###   ###
                              # #   # #        # #   # #
                              ###   ###        ###   ###
                              #########        #########
                              # ## ## #        # ## ## #
                              #########        #########
                              ###########################
                              # ## ## ## ## ## ## ## ## #
                              ###########################
                              ###   ######   ######   ###
                              # #   # ## #   # ## #   # #
                              ###   ######   ######   ###
                              ###########################
                              # ## ## ## ## ## ## ## ## #
                              ###########################
```