Zajęcia 4 – procedury i funkcje

1. Napisz funkcję, która dokonuje dodania dwóch liczb przekazanych jako parametry. Następnie:

   - zmień wartości zmiennych przekazanych jako parametry wewnątrz tej funkcji,
   - ustaw jeden z parametrów do przekazywania przez wartość, a drugi przez referencję,
   - wyświetl wartości obydwóch zmiennych w programie, w którym wywołana została funkcja.

   Na ekranie należy wypisać wartość zwróconą przez funkcję oraz wartości parametrów przed i po wywołaniu funkcji.

   Przykład deklaracji funkcji:

   ```
   int suma(int a, int & b); // gdzie b przekazane jest przez referencję
   ```

2. 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);
   ```

3. 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);
   ```

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 2 parametry: pierwszy to łańcuch znaków, natomiast drugi

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

Przykład deklaracji funkcji:

```
string szyfruj(string tekst, unsigned int przesuniecie);
string deszyfruj(string tekst, unsigned int przesuniecie);
```

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 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(...)
   ```

6. 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
```

Zajęcia 5 – łańcuchy znaków (char [])

1. Zdefiniować funkcję strpos(char str[], char z), która zwraca indeks na którym znajduje się znak 'z' w podanym łańcuchu 'str' Jeżeli znak 'z' nie występuje w łańcuchu to funkcja powinna zwrócić -1.

   ```
   int strpos(char str[], char z)
   ```

2. Zdefiniować funkcję strint(char str[]), która zamienia liczbę całkowitą zapisaną w postaci łańcucha na liczbę całkowitą typu int. Funkcja powinna przerywać konwersję w momencie napotkania pierwszego znaku nie należącego do zapisu liczby, zatem np. dla strint("-13krowa") wynikiem powinno być -13.

   ```
   int strint(char str[])
   ```

   Przykłady:

   ```
   strint("+12") - wynik 12
   strint("0001") - wynik 1
   strint("991-234-23") - wynik 991
   strint("-12e5") - wynik -12*10^5 = -120000
   strint("-12e-5") - wynik -12
   strint("+zonk") - wynik 0
   strint("") - wynik 0
   ```

3. Zdefiniować funkcję strfind(char gdzie[], char co[]), która szuka łańcucha 'co' w łańcuchu 'gdzie' i jeżeli go znajdzie, to jej wynikiem jest pozycja, na której ten łańcuch zaczyna się w łańcuchy 'gdzie'. Jeżeli nie udało się znaleźć łańcucha to wtedy wynikiem ma być -1.

   ```
   int strfind(char gdzie[], char co[])
   ```

   Przykłady:

```
strfind("Ala ma kota", "ma") - wynik to 4
strfind("Ala ma kota", "Ala ma kota") - wynik to 0
strfind("Ala ma kota", "") - wynik to 0, bo pusty łańcuch jest
                             podłańcuchem każdego innego łańcucha
strfind("Pies", "jakis napis") - wynik to -1
strfind("Ala ma kota", "pies") - wynik to -1
```

4. Zdefiniować procedurę strcut(char lan[], int od, int ile), która wycina z podanego łańcucha wszystko co znajduje się poza podanym zakresem. Po zakończeniu działania łańcuch lan zawiera tylko to, co pierwotnie znajdowało się na pozycjach od - od+ile.
   W procedurze można wykorzystać gotową funkcję: strcpy(lanDoc, lanZr), która kopiuje zawartość łańcucha lanZr do łańcucha lanDoc, należy pamiętać o odpowiednim przydziale pamięci.

   ```
   void strcut(char lan[], int od, int ile)
   ```

   Przykłady:

   ```
   strcut("Ala ma kota", 1, 2) - wynik to "la"
   strcut("Ala ma kota", 5, 4) - wynik to "a ko"
   ```

5. Zdefiniować funkcję strprefix(char str1[], char str2[]), która sprawdza, czy łańcuch str2 jest prefiksem łańcucha str1.

   ```
   bool strprefix(char str1[], char str2[])
   ```

   Przykłady:

   ```
   strprefix("Alibaba", "Ali") - wynik true, ponieważ wyraz
                                 "Alibaba" zaczyna się wyrazem "Ali".
   strprefix("Alibaba", "Alibaba") - wynik true, ponieważ wyraz jest
                                     zawsze swoim prefiksem.
   strprefix("Kot", "Pies") - wynik false, ponieważ wyraz "Pies"
                              nie jest prefiksem wyrazu "Kot"
   ```

6. Zdefiniować funkcję strcountfind(char gdzie[], char co[]), która zlicza wystąpienia łańcucha 'co' w łańcuchu 'gdzie', jej wynikiem jest wyznaczona liczba wystąpień. Jeżeli nie udało się znaleźć łańcucha to wtedy wynikiem ma być 0.
   W funkcji można wykorzystać gotową funkcję: strcpy(lanDoc, lanZr), która kopiuje zawartość łańcucha lanZr do łańcucha lanDoc, należy pamiętać o odpowiednim przydziale pamięci.

```
int strcountfind(char gdzie[], char co[])
```

Przykłady:

```
strcountfind("Ala ma kota", "ma") - wynik to 1
strcountfind("mama ma kota", "ma") - wynik to 3
strcountfind("Ala mmaa ma kota", "ma") - wynik to 2
strcountfind("Ala ma kota", "Asia") - wynik to 0
```

7. Napisać program, który wykorzystując część z zaimplementowanych wcześniej funkcji wyznacza:

   - Sumę wszystkich liczb znajdujących się w tablicy (jako liczbę traktuje się łańcuch, którego początkiem jest liczba - format jak w funkcji strint().
   - Łańcuch będący połączeniem wszystkich komórek tablicy, których wartość łańcucha nie jest liczbą (definicja liczby analogiczna do pkt. 1).
   - Liczbę wystąpień określonej frazy we wszystkich komórkach „nieliczbowych" tablicy.
   - Liczbę wystąpień określonej frazy w łańcuchu, o którym mowa w pkt. 2.
   - Stosunek wystąpień frazy w komórkach tablicy (pkt. 3) do liczby wystąpień w powstałym łańcuchu (pkt. 4).

   Przykład:

```
Tablica, o której mowa w zadaniu:
zadania[N][M]={"mamla", " mama ", "+12", "0001", "991-234-3",
               "-12e5", "-12e-5", "+zonmakm", "ax2", "amakotma"};
               // gdzie N=M=10;
Szukana fraza:
f[N]="ma";
Wynik wyświetlony na konsolę:
Pkt. 1: -1199008
Pkt. 2: mamla mama +zonmakmax2amakotma
Pkt. 3: 6
Pkt. 4: 7
Pkt. 5: 0.857143
```