LISTA ZADAŃ Z PRZEDMIOTU PODSTAWY PROGRAMOWANIA

Zadanie 1.
Celem tego zadania jest nabycie umiejętności pracy z tablicami znakowymi oraz argumentami uruchomienia programu. Wszystkie potrzebne argumenty funkcji należy pobrać jako parametry uruchomienia programu.

W tym zadaniu nie wolno korzystać z funkcji bibliotecznych dostępnych po dołączeniu pliku nagłówkowego `cstring`.

Zaprojektuj i zaimplementuj następujące funkcje:

1. `dlugosc` – operującą na tablicy znaków i zwracającą długość napisu w tablicy źródłowej

   ```
   size_t dlugosc(const char* src);
   ```

   ---

2. `kopiuj` – operującą na tablicy znaków i służącą do kopiowania tablicy źródłowej do tablicy docelowej

   ```
   char* kopiuj(const char* src);
   ```

3. `kopiuj` – operującą na tablicy znaków i służącą do kopiowania tablicy źródłowej do tablicy docelowej

   ```
   void kopiuj(char*& dest, const char* src);
   ```

   ---

4. `dolacz` – operującą na tablicy znaków i służącą do doklejenia pierwszych `n` znaków z napisu źródłowego do napisu docelowego, plus kończący znak `NULL`.

   ```
   char* dolacz(const char* src, const char* ins, const size_t
   n);
   ```

5. `dolacz` – operującą na tablicy znaków i służącą do doklejenia pierwszych `n` znaków ze źródła do miejsca docelowego, plus kończący znak `NULL`. Jeśli długość tablicy w źródle jest mniejsza niż `n`, kopiowana jest tylko zawartość do kończącego znaku `NULL`

   ```
   void dolacz(char*& dest, const char* src, const char* ins, const
   size_t n);
   ```

PRZYKŁAD dla poniższych funkcji `wstaw`:

wstawienie `X` do `Ala ma kota` daje `XAlaX maX koXta`

6. `wstaw` – operującą na tablicy znaków i służącą do wstawienia co trzeci element tej tablicy symbolu przekazanego jako parametr wywołania tej funkcji, a zwracającą wskaźnik do nowo utworzonej wewnątrz tej funkcji tablicy w pamięci dynamicznej

   ```
   char* wstaw(const char* src, char s);
   ```

7. funkcję `wstaw` – operującą na tablicy znaków i służącą do wstawienia co trzeci element tej tablicy symbolu przekazanego jako parametr wywołania tej funkcji, pamięć dla tablicy docelowej należy przydzielić wewnątrz funkcji

   ```
   void wstaw(char*& dest, const char* src, char s);
   ```

8. `porownaj` – porównującą dwa napisy w stylu języka C przekazane jako argumenty tej funkcji. Funkcja zwraca wartość:

   - większą od zera, gdy pierwszy niepasujący znak ma większą wartość w napisie 1 niż w napisie 2,
   - równą zero, gdy zawartość obu napisów jest równa,
   - mniejszą od zera, gdy pierwszy niepasujący znak ma mniejszą wartość w napisie 1 niż w napisie 2.

   ```
   int porownaj(const char* str1, const char* str2);
   ```

---

9. `szukaj` – operującą na tablicy znaków i zwracającą liczbę wystąpień symbolu przekazanego jako parametr wywołania tej funkcji,

---

10. `zamien_na_duze` – zwracającą wskaźnik do nowo utworzonej wewnątrz tej funkcji tablicy w pamięci dynamicznej, do której przekopiowano tablicę źródłową zamieniając wszystkie małe litery na duże.
11. `zamien_na_male` – zwracającą wskaźnik do nowo utworzonej wewnątrz tej funkcji tablicy w pamięci dynamicznej, do której przekopiowano tablicę źródłową zamieniając wszystkie duże litery na małe.

Zaprojektuj i zaimplementuj funkcję `main` zapewniającą prawidłowe działanie programu oraz wywołania wszystkich funkcji.

Ciąg znaków (napis w stylu języka C) oraz symbol należy przekazać, **jako parametry uruchomienia programu**.

Przy realizacji zadań należy pamiętać o używaniu modyfikatora `const`.

Zadanie 2.
Celem tego zadania jest doskonalenie umiejętności pracy z tablicami znakowymi, literaturą, analizą treści, logicznym myśleniem oraz wyciąganiem wniosków.

Zaprojektuj i zaimplementuj rozwiązanie problemów przedstawionych w *Zadanie 1* z wykorzystaniem funkcji bibliotecznych dostępnych po dołączeniu pliku nagłówkowego `cstring`. Rozwiązanie może wykorzystywać dodatkowe funkcje poza funkcją `main`.

Zadanie 3.
Celem tego zadania jest nabycie umiejętności pracy z typem `string` oraz parametrami uruchomienia programu.

Zaprojektuj i zaimplementuj funkcję/instrukcję:

- kopiującą napis źródłowy `const string src` do napisu docelowego `char* dest`,
- kopiującą napis źródłowy `const string src` do napisu docelowego `string dest` wykorzystując operator przypisania (w funkcji `main`),
- łączącą dwa napisy `const string str1` i `const string str2` wykorzystując operator dodawania (w funkcji `main`),
- kopiującą napis źródłowy `const char* src` do napisu docelowego `string& dest` usuwając `n` pierwszych znaków, a nic nie zwracającą,
- kopiującą napis źródłowy `const string src` do napisu docelowego `string dest` usuwając `n` pierwszych znaków i zwracającą `dest`,
- zwracającą pierwszą pozycję `pos` wystąpienia poszukiwanego napisu `const char* str` w napisie źródłowym `const string src`,
- zwracającą pierwszą pozycję `pos` wystąpienia poszukiwanego napisu `const string str` w napisie źródłowym `const string src`,
- zwracającą pierwszą pozycję `pos`, od której zaczyna się napis `const string str` poszukiwany w napisie źródłowym `const string src`,
- zwracającą wynik porównania dwóch napisów `const string str1` oraz `const string str2`. Kiedy napis `str1` jest mniejszy od napisu `str2` funkcja zwraca wartość -1, kiedy napis `str1` jest większy od napisu `str2` funkcja zwraca wartość 1, kiedy napis `str1` jest taki sam, jak napis `str2` funkcja zwraca wartość 0.

Wszystkie potrzebne do działaania programu dane należy pobrać z linii uruchomienia programu.

Zaprojektuj i zaimplementuj funkcję `main` zapewniającą prawidłowe działanie programu i pozwalającej **przetestować** wszystkie zdefiniowane funkcje/instrukcje.

Zadanie 4.
Celem tego zadania jest doskonalenie umiejętności pracy z typem `string` oraz parametrami uruchomienia programu.

Zaprojektuj i zaimplementuj instrukcje prezentujące działanie funkcji składowych klasy `string`:

- przypisującą nową wartość do napisu, zastępując jego obecną zawartość (`string::assign`),
- rozszerzającą napis, dodając dodatkowe znaki na końcu jego bieżącej wartości (`string::append`),
- zwracającą odwołanie do znaku znajdującego się na pozycji w ciągu znaków. Funkcja ta automatycznie sprawdza, czy zwracana pozycja jest prawidłową pozycją znaku w ciągu (`string::at`),
- zwracającą referencję do ostatniego znaku w napisie i zamieniającą go na wybrany znak (`string::back`),
- porównując wartość obiektu ciągu (lub podciągu) z sekwencją znaków określoną przez jego argumenty (`string::compare`),
- kopiującą podciąg bieżącej wartości obiektu `string` do tablicy znaków wskazywanej przez `str`. Ten podciąg zawiera liczbę kopiowanych znaków rozpoczynających się od danej pozycji (`string::copy`),
- przeszukującą napis pod kątem pierwszego wystąpienia sekwencji określonej przez jego argumenty. Jeśli określono miejsce, wyszukiwanie obejmuje tylko znaki znajdujące się w tym miejscu lub po nim, ignorując wszelkie możliwe wystąpienia zawierające znaki przed wybranym miejscem (`string::find`),
- wstawiającą dodatkowe znaki do napisu tuż przed pozycją wskazanego znaku (`string::insert`),
- zastępującą nową zawartością część napisu rozpoczynającą się od pozycji wystąpienia znaku i obejmującą długość znaków (`string::replace`),
- pobierającą napis ze strumienia aż do znalezienia znaku rozgraniczającego (`getline`). Do przetestowania należy wykorzystać również prze-

kierowanie danych z pliku do strumienia standardowego wejścia.

Wszystkie potrzebne do działaania programu dane należy pobrać z linii uruchomienia programu.

Zaprojektuj i zaimplementuj funkcję `main` zapewniającą prawidłowe działanie programu i pozwalającej **przetestować** wszystkie funkcje/instrukcje.

Zadanie 5.
Celem tego zadania jest ćwiczenie umiejętności pracy z typem `string`.

Zaprojektuj i zaimplementuj funkcję `liczba_wyrazow` zwracającą liczbę wyrazów w zdaniu przekazanym tej funkcji poprzez argument. Zakładamy, że zdanie jest poprawne gramatycznie, ale mogą zdarzać się błędy edytorskie np. wiele spacji pomiędzy wyrazami lub spacja przed znakami przestankowymi. Zdanie należy pobrać, jako parametr uruchomienia programu.

Zaprojektuj i zaimplementuj funkcję `main` zapewniającą prawidłowe działanie programu.

Zadanie 6.
Celem tego zadania jest ćwiczenie umiejętności pracy z typem `string`.

Zaprojektuj i zaimplementuj funkcję orzekającą `palindrom` stwierdzającą czy zdanie przekazane, jako argument tej funkcji jest palindromem.[1] Należy rozważyć sytuację, gdy w zdaniu występują znaki tj. "? , . - ( ) Zanki te nie wliczają się do palindromu.

Zdanie należy pobrać, jako parametr uruchomienia programu.

Zaprojektuj i zaimplementuj funkcję `main` zapewniającą prawidłowe działanie programu.

Zadanie 7.
Celem tego zadania jest ćwiczenie umiejętności pracy z typem `string`.

Szyfrowanie tekstów metodą Cezara.
W szyfrowaniu metodą Cezara każdemu znakowi tekstu jawnego odpowiada znak przesunięty o określoną liczbę[2] znaków w alfabecie. Przyjmujemy, że jest to alfabet łaciński, który zawiera tylko 26 liter:
A, B, C, D, E, F, G, H, I, J, K, L, M, N, O, P, Q, R, S, T, U, V, W, X, Y, Z.

[1] palindrom to wyrażenie, które czytane od końca (wspak) brzmi tak samo jak wyjściowe.

[2] klucz (`bias`) – liczba pobrana jako parametr uruchomienia programu

W metodzie tej nie rozróżnia się liter wielkich i małych, tak więc wiadomość[3], będzie zamieniany na wielkie litery.
Np.: dla `bias = 3` litera A staje się literą D, litera B staje się literą E, a litera Z literą C, itd.
Jest to bardzo prymitywny szyfr, który można w bardzo prosty sposób złamać, więc nie gwarantuje żadnego bezpieczeństwa.

Zaprojektuj i zaimplementuj:

- funkcję `szyfruj` przyjmującą jako argument wiadomość do zaszyfrowania `wiadomosc` i klucz `bias` oraz zwracającą zaszyfrowany tekst,
- funkcję `rozszyfruj` przyjmującą jako argument zaszyfrowaną wiadomość i klucz oraz zwracającą rozszyfrowany tekst.

Zaprojektuj i zaimplementuj funkcję `main` zapewniającą prawidłowe działanie programu.

Zadanie 8.
Celem tego zadania jest podsumowanie wiedzy i umiejętności z używania typu `string`. Szyfrowanie tekstów metodą XOR.
Zaszyfruj i odszyfruj tekst pobrany jako parametr uruchomienia programu, stosując metodę XOR. Operacja logiczna XOR:

$$0 \text{ XOR } 0 = 0$$
$$0 \text{ XOR } 1 = 1$$
$$1 \text{ XOR } 0 = 1$$
$$1 \text{ XOR } 1 = 0$$

Szyfrowanie polega na zastosowaniu operacji XOR na kodach ASCII znaków wejściowych i klucza (hasła) będącego jednym znakiem. Odszyfrowanie następuje w ten sam sposób, tym samym algorytmem, z zastosowaniem takiego samego klucza.

Wyjaśnienie działania szyfru:

> Szyfrowanie polega na zastosowaniu operacji XOR na kodach ASCII znaków wejściowych i klucza (hasła) będącego jednym znakiem. Odszyfrowanie następuje w ten sam sposób, tym samym algorytmem, z zastosowaniem takiego samego klucza.

[3]tekst (`wiadomosc`) – zdanie pobrane jako prarametr uruchomienia programu

Zbudowany program będzie zatem służył jednocześnie do szyfrowania i odszyfrowywania.
UWAGA! Operowanie na kodach ASCII za pomocą funkcji XOR może doprowadzić do otrzymania tzw. znaku sterującego. W extendedASCII są to znaki o kodach od 0 do 31. Znaki te są niedrukowalne i sterują pracą urządzeń oraz mogą powodować zmianę interpretacji innych znaków. Np. próba zakodowania tekstu `Ala ma kota` z kluczem `a` w wyniku operacji `a XOR a` da wartość `NULL`. DLATEGO, lepiej jest dla tekstów literowych (zdanie, wiadomość) używać klucza w postaci cyfry.

Przykład:

- Tekst do zaszyfrowania: Kot
- Klucz: Z
- Zamiana kodów ASCII na liczby binarne

  kod_ASCII(k) = 107(10) = 1101011(2)

  kod_ASCII(o) = 111(10) = 1101111(2)

  kod_ASCII(t) = 116(10) = 1110100(2)

  kod_ASCII(Z) = 90(10) = 1011010(2)
- Operacje XOR:

  0 1 1 0 1 0 1 1

  0 1 0 1 1 0 1 0

  ———————

  0 0 1 1 0 0 0 1

  0 1 1 0 1 1 1 1

  0 1 0 1 1 0 1 0

  ———————

  0 0 1 1 0 1 0 1

  0 1 1 1 0 1 0 0

  0 1 0 1 1 0 1 0

  ———————

0 0 1 0 1 1 1 0

- Zamiana liczby binarnej na kod ASCII

  $00110001(2) = 49(10) \rightarrow$ znak $= 1$

  $00110101(2) = 53(10) \rightarrow$ znak $= 5$

  $00101110(2) = 46(10) \rightarrow$ znak $=$. (kropka)

Zatem kodowanie tekstu `kot` z kluczem `Z` dało tekst `15.`

Zaprojektuj i zaimplementuj:

- funkcję `szyfruj` przyjmującą jako argument wiadomość do zaszyfrowania i klucz oraz zwracającą zaszyfrowany tekst,
- funkcję `rozszyfruj` przyjmującą jako argument zaszyfrowaną wiadomość i klucz oraz zwracającą rozszyfrowany tekst,
- dodatkowe funkcje ułatwiające przejrzysty zapis kodu źródłowego (jedna funkcja = jedno działanie).

Zaprojektuj i zaimplementuj funkcję `main` zapewniającą prawidłowe działanie programu. Wiadomość i klucz należy pobrać jako parametr uruchomienia programu.