Temat: Łańcuchy znaków w języku C++

Klasa string:

- Klasa string jest listą przystosowaną do przechowywania i operowania na łańcuchach znaków (zmiennych tekstowych).
- Klasa dysponuje szeregiem operatorów i metod, które można wykorzystać do przetwarzania tekstu zawartego w obiekcie tej klasy.
- Chcąc korzystać z klasy **string** musimy do programu dodać nagłówek: **#include <string>**

Przeciążone operatory:

- **==, !=, <=, >, >=** – porównanie dwóch łańcuchów znaków,
- **=** – przypisanie,
- **+, +=** – konkatenacja dwóch łańcuchów znaków (złączenie jednego łańcucha z drugim),
- **[]** – odwołanie się do konkretnego znaku w łańcuchu znaków.

Ważniejsze metody:

- **size()** – zwraca liczbę znaków w łańcuchu znaków,
- **length()** – jak wyżej,
- **clear()** – czyści zawartość łańcucha znaków,
- **empty()** – sprawdza, czy łańcuch znaków jest pusty,
- **resize()** – zmienia wielkość łańcucha znaków,
- **max_size()** – zwraca maksymalną liczbę znaków, jaką może przechować łańcuch znaków,
- **append()** – dodaje do łańcucha znaków kolejne znaki lub inny łańcuch znaków,

- **push_back()** – dodaje znak na koniec łańcucha znaków,
- **assign()** – podmienia łańcuch znaków na inny,
- **insert()** – wstawia łańcuch znaków w określone miejsce,
- **replace()** – podmienia łańcuch znaków począwszy od podanej pozycji,
- **swap()** – zamienia ze sobą dwa łańcuchy znaków,
- **pop_back()** – usuwa ostatni znak,
- **c_str()** – rzutuje typ string na char *,
- **copy()** – kopiuje podciąg znaków ze łańcucha znaków i zapisuje go w drugim łańcuchu znaków,
- **find()** – wyszukuje pozycję wystąpienia danego podciągu w łańcuchu znaków,
- **compare()** – porównuje dwa łańcuchy znaków ze sobą,
- **getline()** – zapisanie całej linii (z białymi znakami) do zmiennej z łańcuchem znaków.

Dokumentacja:

https://www.cplusplus.com/reference/string/string/

https://en.cppreference.com/w/cpp/string/basic_string

Tablica znaków ASCII:

https://www.rapidtables.com/code/text/ascii-table.html

Zadania przykładowe:

1. Pobierz słowo od użytkownika. Wyświetl jego długość oraz pierwszy i ostatni znak. Jeżeli słowo zaczyna się literą „a” lub „A”, to wypisz informację „Pierwszy znak jest pierwsza litera alfabetu.”.

```cpp
#include <iostream>
#include <string>
using namespace std;

int main()
{
  string s;
  cout<<"Wpisz slowo: ";
  cin>>s;
  cout<<"\nDlugosc slowa: "<<s.length()<<endl;
  if(s.length()>0) {
    cout<<"Pierwszy znak: "<<s[0]<<endl;
    cout<<"Ostatni znak: "<<s[s.length()-1]<<endl;
    if(s[0]=='A' || s[0]=='a')
      cout<<"Pierwszy znak jest pierwsza litera alfabetu."<<endl;
  }
  return 0;
}
```

```
Wpisz slowo: Arystoteles

Dlugosc slowa: 11
Pierwszy znak: A
Ostatni znak: s
Pierwszy znak jest pierwsza litera alfabetu.
```

2. Wczytaj dwa słowa. Przeliteruj drugie słowo i sprawdź, czy drugie słowo zawiera się w pierwszym. Wyniki wyświetl na ekranie:
   Wskazówka: https://pl.wikibooks.org/wiki/C%2B%2B/String

```cpp
#include <iostream>
#include <string>
using namespace std;

int main()
{
  string s1, s2;
  cout<<"Podaj slowo 1: ";
  cin>>s1;
  cout<<"Podaj slowo 2: ";
  cin>>s2;
  cout<<endl<<"Literujemy slowo \""<<s2<<"\": ";
  for(int i=0;i<s2.size();i++)
    cout<<s2[i]<<", ";
  cout<<endl<<endl;
  if(s1.find(s2)==string::npos)
    cout<<"Slowo \""<<s2<<"\" nie zawiera sie w slowie \""<<s1<<"\"."<<endl;
  else
    cout<<"Slowo \""<<s2<<"\" zawiera sie w slowie \""<<s1<<"\"."<<endl;
  return 0;
}
```

```
Podaj slowo 1: stokrotka
Podaj slowo 2: tok

Literujemy slowo "tok": t, o, k,

Slowo "tok" zawiera sie w slowie "stokrotka".
```

Sprawdź jaki będzie efekt jeśli pierwsze słowo zastąpimy zdaniem „Moja stokrotka.”.

```cpp
#include <iostream>
#include <limits>
#include <string>
using namespace std;

int main()
{
  string s1, s2;
  cout<<"Podaj zdanie: ";
  getline(cin,s1);
  cout<<"Podaj slowo: ";
  cin>>s2;
  cout<<endl<<"Literujemy slowo \""<<s2<<"\": ";
  for(int i=0;i<s2.size();i++)
    cout<<s2[i]<<", ";
  cout<<endl<<endl;
  if(s1.find(s2))
    cout<<"Slowo \""<<s2<<"\" zawiera sie w \""<<s1<<"\"."<<endl;
  else
    cout<<"Slowo \""<<s2<<"\" nie zawiera sie w \""<<s1<<"\"."<<endl;
  return 0;
}
```

```
Podaj zdanie: Mniszek lekarski
Podaj slowo: lekarz

Literujemy slowo "lekarz": l, e, k, a, r, z,

Slowo "lekarz" nie zawiera sie w slowie "Mniszek lekarski".
```

3. Napisać funkcję o nagłówku **bool czyPalindrom(string s)** sprawdzającą, czy jej parametr jest palindromem. Wczytaj od użytkownika słowo i sprawdź przy pomocy napisanej funkcji, czy jest ono palindromem. Przetestuj też działanie funkcji dla słów: „zakaz”, „nakaz”, „potop”, „a” i „aa”.(Palindrom jest to pojedyncze słowo lub całe zdanie, który czytane od tyłu i od przodu brzmi tak samo.)

```cpp
#include <iostream>
#include <string>
using namespace std;

bool czyPalindrom(string s) {
  int d=s.length();
  for(int i=0;i<=d/2;i++) {
    if(s[i]!=s[d-1-i])
      return false;
  }
  return true;
}

void analiza(string s) {
  if(czyPalindrom(s))
    cout<<"Slowo \""<<s<<"\" jest palindromem."<<endl;
  else
    cout<<"Slowo \""<<s<<"\" nie jest palindromem."<<endl;
}

int main()
{
  string s;
  cout<<"Wpisz slowo: ";
  cin>>s;
  analiza(s);
  analiza("zakaz");
  analiza("nakaz");
  analiza("potop");
  analiza("a");
  analiza("aa");
  return 0;
}
```

```
Wpisz slowo: kajak
Slowo "kajak" jest palindromem.
Slowo "zakaz" jest palindromem.
Slowo "nakaz" nie jest palindromem.
Slowo "potop" jest palindromem.
Slowo "a" jest palindromem.
Slowo "aa" jest palindromem.
```

Porównaj implementację funkcji *czyPalindrom* z poniższą implementacją.

```
1   #include <iostream>
2   #include <string>
3   #include <algorithm>
4   using namespace std;
5
6   bool czyPalindrom(string s) {
7     string s2=s;
8     reverse(s2.begin(),s2.end());
9     return s==s2;
10  }
```

4. Wczytaj od użytkownika tekst. Policz ile razy występują wczytane znaki. Wynik dla każdego występującego znaku wyświetl w postaci *znak(kod ASCII):ilość wystąpień*.

```cpp
#include <iostream>
#include <string>
using namespace std;
int znaki[256]; //zmienna globalna

int main()
{
  string s;
  cout<<"Wpisz tekst: ";
  getline(cin,s);
  for(int i=0;i<s.size();i++)
    znaki[(int)s[i]]++;
  for(int i=0;i<256;i++)
    if(znaki[i]!=0) {
      cout<<(char)i<<"("<<i<<"):"<<znaki[i]<<endl;
    }
  return 0;
}
```

```
Wpisz tekst: Ala ma kota, a kot ma wszystko co chce.
 (32):8
,(44):1
.(46):1
A(65):1
a(97):5
c(99):3
e(101):1
h(104):1
k(107):3
l(108):1
m(109):2
o(111):4
s(115):2
t(116):3
w(119):1
y(121):1
z(122):1
```

5. Napisać funkcję o nagłówku **string wymieszajZnaki(string s)**, która zwróci kopię łańcucha znaków będącego jej parametrem i zmieni w niej kolejność znaków na losową (pseudolosową). Przetestuj 10 razy działanie tej funkcji dla słowa "algorytm".

```cpp
#include <iostream>
#include <string>
#include <cstdlib>
#include <ctime>
#include <algorithm>
using namespace std;

string wymieszajZnaki(string s) {
  string ss=s;
  int ile=s.size()*(rand()%5+2);
  for(int i=0;i<ile;i++)
    swap(ss[rand()%s.size()],ss[rand()%s.size()]);
  return ss;
}

int main()
{
  srand(time(NULL));
  for(int i=0;i<10;i++)
    cout<<"algorytm -> "<<wymieszajZnaki("algorytm")<<endl;
  return 0;
}
```

```
algorytm -> agylrtmo
algorytm -> ymrlgtao
algorytm -> rgltmaoy
algorytm -> togyraml
algorytm -> otlmgyra
algorytm -> gmtoalyr
algorytm -> aogtlymr
algorytm -> oatmrlgy
algorytm -> lotgryam
algorytm -> galorytm
```

6. Napisać funkcję o nagłówku **string suma(string a, string b)** umożliwiającą sumowanie dużych liczb naturalnych reprezentowanych jako łańcuchy znaków. Przetestuj działanie utworzonej funkcji.

```
#include <iostream>
#include <string>
using namespace std;

//funkcja sumuje duze liczby naturalne zapisane jako stringi
string suma(string a, string b)
{
  int pa=a.length()-1; //pozycja ostatniej cyfry w lancuchu a
  int pb=b.length()-1; //pozycja ostatniej cyfry w lancuchu b
  string wynik="";     //miejsce na obliczona sume liczb
  int p=0;             //wartosc przeniesienia

  while(pa>=0 || pb>=0) //dodajemy kolejne cyfry
  {
    int ca; //cyfra liczby a
    int cb; //cyfra liczby b
    int w;  //suma cyfr
    if(pa>=0) ca=a[pa]-48; else ca=0; //odczytujemy ca
    if(pb>=0) cb=b[pb]-48; else cb=0; //odczytujemy cb
    w=ca+cb+p; //obliczamy sume cyfr i uwzgledniamy przeniesienie
    wynik=char(w%10+48)+wynik; //wstawiamy ostatnia cyfre do lancucha wynikowego
    p=w/10; //obliczamy przeniesienie do nastepnej kolumny
    pa--;   //aktualizacja indeksu w lancuchu a
    pb--;   //aktualizacja indeksu w lancuchu b
  }
  if(p) wynik="1"+wynik; //dopisujemy "1" jesli jest przeniesienie
  return wynik;
}

int main()
{
  string a, b;
  cout<<"Dodawanie duzych liczb naturalnych."<<endl<<endl;
  cout<<"Podaj liczbe a = ";
  cin>>a;
  cout<<"Podaj liczbe b= ";
  cin>>b;
  cout<<"\na+b = "<<suma(a,b)<<endl;
  return 0;
}
```

```
Dodawanie duzych liczb naturalnych.

Podaj liczbe a = 1111111111111111111111111111111111111111111111111111111111111111111111
1111111111111111111111111111111111111111111111111111111111111111
Podaj liczbe b= 2222222222222222222222222222222222222222222222222222222222222222222222
2222222222222222222222222222222222222222222222222222222222222222222222
2222222222222299999

a+b = 2222222222222222222222222222222222222223333333333333333333333333333333333
3333333333333333333333333333333333333333333333333333333333333333333333333
3333411110
```

Zadania do samodzielnej realizacji:

1. (1p) Napisać funkcję o nagłówku **int ileSamoglosek(string s)** zliczającą ilość samogłosek występujących w argumencie jej wywołania. Uwaga: zarówno „a” i „A” to ta sama samogłoska. Przetestuj działanie utworzonej funkcji.

2. (2p) Napisać funkcję o nagłówku **string zamienWielkoscLiter(string s)**, która zamieni małe litery na duże i odwrotnie w łańcuchu znaków podanym jako jej parametr. Wynikiem ma być łańcuch znaków zawierający kopię łańcucha po zmianie wielkości liter. Przetestuj działanie utworzonej funkcji.

3. (2p) Napisać funkcję o nagłówku **bool czyAnagram(string s1, string s2)**, która sprawdza, czy łańcuch znaków s2 jest anagramem łańcucha s1, czyli czy składa się z tych samych znaków, ale ustawionych niekoniecznie w tej samej kolejności.
   Uwaga: Należy sprawdzać jedynie małe i duże litery alfabetu angielskiego, jednak bez względu na ich wielkość, tzn. zarówno małe „a” jak i duże „A” liczone są tak samo. Pozostałe znaki powinny być ignorowane.