Programowanie w języku C++ (EAR1S03006)

Politechnika Białostocka - Wydział Elektryczny

Automatyka i Robotyka, semestr III, studia stacjonarne I stopnia

Rok akademicki 2021/2022

Zajęcia nr 10 (15.12.2021)

dr inż. Jarosław Forenc

STL (ang. Standard Template Library)

- standardowa biblioteka szablonów (wzorców)
- opracowana w 1990 roku (Alex Stepanow, Meng Lee)
- w 1994 roku przyjęta jako standard - ANSI/ISO Standard C++
- zawiera szablony:
  - kontenerów (pojemników)
  - iteratorów
  - algorytmów
- kontener
  - jednostka umożliwiająca przechowywanie wielu wartości tego samego typu (kontenery w STL są homogeniczne - jednorodne)
  - przykłady: vector, deque, list, set, multiset, map, multimap
  - wybór kontenera zależy od problemu, który rozwiązujemy

STL (ang. Standard Template Library)

- iterator
  - obiekt pozwalający przemieszczać się po elementach kontenera (przypomina wskaźnik używany do odwoływania się do elementów tablicy)
  - do iteratorów można stosować m.in. operatory: * (wyłuskania), ++, ==, !=
- algorytmy
  - grupa instrukcji opisujących sposób wykonywania konkretnego zadania, np. sortowanie, wyszukiwanie
  - szablony funkcji globalnych działające dla każdego kontenera (np. sort, search, count, replace)
  - algorytmy działają z iteratorami (ale także ze zwykłymi tablicami)

STL (ang. Standard Template Library)

- deque (kolejka, double ended queue)
  - kontener sekwencyjny - pozycja elementu w kontenerze zależy od czasu i miejsca, gdzie został wstawiony, a nie od jego wartości
  - dynamiczna tablica dwukierunkowa (kolejka o dwóch końcach)
  - umożliwia szybkie dodawanie i usuwanie elementów na obu końcach
- list (lista)
  - kontener sekwencyjny, lista dwukierunkowa
  - brak indeksowania elementów - elementy można dodawać i usuwać wszędzie (na początku, na końcu, w środku)
- set (zbiór), multiset (multi-zbiór)
  - kontener asocjacyjny - aktualna pozycja elementu w pojemniku zależy od jego wartości (automatycznie sortuje wartości)
  - implementowany w postaci zbalansowanego drzewa binarnego (BST)
  - każdy element może wystąpić tylko raz (set) lub wiele razy (multiset)

STL (ang. Standard Template Library)

- map (mapa, słownik), multimap (multi-mapa)
  - zawiera pary elementów: klucz / wartość
  - kontener asocjacyjny (automatycznie sortuje elementy względem klucza)
  - implementowany w postaci zbalansowanego drzewa binarnego (BST)
  - każdy klucz może wystąpić tylko raz (map) lub wiele razy (multimap)

STL - kontener vector

- dynamiczna tablica (wektor) mogąca zawierać elementy dowolnego typu
- wymaga dołączenia pliku nagłówkowego: #include <vector>
- można stworzyć pusty wektor i rozszerzyć go (zmniejszyć) na bieżąco

```
vector<int> vec;
```

- można określić rozmiar wektora przy deklaracji

```
vector<double> vec(10);
```

- można określić rozmiar wektora przy deklaracji i zainicjalizować go 0

```
vector<double> vec(10,0.);
```

- vector działa najszybciej, gdy z góry zarezerwujemy obszar pamięci - metoda reserve()

STL - metody kontenera vector

- push_back() - dodanie elementu na końcu
- pop_back() - usunięcie elementu z końca
- size() - zwraca liczbę elementów
- at(i) - zwraca element o indeksie i (sprawdza poprawność indeksu i)
- [i] - zwraca element o indeksie i (nie sprawdza poprawności indeksu i)
- clear() - usuwa wszystkie elementy
- max_size() - zwraca maksymalny rozmiar kontenera
- empty() - sprawdza czy kontener jest pusty
- front() - zwraca pierwszy element wektora
- back() - zwraca ostatni element wektora

Klasa std::string

- Klasa przeznaczona do wykonywania operacji na tekstach
- Informacje:
  - J. Grębosz: „Symfonia C++ standard”. Edition 2000, 2005. Rozdział 11, str. 503-614
  - http://www.cplusplus.com/reference/string/string/ (ang.)
- Dołączenie pliku nagłówkowego (bez .h na końcu)

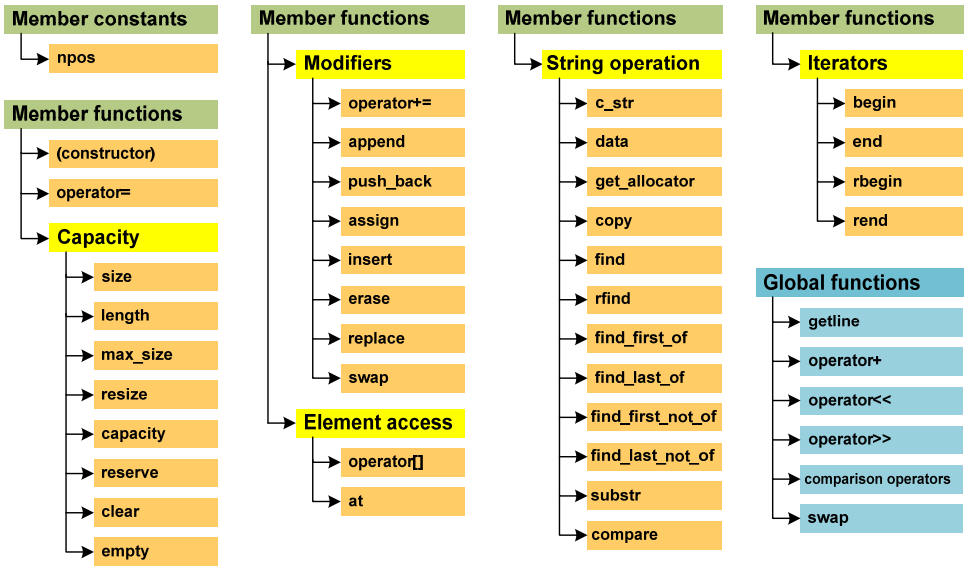
```
#include <string>
```

Klasa std::string

- Klasa jest częścią biblioteki standardowej, jej nazwa jest zadeklarowana w przestrzeni nazw std
- Dodając using namespace std; nie musimy poprzedzać identyfikatorów kwalifikatorem zakresu std::

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


Klasa std::string



Klasa std::string - definiowanie obiektów

- Standardowa definicja obiektu:

```
nazwa_klasy nazwa_obiektu(argumenty konstruktora);
```

Przykłady:

```
string s1;                  // pusty string
string s2("Tekst");         // inicjalizacja C-stringiem
string s3("Tekst",3);       // inicjalizacja fragmentem C-stringu
string s4(10,'@');          // inicjalizacja 10 znakami '@'
cout << s1 << " " << s2 << " ";
cout << s3 << " " << s4 << endl;
```

```
Tekst Tek @@@@@@@@@@
```


Klasa std::string - definiowanie obiektów

- Inne przykłady:

```
char tab[10] = "Tekst";
string s1(tab);             // inicjalizacja tablicą znaków
string s2(s1);              // inicjalizacja innym obiektem klasy string
string s3(s1,2,4);          // inicjalizacja fragmentem obiektu s1
string s4(tab,1,3);         // inicjalizacja fragmentem tablicy tab
string s5 = "Tekst";        // inicjalizacja C-stringiem
cout << s1 << " " << s2 << " " << s3 << " ";
cout << s4 << " " << s5 << endl;
```

```
Tekst Tekst kst eks Tekst
```

Klasa std::string - funkcje składowe

size()	size_t size();

length()	size_t length();

- Zwraca liczbę znaków w obiekcie

```
string str("12345678");
size_t d1 = str.size();
size_t d2 = str.length();
cout << "Dlugosc = " << d1 << " " << d2 << endl;
```

```
Dlugosc = 8 8
```


Klasa std::string - funkcje składowe

max_size()	size_t max_size();

- Zwraca maksymalną liczbę znaków, które mogą być przechowywane w obiekcie

```
string str;
cout << str.max_size() << endl;
```

```
4294967294
```


Klasa std::string - funkcje składowe

resize()	void resize(size_t n, char c = '\0');

- Zmienia długość tekstu przechowywanego w obiekcie
  - jeśli n < size(), to pozostawia tylko n pierwszych znaków
  - jeśli n > size(), to zapisuje c jako dodatkowe znaki

```
string s("123456789");
cout << s << endl; s.resize(5);
cout << s << endl; s.resize(15,'*');
cout << s << endl;
```

```
123456789
12345
12345**********
```


Klasa std::string - funkcje składowe

capacity()	size_t capacity();

- Zwraca bieżący rozmiar zarezerwowanej pamięci do przechowywania tekstu w obiekcie

```
string str("12345678");
cout << str.capacity() << endl;
```

```
15
```

Klasa std::string - funkcje składowe

reserve()	void reserve(size_t res_arg=0);

- Żąda zmiany rozmiaru zarezerwowanej pamięci do przechowywania tekstu w obiekcie
- Argument res_arg określa minimalny rozmiar, który będzie zarezerwowany (może on być większy)

```
string str;

str.reserve(100);
cout << str.capacity() << endl;
```

```
111
```


Klasa std::string - funkcje składowe

empty()	bool empty();

- Zwraca wartość true jeśli obiekt jest pusty

clear()	void clear();

- Usuwa tekst przechowywany w obiekcie

```
string str("12345678");

if (!str.empty()) str.clear();
cout << str.size() << "->" << str << "<-" << endl;
```

```
0-><-
```


Klasa std::string - funkcje składowe

at()	char & at(size_t pos);
operator[]	char & operator [](size_t pos);

- Umożliwia odwoływanie się do wybranych znaków w tekście, [] - bez kontroli zakresu, at() - z kontrolą zakresu (błąd powoduje rzucenie wyjątku out_of_range)

```
string str("Napis testowy");

for (int i=0; i<str.size(); i++) cout << str.at(i);
for (int i=0; i<str.size(); i++) cout << str[i];
```

```
Napis testowyNapis testowy
```


Klasa std::string - funkcje składowe

operator +=	string & operator +=(const string &str);
	string & operator +=(const char *s);
	string & operator +=(char c);

- Dodaje na końcu istniejącego tekstu dodatkowy tekst z obiektu str, tekst z tablicy s (do znaku \0) lub znak c

```
string str("Dane: "), imie("Jan ");

str += imie;
str += "Kowalski";
str += '\n';
cout << str;
```

```
Dane: Jan Kowalski
```

Klasa std::string - funkcje składowe

append()	string & append(const string & str); string & append(const char * s);

- Dodaje na końcu istniejącego tekstu dodatkowy tekst z obiektu str lub tekst znajdujący się w tablicy s (do znaku \0)

append()	string & append(const string & str, size_t pos, size_t n);

- Dodaje na końcu istniejącego tekstu n znaków z obiektu str począwszy od pozycji pos

Klasa std::string - funkcje składowe

append()	string & append(const char* s, size_t n);

- Dodaje na końcu istniejącego tekstu n pierwszych znaków z tablicy s

append()	string & append(size_t n, char c);

- Dodaje na końcu istniejącego tekstu n takich samych znaków c

Klasa std::string - funkcje składowe

- Przykład (append):

```
string s1("abc"), s2("123");
char s[10]="ABCDEF";

cout << s1 << endl;
s1.append(s2);     cout << s1 << endl;
s1.append(s,4);    cout << s1 << endl;
s1.append(3,'@'); cout << s1 << endl;
```

```
abc
abc123
abc123ABCD
abc123ABCD@@@
```


Klasa std::string - funkcje składowe

assign()	string & assign(const string & str);
	string & assign(const char * s);
	string & assign(const string & str,
	size_t pos, size_t n);
	string & assign(const char * s, size_t n);
	string & assign(size_t n, char c);

- Zastępuje istniejący tekst nowym tekstem
- Znaczenie parametrów jest takie samo jak dla append()

Klasa std::string - funkcje składowe

- Przykład (assign):

```
string s1("abc"), s2("123");
char s[10]="ABCDEF";

cout << s1 << endl;
s1.assign(s2);     cout << s1 << endl;
s1.assign(s,4);    cout << s1 << endl;
s1.assign(3,'@'); cout << s1 << endl;
```

```
abc
123
ABCD
@@@
```


Klasa std::string - funkcje składowe

insert()	string & insert(size_t pos1, const string & str);

- Z obiektu str kopiuje cały tekst i wstawia począwszy od miejsca określonego przez pos1

insert()	string & insert(size_t pos1, const string & str, size_t pos2, size_t n);

- Z obiektu str kopiuje n znaków począwszy od miejsca określonego przez pos2 i wstawia począwszy od miejsca określonego przez pos1

Klasa std::string - funkcje składowe

- Przykład (insert):

```
string str1("ABCDEFGHIJ"), str2("123456789");
string str3("ABCDEFGHIJ");

cout << str1 << endl;
str1.insert(4,str2);  cout << str1 << endl;

cout << str3 << endl;
str3.insert(4,str2,3,4);  cout << str3 << endl;
```

```
ABCDEFGHIJ
ABCD123456789EFGHIJ
ABCDEFGHIJ
ABCD4567EFGHIJ
```


Klasa std::string - funkcje składowe

insert()	string & insert(size_t pos1, const char *s, size_t n);

- Z tablicy znaków s kopiuje n pierwszych znaków i wstawia począwszy od miejsca określonego przez pos1

insert()	string & insert(size_t pos1, const char *s);

- Kopiuje wszystkie znaki z tablicy s i wstawia począwszy od miejsca określonego przez pos1

Klasa std::string - funkcje składowe

insert()	string & insert(size_t pos1, size_t n, char c);

- Wstawia n znaków c począwszy od miejsca określonego przez pos1

Klasa std::string - funkcje składowe

erase()	string & erase(size_t pos=0, size_t n=npos);

- Kasuje n znaków począwszy od pozycji pos
- npos - oznacza wszystkie znaki do końca

```
string str("123456789");

str.erase(3,2); cout << str << endl;
str.erase(4);   cout << str << endl;
str.erase();    cout << str << endl;
```

```
1236789
1236
```


Klasa std::string - funkcje składowe

swap()	void swap(string & str);

- Wymienia tekst z obiektem str

```
string s1("12345"), s2("ABC");

cout << "s1: " << s1 << " s2: " << s2 << endl;
s1.swap(s2);
cout << "s1: " << s1 << " s2: " << s2 << endl;
```

```
s1: 12345 s2: ABC
s1: ABC s2: 12345
```


Klasa std::string - funkcje składowe

c_str()	const char* c_str() const;

- Tworzy tablicę znaków (C-string) zakończoną znakiem \0 zawierającą tekst przechowywany w obiekcie i zwraca wskaźnik do tej tablicy

```
char *s;
string str("ABC 123");

s = new char[str.size()+1];
strcpy(s,str.c_str());
printf("%s\n",s);
```

```
ABC 123
```

Klasa std::string - funkcje składowe

data()	const char* data() const;

- Zwraca wskaźnik do tablicy znaków zawierającej tekst przechowywany w obiekcie, ale bez \0 na końcu

copy()	size_t copy(char* s, size_t n, size_t pos = 0) const;

- Z tekstu przechowywanego w obiekcie kopiuje n znaków począwszy od pozycji pos zapisując znaki w tablicy s
- Nie dodaje na końcu tablicy s znaku \0
- Zwraca liczbę faktycznie przekopiowanych znaków

Klasa std::string - funkcje składowe

find()	size_t find(const string& str, size_t pos = 0);
	size_t find(const char* s, size_t pos, size_t n);
	size_t find(const char* s, size_t pos = 0);
	size_t find(char c, size_t pos = 0);

- Szuka w stringu zawartości innego stringu (str), tablicy znaków (s) lub pojedynczego znaku (c)
- Zwraca pozycję pierwszego znaku zawierającego poszukiwaną zawartość
- pos określa pozycję, od której rozpoczyna się poszukiwanie, zaś n - liczbę poszukiwanych znaków
- Zwraca npos, gdy zawartość nie została znaleziona

Klasa std::string - funkcje składowe

substr()	string substr(size_t pos=0, size_t n=npos);

- Zwraca obiekt klasy string zawierający n znaków tekstu począwszy od pozycji pos

```
string s1("123456789"), s2;

s2 = s1.substr(5,3);
cout << "s2: " << s2 << endl;
```

```
s2: 678
```


Klasa std::string - funkcje globalne

getline()	istream& getline(istream& is, string& str, char delim); istream& getline(istream& is, string& str)

- Czyta znaki ze strumienia is do napotkania znaku delim lub \n, a następnie zapisuje je w obiekcie str

```
string str;

cout << "podaj tekst: ";
getline(cin,str);
cout  << "wczytany tekst: " << str << endl;
```

Klasa std::string - funkcje globalne

operator+	string + string char* + string string + char * char + string string + char

- Zwraca string zawierający tekst będący połączeniem dodawanych argumentów

```
string str, str1("ABC"), str2("123");

str = str1 + str2;   cout << str << endl;
str = str2 + "DEF";  cout << str << endl;
```

```
ABC123
123DEF
```


Klasa std::string - funkcje globalne

operatory relacyjne	string != string string == string string < string string <= string string > string string >= string

- Porównuje dwa stringi zwracając 1 jeśli warunek jest prawdziwy i 0 jeśli warunek nie jest prawdziwy