Laboratorium Informatyki II
Ćwiczenie 5

1 Klasa *string*

WW języku C ciągi tekstowe przetwarza się za pomocą tablicy znakowej (`char`) i narzędzi z biblioteki `<string.h>`. Nie jest to rozwiązanie optymalne głównie ze względu na operowanie na tablicy i konieczność zaprogramowanie w kodzie programu zarządzania rozmiarem tablicy.

W języku C++ także można stosować tablicę `char`, ale zdecydowanie wygodniejszym rozwiązanie jest zastosowanie klasy `string` i funkcji i metod dostępnych w bibliotece `<string>`. Klasa dostarcza kilka konstruktorów i szereg metod dedykowanych:

- przypisywania ciągów,
- łączenia,
- porównywania,
- przeglądania
- wyłuskiwania.

Standardowy konstruktor klasy string tworzy obiekt pusty, pozostałe konstruktory pozwalają na inicjowanie obiektu ciągiem tekstowym tworzonym na bazie innego obiektu string, tablicy znakowej lub znaków. Poniżej pokazano kilka wybranych metod inicjacji obiektu klasy `string`.

```
char ciag[] = "Ciąg tekstowy";
string tekst1;
string tekst2 = "Ciąg tekstowy";
string tekst3(ciag,4);
string tekst4(10,'$');
string tekst5(tekst2,5,8);
```

Przypisywanie danych do obiektu string pobieranych z klawiatury zrealizować można bezpośrednio:

```
string tekst;
cinn >> tekst;
getline(cin,tekst);
getline(cin,tekst,';');
```

lub pośrednio z wykorzystanie tablicy `char` lub innego obiektu `string` i metod klasy `string`:

```
char bufor[50];
cin.getline(bufor,50);
string tekst;
tekst.append(bufor);
```

Łączenie ciągów tekstowych realizowane jest z wykorzystaniem klasy `string` może być realizowane za pomocą operatorów (`+` lub `+=`) i metod (`.append()`, `.insert()`, `.push_back()`). Określanie rozmiaru ciągu tekstowego w obiekcie klasy `string` realizowane jest za pomocą jednej z dwóch metod `.length()` lub `.size()`. Do głównych metod klasy należy jednak zaliczyć metody umożliwiające wyszukiwanie znaków lub ciągów znaków:

- `find()` - wyszukiwanie od początku
- `rfind()` - wyszukiwanie od końca
- `find_first_of()` - wyszukuje pierwszego wystąpienia dowolnego znaku z zestawu poszukując od początku ciągu
- `find_last_of()` - to samo co powyżej, ale wyszukuje od końca ciągu
- `find_first_not_of()` - wyszukuje pierwszego znaku który nie występuje w zestawie poszukując od początku ciągu
- `find_last_not_of()`- to samo co powyżej, ale wyszukuje od końca ciągu

2 Biblioteka STL

Biblioteka **STL** jest zestawem szablonów na które składają się *kontenery*, *iteratory* oraz *algoytmy*. Zostały one pierwotnie opracowane i były stosowane w HP lab. Z czasem zostały włączone do standardu języka C++. Biblioteka dostarcza narzędzi do tworzenia i przetwarzania danych.

2.1 Kontenery

Kontener jest "pojemnikiem" na dane z określonym sposobem przechowywania i dostępu. W standardzie dostępne są kontenery:

- Lista `<list>`
- Tablica `<vector>`
- Tablica podwójnie kończona `<deque>`
- Tablica jednowymiarowa `<array>`
- Tablica bitowa `<bitset>`
- Drzewo poszukiwań (zbiór) `<set>`
- Wielokrotne drzewo poszukiwań `<multiset>`
- Mapa poszukiwań `<map>`
- Wielokrotna mapa poszukiwań `<multimap>`

- Stos `<stack>`
- Kolejka `<queue>`

Każdy z powyższych kontenerów pozwala na przechowywanie dowolnego typu danych. W wielu przypadkach zastosowanie odpowiedniego typu kontenera dla programowanego algorytmu zapewnia wygodniejszy a co najważniejsze wydajniejszy dostęp do danych niż standardowa tablica. Operacje na poszczególnych typach kontenerów realizowane są z wykorzystaniem dostarczonych wraz z biblioteką **STL** algorytmów. Nawigowanie po elementach składowych kontenera realizowane jest za pośrednictwem *iteratora*.

2.2 Iteratory

Iterator jest swego rodzaju wskaźnikiem, który nie odnosi się jednak do określonej komórki pamięci, a do określonego elementu kontenera. Z jago pośrednictwem można zapisywać i odczytywać dane z kontenera.

2.3 Algorytmy

Algorytmy są zestawem uniwersalnych funkcji dostarczonych przez bibliotekę `algorithm` ułatwiających przetwarzanie danych zgromadzonych w kontenerach.

1. `binary_search`: Wykonuje algorytm binarnego przeszukiwania na posortowanym zakresie danych.
2. `copy`: Kopiuje elementy.
3. `copy_backward`: Kopiuje elementy od końca.
4. `count`: Zlicza wystąpienia elementu.
5. `count_if`: Zlicza wystąpienia elementów spełniających warunek.
6. `equal`: Sprawdza, czy elementy w dwóch zakresach są równe.
7. `fill`: Wypełnia zakres wartością.
8. `fill_n`: Wypełnia sekwencję wartością.
9. `find`: Szuka wystąpienia elementu.
10. `find_end`: Szuka ostatniego wystąpienia drugiego zakresu w pierwszym zakresie.
11. `find_if`: Szuka elementu spełniającego warunek.
12. `for_each`: Wywołuje funkcję dla każdego elementu.
13. `inplace_merge`: Scala dwa posortowane zakresy w jeden posortowany zakres.
14. `iter_swap`: Zamienia wartości obiektów wskazywanych przez iteratory.
15. `sort`: Sortuje elementy w podanym zakresie.
16. `swap`: Zamienia wartości dwóch obiektów
17. `max`: Zwraca większą wartość.

18. `max_element`: Szuka największej wartości w zakresie.
19. `min`: Zwraca mniejszą wartość.
20. `min_element`: Szuka najmniejszej wartości w zakresie.
21. `replace`: Zamienia wszystkie wystąpienia wartości.
22. `replace_if`: Zamienia wszystkie wartości spełniające warunek.
23. `reverse`: Odwraca kolejność elementów w podanym zakresie.
24. `reverse_copy`: Kopiuje elementy z określonego zakresu w odwróconej kolejności.
25. `search`: Szuka wystąpienia drugiego zakresu w pierwszym zakresie.

Zadania

W oparciu o wybrany kontener z biblioteki STL napisać program Zbierający dane kontaktowe. Obsługę i przetwarzanie danych tekstowych oprzeć na klasie string i bibliotece ¡string¿. Program zbiera dane:

a) imiona (do trzech)

b) nazwisko

c) adres pocztowy

d) email (do trzech)

e) nr telefonu (dowolna liczba, różne kategorie)

f) uwagi (możliwość zapisania dowolnej długości ciągu tekstowego)

Program umożliwia:

- dodawanie nowych kontaktów
- kasowanie kontaktu
- modyfikowanie kontaktu
- wyszukiwanie kontaktów po wybranym parametrze
- wyświetlanie listy wizytówek posortowanych zgodnie z wybranym parametrem