Języki Programowania, więcej o klasach szablonowych w oddzielnym pliku

Stos jest to struktura danych, która umożliwia wykonywanie dwóch operacji: włożenie elementu na stos oraz zdjęcie elementu ze stosu. Elementy są zdejmowane w kolejności odwrotnej do wkładania: wykorzystywana jest kolejka LIFO (*Last In First Out* - ostatni wchodzi pierwszy wychodzi).
Należy napisać klasę **szablonową** `Stack` która implementuje strukturę stosu dla dowolnego typu elementów.

Klasa powinna wspierać następujące funkcje składowe:
`Push` – dodanie elementu na stos
`Pop` – zdjęcie elementu ze stosu
`Top` – zwracającej element z wierzchu stosu
Oraz dwa konstruktory: domyślny oraz z jednym parametrem (przyjmujący pierwszy element na stosie).

Dodatkowo należy napisać funkcje która wypisuje wszystkie elementy odłożone na stosie.

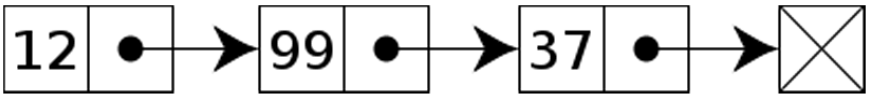
W przypadku klas szablonowych wszystkie funkcje składowe powinny znajdować się w pliku .h!

Zaimplementowany stos należy przetestować (wszystkie metody!) dla:
- liczb całkowitych: `3 6 1 8`
- znaków: `a, y, w`
- ciągów znaków (string): `ala, kot, owoc`

Następnie należy użyć struktury `stack` zaimplementowanej w bibliotece STL (`#include <stack>`), wykonując na niej te same operacje. Polecam dokumentację *C++ Reference* (google).

Podobnie należy użyć struktury `stack` z biblioteki STL (dla liczb całkowitych: `3 6 1 8`). Wszystkie elementy stosu należy wypisać.

1. Stworzyć klasę **Stack** (nie należy używać nazwy “stack” z małej litery!) z klasą wewnętrzną **Element**.
Klasa **Element** powinna mieć dwa publiczne elementy:
```
int value
Element* next
```
Klasa **Stack** powinna mieć jeden prywatny składnik:
`Element* top` – wkaźnik na szczytowy element stosu
oraz metody:
`Push(int val)` – dodającą jedną liczbę na stos,
`Pop()` – zwracającą ostatnią wartość włożoną na stos i kasującą równocześnie ten element ze stosu,
`Top()` – zwracającą ostatnią wartość włożoną na stos (jedynie odczytującą), oraz
`Print()` – wypisującą stos na ekran,
oraz destruktor.
Struktura stosu jest bardzo podobna do struktury listy jednokierunkowej:



Budowa listy jednokierunkowej (lub stosu) jest następująca: każdy jej element (rekord) posiada swoje **dane** oraz **1 wskaźnik** – do elementu następnego (lub poprzedniego).

W przypadku stosu mamy wyróżniony jeden element: ten, znajdujący się na szczycie stosu (ten, który został dodany jako ostatni). Nasza klasa przechowuje wskaźnik do tego elementu (`*top`). Możemy go odczytać (używając metody `Top()`) bądź zdjąć ze stosu (skasować) używając metody `Pop`.

Dodawanie nowego elementu (metoda `Push`) powinno odbywać się na zasadzie:

a. stworzenie nowego elementu
b. ustawienie odpowiednich wartości dla tego elementu: `value`, oraz dla wskaźnika `next` (będzie dodany na szczyt stosu!)
c. zmiana wartości wskaźnika `top`.

- Przetestować w funkcji main:

```
Stack stos;
stos.Push(3);
stos.Push(6);
stos.Print();
```

2. Zamienić klasę Stack na klasę szablonową.
- Przetestować w funkcji main dla liczb całkowitych (dodać `3, 6, 1, 8`, wypisać, usunąć ostatnią, wypisać).
- Przetestować dla zmiennych typu char (dodać `a, y, w`, wypisać, wypisać ostatnią).
- Przetestować dla zmiennych typu string (dodać `ala, kot, owoc`, wypisać, wypisać ostatnią).

3. Stworzyć stos z biblioteki STL (#include <stack>), dodać trzy zmienne typu double: 3.14, 2.44, 8.3. Wypisać tą umieszczoną na wierzchu, usunąć ją ze stosu, wypisać następną, ponownie usunąć I wypisać ostatnią.

4. Stworzyć listę z biblioteki STL (#include <list>)
Dodać `3, 6, 1, 8`, wypisać używając iteratora.

```
list<int>::iterator it;
  for ( it=mylist.begin() ; it != mylist.end(); it++ )
    cout << " " << *it;
```

`Dodatkowe:`
- Stworzyć na podstawie klasy Stos klasę Lista.