laboratorium: zadanie 4 termin: 30 marca – 2 kwietnia 2020 r.

# KURS JĘZYKA C++

## STOS ZBUDOWANY NA TABLICY

Instytut Informatyki Uniwersytetu Wrocławskiego *Paweł Rzechonek*

Stos to bufor na dane, do którego można wkładać nowe elementy oraz wyciągać elementy w pewnej ustalonej kolejności. Charakterystyczną cechą stosu jest to, że w danym momencie można z niego wyciągnąć tylko ten element, który został do niego najpóźniej włożony.

Odpowiednikiem tej strukruty w codziennym życiu może być na przykład stos książek na biurku — książka, która została odłożona na stos jako ostatnia jako pierwsza zostanie z niego ściągnięta. Stąd się wzięła nazwa takiego bufora: LIFO (ang. *Last In, First Out*).

**Zadanie.**

Zdefiniuj klasę `stos`, która będzie strukturą typu LIFO — element, który został do tej struktury dodany najpóźniej, będzie z niej wyciągnięty najszybciej. Struktura ta ma służyć do przechowywania napisów typu `string`.

Sama funkcjonalność stosu ma być bardzo prosta: wkładamy napis na stos (metoda `void wloz(string)`), ściągamy napis ze stosu (metoda `string sciagnij()`), sprawdzamy jaki napis leży na wierzchu (metoda `string sprawdz()`) oraz pytamy o liczbę wszystkich elementów na stosie (metoda `int rozmiar()`).

Pojemność stosu ma być ustalona w konstruktorze — zdefiniuj więc prywatne pole `pojemnosc` typu `int`, w którym będzie pamiętany maksymalny rozmiar stosu. Będziesz też potrzebować informacji o liczbie elementów aktualnie włożonych na stos — zdefiniuj zatem prywatne pole `ile` typu `int`, w którym będziesz pamiętać liczbę elementów przechowywanych na stosie. Sam stos zaimplementuj na tablicy utworzonej dynamicznie na stercie (za pomocą operatora `new[]`). W klasie `stos` zdefiniuj więc prywatny wskaźnik na tablicę `tablica` typu `string*`. W destruktorze należy zwolnić pamięć przydzieloną dla tej tablicy (za pomocą operatora `delete[]`).

Stos ma posiadać pięć konstruktorów: konstruktor z zadaną pojemnością (pojemność ma być liczbą dodatnią nie większą od 100), konstruktor bezparametrowy i jednocześnie delegatowy (domyślna pojemność stosu niech wynosi 1), konstruktor inicjalizyjący stos za pomocą listy napisów (za pomocą niepustej kolekcji `initializer_list<string>`), konstruktor kopiujący i konstruktor przenoszący. Aby uzupełnić semantykę kopiowania i przenoszenia zdefiniuj opowiednie operatory przypisania — przypisanie kopijące i przypisanie przenoszące.

Definicję stosu uzupełnij o metodę tworzącą nowy stos z odwróconą kolejnością elementów w stosunku do stosu macierzystego. Metoda ta ma zwracać stos przez wartość:

```
stos stos::odwroc();
```

Napisz też interaktywny program testujący działanie stosu (interpretuj i wykonuj polecenia wydawane z klawiatury). Obiekt stosu, który będziesz testować stwórz na stercie operatorem `new` i nie zapomnij zlikwidować go operatorem `delete` przed zakończeniem programu! Aby lepiej poznać działanie kopiowania i przenoszenia możesz konstruktorach, destruktorze i operatorach przypisania wypisywać odpowiednie komunikaty na standardowe wyjście dla błędów.

**Elementy w programie, na które należy zwracać uwagę.**

- Podział programu na plik nagłówkowy (np. `stos.hpp`) z definicją kasy reprezentującej stos zbudowany na tablicy, plik źródłowy (np. `stos.cpp`) z definicją funkcji składowych dla stosu oraz plik źródłowy (np. `main.cpp`) z funkcją `main()`.
- Obiekt stosu ma być inicjalizowany na kilka różnych sposobów: konkretną pojemnością, domyślnie (konstruktor delegatowy), przez skopiowanie z innego stosu (konstruktor przenoszący), za pomocą listy wartości początkowych (lista wartości inicjalizujących zawartość stosu), za pomocą przeniesienia zawartości ze stosu tymczasowego (konstruktor przenoszący).
- Obiekt stosu ma być kopiowalny (przypisanie kopiujące i przenoszące).
- W funkcji `main()` należy zaprogramować interakcję programu z użytkownikiem — użytkownik wydaje polecenie dotyczące stosu, program je interpretuje i realizuje (wstawienie, uzunięcie, odpytanie o ilość elementów, odpytanie o pojemność stosu, wypisanie zawartości całego stosu).
- Czy programista potrafił zmusić program do uruchomienia konstruktora albo przypisania przenoszącego.