Zadanie 5 *4–7 kwietnia 2017 r.*

JĘZYK PROGRAMOWANIA C++

BUFORY ZAIMPLEMENTOWANE NA LISTACH

Instytut Informatyki Uniwersytetu Wrocławskiego *Paweł Rzechonek*

Bufor jest abstrakcyjną strukturą danych przeznaczoną do przechowywania pewnego dynamicznego zbioru danych. Do bufora możemy dodawać nowe dane, jak również usuwać dane, które się w nim znajdują. Z usuwaniem danych z bufora może być związana określona strategia. Kolejka (ang. *queue*) jest buforem typu FIFO (ang. *First In – First Out*) czyli takim, w którym wyciągany jest zawsze element najwcześniej włożony. Przeciwieństwem kolejki jest stos (ang. *stack*), czyli bufor typu LIFO (ang. *Last In – First Out*) a więc taki, w którym zawsze jest wyciągany element najwpóźniej włożony. Zarówno kolejka jak i stos są podstawowymi abstrakcyjnymi strukturami danych w informatyce i można je zaimplementować na wiele różnych sposobów.

Zadanie.

Zaimplementuj listę dwukierunkową przechowująca liczby rzeczywiste. Lista ma się składać z powiązanych ze sobą (za pomocą wskaźników) węzłów.

Zdefiniuj więc klasę `lista` reprezentującą listę dwukierunkową oraz zagnieżdżoną w liście w chronionej sekcji klasę *wezel* czyli węzeł listy przechowujący pojedynczą liczbę rzeczywistą typu `double`. Zakładamy, że elementy listy będą wirtualnie ponumerowane kolejnymi liczbami naturalnymi zaczynając od 0. Twoja lista powinna obsługiwać takie operacje jak wstawienie nowego elementu na zadaną pozycję do listy (pozycja 0 oznacza wstawienie na początek), usunięcie elementu z listy z zadanej pozycji, pobranie referencji do określonego elementu za pomoca operatora indeksowania oraz podanie rozmiaru listy. Obiekt listy ma być kopiowalny i przenaszalny.

Klasa `lista` ma być tylko opakowaniem dla homogenicznej struktury złożonej z węzłów (klasa `wezel`). Wszystkie operacje listowe wymienione wcześniej zaimplementuj w klasie `wezel` (możesz użyć rekurencji); klasa `lista` ma jedynie kontolować początek i koniec listy i zlecać wykonanie operacji listowych skrajnym węzłom listy.

W klasie węzła zdefiniuj operatory `new` i `delete`, które będą działały na pewnej globalnej tablicy (o określonnym z góry rozmiarze), przeznaczonej tylko na obiekty typu `wezel`. Zdefiniuj też jakąś prostą klasę do zarządzania takim obszarem pamięci — należy pamiętać, które sloty w tablicy przeznaczonej na węzły są wolne a które zajęte.

Następnie stosując niepubliczne dziedziczenie zdefiniuj klasę `kolejka` reprezentującą bufor FIFO. Struktura ta powinna zaimplementować operacje wstawiania nowych elementów i usuwania elementów zgodnie ze strategią kolejkową, wykorzystując listę. Wstawienie elementu do kolejki zrealizuj funkcją składową `wstaw` i równocześnie operatorem `+=`; usunięcie elementu z kolejki zrealizuj funkcją składową `usun` i równocześnie operatorem `--` w wersji prefiksowej; podgląd

elementu przeznaczonego do usunięcia zrealizuj funkcją składową `gotowy` i równocześnie operatorem `*`; zdefiniuj też funkcję składową `ile` do podania liczby wszytkich elementów w kolejce. Analogicznie do kolejki zdefiniuj stos jako klasę `stos` reprezentującą bufor LIFO z odmienną (stosową) strategią usuwania elementów.

Na koniec napisz program, który rzetelnie przetestuje działanie Twoich struktur.

Wskazówka.

W zagnieżdżonej klasie `wezel` zdefiniuj konstruktor domyślny. Pamiętaj, że klasa `wezel` ma posiadać w pełni zaprogramowaną funkcjonalność listy (w metodach klasy `lista` powinieneś tylko wywoływać odpowiednie metody na obiekcie pierwszego/ostatniego węzła w liście).

Listę zaprogramuj w taki sposób, aby w stałym czasie umożliwić dostęp do pierwszego i ostatniego elementu. Pamiętaj, aby w klasie `list` zdefiniować konstruktor domyślny, konstuktor kopiujący, konstuktor przenoszący, destruktor, przypisanie kopiujące, przypisanie przenoszące oraz zaprzyjaźniony operator pisania do strumienia (oraz ewentualnie czytania ze strumienia).

Uwaga.

Podziel program na pliki nagłówkowe i źródłowe.

Elementy w programie, na które należy zwrócić szczególną uwagę.

- Podział programu na pliki nagłówkowe i źródłowe.
- Zagnieżdżona definicja węzła.
- Niepubliczne dziedziczenie z listy.
- Obiekt listy ma implementować semantykę kopiowania i przenoszenia.
- Własne zarządzanie pamięcią za pomocą przeciążonych operatorów `new` i `delete`.
- Obiekt listy ma mieć operatorową funkcjonalność wstawiania, usuwania i podglądania skrajnego elementu.
- W funkcji `main()` należy przetestować wszystkie funkcje składowe i operatory listy.