Zestaw 1

Stos i kolejka

Termin 06.11.2020

Zadanie 1. Stos

> ***Stos*** *(ang.* stack*) to podstawowa struktura danych (abstrakcyjny typ danych), która implementuje zbiory dynamiczne. Elementy są usuwane ze stosu w kolejności od* najpóźniej *dodanego (strategia last-in, first-out -* ***LIFO****).*

Proszę najpierw przeczytać wskazówki dotyczące *języka C++*. W rozwiązaniach należy wykorzystać następujące elementy: szablony, uniwersalne referencje, doskonałe przekazywanie, semantyka przenoszenia.

Plik `stack.hxx`

Napisać szablon klas wg poniższego schematu, który implementuje stos przechowujący obiekty typu `T` w oparciu o tablicę o rozmiarze `N`. Operacje mają mieć złożoność $O(1)$.

```
template<class T, int N>
class Stack {
  template<class U>   // Uniwersalne referencje
  void push(U&& x);   // Wstawia element x na stos
  T pop();            // Usuwa element ze stosu i zwraca jego wartość
  T& top();           // Zwraca referencję do najmłodszego elementu
  int size();         // Zwraca liczbę elementów na stosie
  bool empty();       // Sprawdza czy stos jest pusty
};
```

W przypadku wystąpienia błędu **niedomiaru** lub **przepełnienia** operacje powinny wyrzucać wyjątek std::out_of_range. Należy również zaimplementować:

- Konstruktor domyślny, kopiujący i przenoszący
- Destruktor
- Przenoszący i kopiujący operator przypisania

> *Złożoność obliczeniowa programów powinna być* optymalna *dla danej implementacji. Dla klas szablonowych* deklaracje *i* definicje *muszą znajdować się w jednym pliku nagłówkowym.*

Plik `stack.cxx`

Program `stack.x` ma wczytać ze standardowego wejścia dane wygenerowane przez program opisany w zadaniu **Generator** i wypisać wynik działania odpowiednich operacji na stosie na standardowe wyjście. Format danych wejściowych oraz opis operacji znajduje się w zadaniu **Generator**. Stos przechowuje elementy typu `int`. Założyć, że na stosie może się naraz znajdować maksymalnie 10^6 elementów.

> ***Uwaga:*** *Programy* ***muszą*** *wczytywać dane wejściowe ze standardowego wejścia i wypisać rezultat na standardowe wyjście.*

Zadanie 2. Kolejka

> ***Kolejka*** *(ang.* queue*) to następna podstawowa struktura danych, która implementuje zbiory dynamiczne. Elementy są usuwane z kolejki w kolejności od* najwcześniej *dodanego (strategia first-in, first-out -* ***FIFO****).*

Plik queue.hxx

Napisać szablon klas `Queue`, który implementuje kolejkę w oparciu o bufor cykliczny i wg następującego schematu:

```
template<class T, int N>
class Queue {
  template<class U>
  void push(U&& x);      // Wstawia element x do kolejki (także enqueue)
  T pop();               // Usuwa element z kolejki (także dequeue) i zwraca jego wartość
  T& front();            // Zwraca referencję do najstarszego elementu (także peek)
  int size();            // Zwraca liczbę elementów w kolejce
  bool empty();          // Sprawdza czy kolejka jest pusta
};
```

Wszystkie powyższe operacje powinny działać w czasie $O(1)$.
Kolejka powinna móc przechowywać maksymalnie `N` obiektów typu `T`. W przypadku wystąpienia błędu **niedomiaru** lub **przepełnienia** operacje powinny wyrzucać wyjątek std::out_of_range.

Plik queue.cxx

Program `queue.x` ma wczytać ze standardowego wejścia dane wygenerowane przez program opisany w zadaniu **Generator** i wypisać wynik działania odpowiednich operacji na kolejce na standardowe wyjście. Format danych wejściowych oraz opis operacji znajduje się w zadaniu **Generator**. Założyć, że w kolejce może się naraz znajdować maksymalnie 10^6 elementów typu `int`.

Zadanie 3. Generator

Napisz program `generator.x`, który generuje dane wejściowe dla programów `stack.x` i `queue.x`. Dane powinny być generowane losowo (inne przy każdym uruchomieniu) i muszą być zgodne z poniższym formatem.

Format danych:

W pierwszej linii podana jest liczba $n \leq 10^6$ wskazującą na liczbę operacji do wykonania oraz n linii poleceń. Operacje mogą być następującego typu:

- `A x` - połóż na stos lub wstaw do kolejki liczbę $0 \leq x \leq 10^6$
- `D` - zdejmij/pobierz element z stosu/kolejki i go wypisz, jeśli stos/kolejka jest pusta wypisz “EMPTY”
- `S` - wypisz liczbę elementów na stosie lub w kolejce

Zadanie 4. Tester

Napisać narzędzia `test-stack` i `test-queue`, które testują poprawność działania programów - odpowiednio - `stack.x` i `queue.x`. W tym celu można wykorzystać kontenery `std::stack` i `std::queue` ze standardowej biblioteki *C++*. Mają państwo dużą swobodę w formie implementacji tego zadania (program / skrypt / Makefile / GUI / WebUI). Najlepsze rozwiązania będą premiowane dodatkowymi punktami. Tester powinien sprawdzać skrajne przypadki, mierzyć czas wykonania zadania i prezentować wyniki.

> *Chociaż STL (ang.* Standard Template Library*) miała istotny wpływ na rozwój standardowej biblioteki C++, obie biblioteki są od siebie zupełnie niezależne i nie należy ich utożsamiać.*

Pytania

- Opisz trzy sposoby obsługi cykliczności bufora.
- Omów przykłady zastosowania **stosu**?
- Omów przykłady zastosowania **kolejki**?
- Dlaczego operacja `pop()` z `std::stack` i `std::queue` **nie** zwraca wartości elementu?
- Dlaczego operacja `pop()` z `std::stack` i `std::queue` **nie** zwraca referencji do elementu?

Andrzej Görlich
a.goerlich@outlook.com