Laboratorium 4

Analiza właściwości struktur danych algorytmów zastosowanych do implementacji kolejki priorytetowej

1. Cel

Celem ćwiczenia jest praktyczne zapoznanie się z różnymi implementacjami interfejsu kolejki priorytetowej. W analityczno/doświadczalnej części ćwiczenia porównywane są parametry wydajnościowe algorytmów, zajętość pamięci, łatwość implementacji, a także przydatność danej pary algorytm–struktura danych dla różnych typów danych czy różnych typów sekwencji opracji *wstaw*/*wyjmij*. W praktycznej części ćwiczenia istnieje możliwość zaproponowania usprawnień wykorzystywanych algorytmów (np. poprawek adaptacyjnych) lub implementacji własnych struktur danych.

2. Struktura programu

Struktury danych i procedury interfejsu kolejki priorytetowej zaimplementowane zostały jako zestaw wzorców klas (w dużej części abstrakcyjnych) silnie powiązanych ze sobą hierarchią dziedziczenia (w tym dziedziczenia wielobazowego). Dla ułatwienia orientacji w trekście programu, mimo że jest on napisany w języku C++, do pisania kodu przyjęto podstawowe zasady obowiązujące w języku Java, tzn.:

- wszelkie nazwy są samokomentujące,
- nazwy klas zaczynają się z wielkiej litery,
- nazwy atrybutów i metod zaczynają się z małej litery,
- każda klasa umieszczona jest w pliku o nazwie takiej samej, jak jej nazwa.

Wszystkie pliki zawierające klasy (Uwaga! Ponieważ stosowane są wzorce klas – są to pliki nagłówkowe) wyposażone są w dokładne komentarze wyjaśniające zawartość. W związku z tym w niniejszej instrukcji szczegółowy opis implementacji struktur danych i algorytmów będzie pominięty.

3. Ogólne zasady budowy hierarchii klas

W programie zastosowana została jako podstawowa struktura danych – struktura tablicowa (patrz SmartTable.h – samowymiarująca się tablica). W miarę potrzeby inne podstawowe struktury danych (struktura listowa, drzewo z łączami wskaźnikowymi itp.) powinny być tworzone samodzielnie przez wykonującego ćwiczenie, tak by zachować ogólną budowę typów danych w programie. Podstawowe (abstrakcyjne) struktury są rozszerzane o dodatkowe atrybuty i/lub metody potrzebne do przechowywania w nich danych i wykonywania na tych danych podstawowych operacji (patrz np. SmartDataTable.h), a następnie odpowiednio rozszerzane dla potrzeb implementacji poszczególnych metodologii organizacji kolejki (patrz np. SmartTournament.h – wyszukiwanie turniejowe, SmartHeap.h – kopcowanie, itp.).

Kolejka priorytetowa implementowana jest w dwóch wersjach: leniwej i nadgorliwej (w tym przypadku występuje wyjątek od zasady zgodności nazwy klasy z nazwą pliku – obie wersje kolejki umieszczane są w tym samym pliku, np.: AbstractPriorityQueueLazy i AbstractPriorityQueueOfficious w pliku AbstractPriorityQueue.h). Ostateczna wersja kolejki dziedziczy dwubazowo po typie kolejki i po algorytmie implementacji jej interfejsu (patrz np.: HeapQueueLazy i HeapQueueOfficious w pliku HeapQueue.h).

We wszystkich kolejkach przechowywane mogą być wyłącznie takie obiekty, które należą do klas dziedziczących po AbstractPriorityObject wymuszającym odpowiedni interfejs (patrz SimpleObject.h). Chodzi tu o kompatybilność w obrębie wszystkich klas z metodami dostępu do danych i metodami statystyki działania algorytmów. Własne typy danych wykonującego ćwiczenie, powinny być zorganizowane podobnie do SimpleObject.

4. Przebieg ćwiczenia

Ćwiczenie polega na wykonaniu kilku zadań, wskazanych przez prowadzącego. Niektóre, przykładowe polecenia, podane są niżej (prowadzący może jednak, ustalić inne zadania, które jednak będą merytorycznie podobne). Zadania mogą być trojakiego rodzaju:

1) dotyczące analizy wybranego zestawu/zestawów: typ kolejki – struktura danych – algorytm implementujący interfejs – typ danych;
2) dotyczące optymalizacji metod interfejsu ze względu na osiąganie różnych celów;
3) związane z implementacją innych algorytmów;

Do wykonania analizy (polecenia typu 1) student powinien samodzielnie zaprojektować rodzaj i przebieg eksperymentów, których wykonanie udzieli odpowiedzi na zadane pytania. Od technicznej strony, na ogół sprowadza się to do napisania prostej procedury wywołującej metody interfejsu kolejki priorytetowej.

Polecenia typu 2 wymagają ingerencji w kod implementacji metod – do wprowadzenia modyfikacji, które są przewidziane przez autora ćwiczenia, wystarczą często bardzo drobne poprawki. W kilku przypadkach w kodzie są nawet fragmenty czy całe metody sugerujące studentowi rozwiązania. Oczywiście efektywne poprawki, których autor nie przewidział, a które wymyśli student, mogą być dodatkowo premiowane.

Polecenia typu 3 polegają na napisaniu kodu implementującego nowe struktury danych/algorytmy. Implementacja taka powinna wykorzystywać ogólną strukturę i budowę programu oraz być z nią całkowicie zgodna.

Przykładowe zadania do wykonania w trakcie wykonywania ćwiczenia:

a) zbadać dla jakich sekwencji wykonywania metod put/get podejście leniwe jest lepsze/gorsze/takie samo jak nadgorliwe,
b) przeanalizować i określić sposób zachowania się algorytmów w stosunku do danych o jednakowych priorytetach (np. relacje kolejności wkładania i wyjmowania), zaproponować zmiany w kodzie umożliwiające uzyskanie określonej kolejności,
c) przeanalizować działanie poszczególnych implementacji oraz wskazać przyczyny potencjalnych ograniczeń ich wydajności, stosowalności itp,
d) dokonać analizy implementacji kolejki dla różnych typów danych, zaproponować modyfikacje podnoszące wydajność dla poszczególnych typów danych,
e) dokonać analizy sprawności implementacji w zależności od sposobu używania kolejki przez programy zewnętrzne,
f) zastosować inne algorytmy w implementacji metod kolejki (np. kolejka nadgorliwa ze wstawianiem danej metodą połowienia, kolejka leniwa z sortowaniem daną metodą, itp.),
g) zastosować inne struktury danych w implementacji metod kolejki (np. struktura listowa z sortowaniem bąbelkowym, struktura listowa z turniejem, itp.),
h) zoptymalizować wybrane implementacje pod względem liczby wykonywanych porównań,
i) zoptymalizować wybrane implementacje pod względem liczby wykonywanych kopiowań.

5. Analiza wyników, wnioski, sprawozdanie

Na ocenę z ćwiczenia bezpośrednio wpływają:

- zawartość notatnika z ćwiczenia (w wersji papierowej) Na początku ćwiczenia należy wydrukować umieszczony na końcu niniejszej instrukcji Notatnik. Będzie on stanowił bieżący zapis przebiegu ćwiczenia i wyników pracy.
- zawartość **oddanego w terminie** sprawozdania (wyłącznie w postaci elektronicznej w formacie pdf). Sprawozdanie powinno zawierać następujące elementy:
  - cel ćwiczenia (własnymi słowami),
  - zakres i opis przeprowadzonego ćwiczenia (z wyszczególnieniem wskazówek prowadzącego),
  -
  - opis modyfikacji adaptacyjnych wprowadzonych do programu wraz z analizą rezultatów tych zmian,
  - podsumowanie ćwiczenia czyli wnioski: ***konkretne***, ***na temat***, ***samodzielne***, ***twórcze***.

Sprawozdanie (wyłącznie w formacie **pdf**) powinno zostać w terminie nadesłane pocztą elektroniczną, na adres podany przez prowadzącego zajęcia. Nazwa pliku zawierającego sprawozdanie powinna spełniać wymogi wzorca:

AISD_4_Nazwisko_Imie.pdf

Temat maila ze sprawozdaniem również powinien pasować do wzorca:

AISD_4_Nazwisko_Imie

Sprawozdania oddawane przez studentów powinny być **estetyczne** i **czytelne**. Sprawozdania nie spełniające tego wymogu będą niżej oceniane. Na końcową ocenę za ćwiczenie składają się:

- sposób wykonania ćwiczenia,
- samodzielność i pomysłowość w jego wykonaniu,
- sprawozdanie (jakość merytoryczna, czytelność, estetyka)

6. Wymagania

Do wykonania ćwiczenia będzie niezbędna znajomość algorytmów sortowania oraz struktur drzewiastych (turniej, kopiec, drzewo BST). W ramach przygotowania do ćwiczenia należy również zapoznać się z przykładowym programem (pełne listingi zamieszczone w internecie na stronie przedmiotu).

7. Literatura

1. R. Sedgevick *„Algorytmy w C++”*, Wydawnictwo RM, 1999,
2. J. Grębosz *„Symfonia C++”* Wydawnictwo Oficyna Kallimach 1997
3. J. Grębosz *„Pasja C++”* Wydawnictwo Oficyna Kallimach 1997

opracowanie dr inż. Adam Wojtasik
aw@imio.pw.edu.pl

NOTATNIK

AISDE LAB 4	Imię i nazwisko, Grupa dziekańska	Nr Indeksu	Data wykonania ćwiczenia	Data oddania protokołu

Punkty do wykonania	a	b	c	d	e	f	g	h	i	1	2	3	4

Implementacja/ struktura danych	Nadorliwa/ leniwa	Uwagi / Zauważone błędy / założenia do własnej implementacji

Polecenia niewymienione w instrukcji, ustalone przez prowadzącego:

1. .....
2. .....
3. .....
4. .....

NOTATNIK NALEŻY ODDAĆ PROWADZĄCEMU ZAJĘCIA W CHWILI ZAKOŃCZENIA ĆWICZENIA