

**TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN
KHOA CÔNG NGHỆ THÔNG TIN**

---o0o---



BÀI TẬP LAB MÔN CĐ TCDL

BÀI TẬP 2

Giảng viên hướng dẫn : **VŨ QUỐC HOÀNG**

Sinh viên thực hiện: **CAO QUỐC VIỆT**

MSSV: **22810218**

Lớp : **CN2022/2**

Khoá : **2022/2**

TP. Hồ Chí Minh, tháng 03 năm 2025

Bài làm

1. (6 điểm)

Viết module định nghĩa kiểu dữ liệu Decimal để thao tác với số thập phân có giá trị không vượt quá 1 tỉ và có 3 chữ số sau dấu thập phân. Kiểu số này hỗ trợ các thao tác sau:

- Nhập xuất và chuyển đổi với chuỗi.
- Chuyển đổi với kiểu số nguyên int và kiểu double của C/C++.
- Các phép toán lấy phần nguyên, phần lẻ và làm tròn.
- Các phép toán số học thông thường: cộng, trừ, nhân, chia, lấy đối, trị tuyệt đối.
- Các phép toán so sánh: bằng, khác, nhỏ hơn, lớn hơn, nhỏ hơn hoặc bằng, lớn hơn hoặc bằng.

Viết chương trình minh họa việc sử dụng kiểu Decimal trên.

2. (4 điểm)

Mô hình dữ liệu hàng đợi ưu tiên (priority queue) là mô hình dữ liệu cho phép quản lý một tập các đối tượng theo độ ưu tiên. Độ ưu tiên của mỗi đối tượng là một con số nguyên. Một hàng đợi ưu tiên có các thao tác đi kèm:

- IsEmpty: kiểm tra hàng đợi có rỗng (không chứa đối tượng nào) không.
- Insert: thêm một đối tượng có độ ưu tiên vào hàng đợi.
- RemoveMax: gỡ đối tượng có độ ưu tiên lớn nhất ra khỏi hàng đợi và trả về đối tượng đó.

- Đưa ra cấu trúc dữ liệu phù hợp và cài đặt cụ thể một hàng đợi ưu tiên quản lý các số nguyên.
- Từ cấu trúc dữ liệu đã xây dựng ở Câu (a), viết hàm sắp xếp tăng dần một dãy số nguyên.
- Dùng hàng đợi ưu tiên để cài đặt mô hình dữ liệu ngăn xếp (stack).
- Dùng hàng đợi ưu tiên để cài đặt mô hình dữ liệu hàng đợi (queue).

=====

I. TỰ ĐÁNH GIÁ

Hoàn thành tất cả các yêu cầu của cả 2 bài tập, viết chi tiết các hàm để test chức năng, tham khảo nguồn uy tín để tối ưu code, tham khảo bài giảng của thầy để có cách phân tích bài toán hợp lý.

Bảng đánh giá cá nhân:

STT	Yêu cầu	Mức độ	Giải thích
a	Nhập xuất và chuyển đổi chuỗi	Hoàn thành	- Constructor từ string - toString() - Xử lý các trường hợp
b	Chuyển đổi với int và double	Hoàn thành	- toInt() - toDouble() - fromInt() - fromDouble()
c	Phần nguyên, phần lẻ, làm tròn	Hoàn thành	- getIntegerPart() - getFractionalPart() - round()
d	Các phép toán số học	Hoàn thành	- +, -, *, / - unary - - abs()
e	Các phép toán so sánh	Hoàn thành	- ==, != - <, >, <=, >=

STT	Yêu cầu	Mức độ	Giải thích
a	Cài đặt hàng đợi ưu tiên	Hoàn thành	- isEmpty() - insert() - removeMax() - Xử lý FIFO cho cùng priority
b	Cài đặt HeapSort	Hoàn thành	- Sắp xếp tăng dần - Test với mảng có trùng lặp

c	Cài đặt Stack bằng PQ	Hoàn thành	- push() - pop() - Đảm bảo LIFO
d	Cài đặt Queue bằng PQ	Hoàn thành	- enqueue() - dequeue() - Đảm bảo FIFO

II. TỔ CHỨC SOURCE CODE

IDE: visual studio code

1. Cấu trúc thư mục chung

- **Decimal Project:**
 - Decimal.h: Định nghĩa struct Decimal và khai báo phương thức.
 - Decimal.cpp: Triển khai chi tiết các phương thức của Decimal.
 - Test.h: Khai báo hàm test cho Decimal.
 - Test.cpp: Triển khai các test case.
 - main.cpp: Hàm main() chạy demo chương trình.
- **Priority Queue Project:**
 - PriorityQueue.h: Định nghĩa class PriorityQueue và các phương thức.
 - PriorityQueue.cpp: Triển khai chi tiết phương thức của PriorityQueue.
 - Test.h: File header chứa khai báo các hàm test
 - Test.cpp: File chứa code chi tiết các test case
 - main.cpp: Hàm main() chứa các test case và chạy demo.

2. Công cụ và thư viện

- **Decimal:**
 - <string>, <sstream>, <iomanip>: Xử lý chuỗi và định dạng.

- `<stdexcept>`, `<cmath>`: Xử lý ngoại lệ và toán học.
- **Priority Queue:**
 - `<vector>`, `<utility>`: Quản lý heap và cặp giá trị.
 - `<iostream>`, `<stdexcept>`: Nhập xuất và xử lý lỗi.

III. PHÂN TÍCH BÀI TOÁN

1. Kiểu dữ liệu Decimal

- **Mục tiêu:** Xử lý số thập phân với 3 chữ số phần lẻ và giới hạn 1 tỉ.
- **Thiết kế:**
 - Lưu trữ dưới dạng long long value (giá trị $\times 1000$) và bool isNegative để xử lý dấu.
 - Tránh dùng float/double để đảm bảo độ chính xác.
 - Chuyển đổi qua lại với chuỗi bằng cách tách phần nguyên/thập phân và chuẩn hóa 3 chữ số.
 - Xử lý phép toán số học trên giá trị nguyên để tránh sai số.
 - Phần làm tròn dựa trên vị trí chữ số thập phân (từ -inf đến 3).

2. Priority Queue

- **Mục tiêu:** Quản lý phần tử theo độ ưu tiên và hỗ trợ Stack/Queue.
- **Thiết kế:**
 - Dùng **min-heap** với `vector<pair<int, int>>` (phần tử, độ ưu tiên) và `insertionOrder` để xử lý thứ tự thêm vào.
 - Độ ưu tiên được xác định bằng số nguyên: nhỏ hơn \rightarrow ưu tiên cao hơn.
 - Xử lý các phần tử cùng độ ưu tiên

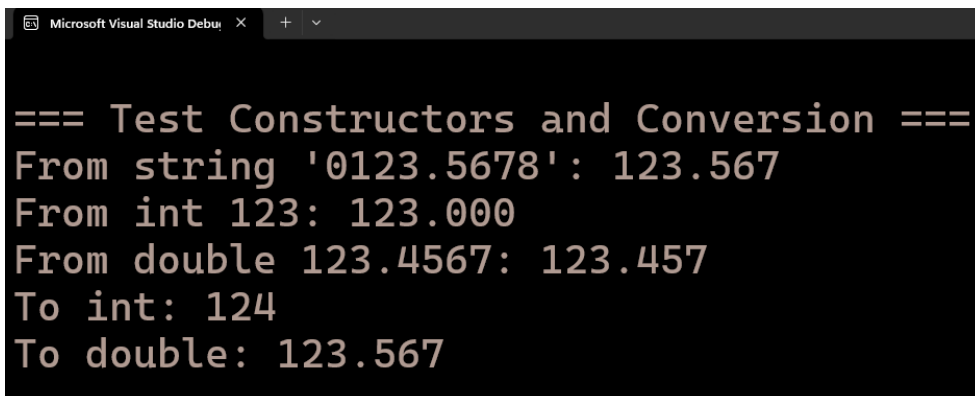
- **Stack:** Dùng số âm làm độ ưu tiên (phần tử mới có số âm nhỏ hơn → ưu tiên cao).
- **Queue:** Dùng số dương tăng dần (phần tử cũ có số dương nhỏ hơn → ưu tiên cao).
- Sắp xếp mảng bằng cách chèn tất cả phần tử vào Priority Queue và lấy ra tuần tự (độ phức tạp $O(n \log n)$).

III. CÁC CHỨC NĂNG CHI TIẾT

A. Kiểu dữ liệu Decimal

1. Nhập xuất và chuyển đổi:

- **Từ chuỗi:** Phân tích dấu, tách phần nguyên/thập phân, thêm số 0 nếu thiếu.
- **Sang chuỗi:** Định dạng lại với dấu chấm thập phân và đảm bảo 3 chữ số phần lẻ.



```

Microsoft Visual Studio Debu  X + -
=== Test Constructors and Conversion ===
From string '0123.5678': 123.567
From int 123: 123.000
From double 123.4567: 123.457
To int: 124
To double: 123.567
  
```

2. Chuyển đổi với int/double:

- `toInt()`: Làm tròn bằng cách chia cho 1000.
- `fromDouble()`: Nhân giá trị với 1000 và làm tròn đến 3 chữ số thập phân.

```
Microsoft Visual Studio Debu x + v

=== Test Constructors and Conversion ===
From string '0123.5678': 123.567
From int 123: 123.000
From double 123.4567: 123.457
To int: 124
To double: 123.567
```

3. Phép toán phần nguyên và làm tròn:

- `getIntegerPart()`: Chia nguyên cho 1000.
- `round(precision)`: Điều chỉnh dựa trên vị trí chữ số cần làm tròn.

```
Microsoft Visual Studio Debu x + v

=== Test Parts and Rounding ===
Number: 123.567
Integer part: 123
Fractional part: 0.567
Rounded to -2 decimal places: 100.000
Rounded to -1 decimal places: 120.000
Rounded to 0 decimal places: 124.000
Rounded to 1 decimal place: 123.600
Rounded to 2 decimal places: 123.570
Rounded to 3 decimal places: 123.567
Rounded to 4 decimal places: 123.567

Negative number: -123.567
Integer part: -123
Rounded to 1 decimal place: -123.600
```

4. Phép toán số học:

- Cộng/trừ: Cộng trừ giá trị value và điều chỉnh dấu.
- Nhân/chia: Tính toán trên giá trị đã nhân 1000, chia lại để giữ 3 chữ số.

```
Microsoft Visual Studio Debu  X + v

=== Test Arithmetic Operations ===
123.456 + 456.789 = 580.245
123.456 - 456.789 = -333.333
123.456 * 456.789 = 56393.342
123.456 / 456.789 = 0.270
Absolute of -123.456 = 123.456
Absolute of 0.123 = 0.123
```

5. So sánh:

- So sánh dấu trước, sau đó so sánh giá trị tuyệt đối.

```
Microsoft Visual Studio Debu  X + v

=== Test Comparison Operations ===
123.456 == 456.789: 0
123.456 != 456.789: 1
123.456 < 456.789: 1
123.456 <= 456.789: 1
123.456 > 456.789: 0
123.456 >= 456.789: 0
123.456 == -234.567: 0
123.456 != -234.567: 1
123.456 < -234.567: 0
123.456 <= -234.567: 0
123.456 > -234.567: 1
123.456 >= -234.567: 1
-234.567 == 456.789: 0
-234.567 != 456.789: 1
-234.567 < 456.789: 1
-234.567 <= 456.789: 1
-234.567 > 456.789: 0
-234.567 >= 456.789: 0
```

B. Priority Queue

1. Cài đặt hàng đợi ưu tiên:

- insert(): Thêm phần tử vào cuối heap, gọi heapifyUp().
- removeMax(): Đổi chỗ phần tử đầu với cuối, gọi heapifyDown().

```
Microsoft Visual Studio Debu x + v
=== Test Basic Priority Queue Operations ===
1. Test empty queue:
  Queue is empty: true
  Inserting: (value: 10, priority: 1)
  Queue is empty: false

2. Test insertion with different priorities:
  Inserting: (value: 5, priority: 2)
  Inserting: (value: 3, priority: 1)
  Inserting: (value: 7, priority: 3)
  Inserting: (value: 1, priority: 0)

3. Test removal (should be in priority order - smallest priority first):
  Expected order: 1, 3, 5, 7
  Actual order:
  Removed: 1
  Removed: 3
  Removed: 5
  Removed: 7

4. Test empty after removal:
  Queue is empty: true

5. Test exception on empty queue:
  Successfully caught exception: Priority queue is empty
```

```
Microsoft Visual Studio Debu  X + v

=== Test Priority Queue with Same Priority (FIFO Order) ===
1. Test insertion with same priority:
   Inserting three elements with priority 1:
   First: (value: 10, priority: 1)
   Second: (value: 20, priority: 1)
   Third: (value: 30, priority: 1)

2. Test removal (should maintain FIFO order):
   Expected order: 10, 20, 30
   Actual order:
   Removed: 10
   Removed: 20
   Removed: 30

3. Test mixed priorities with duplicates:
   Inserting: (value: 1, priority: 0)
   Inserting: (value: 2, priority: 0)
   Inserting: (value: 3, priority: 1)
   Inserting: (value: 4, priority: 1)

   Removing elements (should maintain both priority and FIFO):
   Expected order: 1, 2, 3, 4
   Actual order:
   Removed: 1
   Removed: 2
   Removed: 3
   Removed: 4
```

2. Sắp xếp mảng:

- Chèn tất cả phần tử vào Priority Queue, sau đó lấy ra theo thứ tự ưu tiên.

```
Microsoft Visual Studio Debu  X + v

=== Test Heap Sort Implementation ===
1. Original array: 64 34 25 12 22 11 90
2. Sorted array:  11 12 22 25 34 64 90

3. Array with duplicates:
   Original: 5 2 8 5 1 9 2 8
   Sorted:  1 2 2 5 5 8 8 9
```

3. Cài đặt Stack:

- `push()`: Thêm phần tử với độ ưu tiên giảm dần (số âm nhỏ dần).
- `pop()`: Gọi `removeMax()` để lấy phần tử mới nhất.

```
Microsoft Visual Studio Debug Console
=== Test Stack Implementation (LIFO using Priority Queue) ===
1. Test empty stack:
   Stack is empty: true
   Pushing: 10
   Stack is empty: false

2. Test push operation:
   Pushing: 1, 2, 3, 4

3. Test pop operation (should be LIFO order):
   Expected order: 4, 3, 2, 1
   Actual order:
   Popped: 4
   Popped: 3
   Popped: 2
   Popped: 1

4. Test empty stack operations:
   Stack is empty: true

5. Test exception on empty stack:
   Successfully caught exception: Stack is empty
```

4. Cài đặt Queue:

- enqueue(): Thêm phần tử với độ ưu tiên tăng dần (số dương tăng).
- dequeue(): Gọi removeMax() để lấy phần tử cũ nhất.

```
Microsoft Visual Studio Debug  x  +  v

=== Test Queue Implementation (FIFO using Priority Queue) ===
1. Test empty queue:
  Queue is empty: true
  Enqueuing: 10
  Queue is empty: false

2. Test enqueue operation:
  Enqueuing: 1, 2, 3, 4

3. Test dequeue operation (should be FIFO order):
  Expected order: 1, 2, 3, 4
  Actual order:
  Dequeued: 1
  Dequeued: 2
  Dequeued: 3
  Dequeued: 4

4. Test empty queue operations:
  Queue is empty: true

5. Test exception on empty queue:
  Successfully caught exception: Queue is empty

6. Test mixed operations:
  Enqueuing: 10
  Enqueuing: 20
  Dequeuing: 10
  Enqueuing: 30
  Remaining elements (should be 20, 30):
  Dequeued: 20
  Dequeued: 30
```

Nguồn tham khảo: https://en.wikipedia.org/wiki/Priority_queue

Code: [CaoQuocViet/CD-TCDL: HCMUS course](#)