Tugas Praktikum Queue

Praktikum Algoritma dan Struktur Data

Oleh:

Samudero Dirgantoro / 5223600016

Program Studi D4 Teknologi Game

Departemen Teknologi Multimedia Kreatif

Politeknik Elektronika Negeri Surabaya

2023/2024

A. Percobaan

1. (9.2) Queue Using linked list.

- C# ke C++:

```
#include <iostream>
class Queue{
private:
   class Node{
   public:
      int value;
      Node* next;
      Node(int v, Node* n) : value(v), next(n) {}};
   Node* head = NULL;
   Node* tail = NULL;
   int count = 0;
public:
   int Size(){
      return count;}
   bool Empty(){
      return count == 0;}
   void Print(){
      Node* temp = head;
      while (temp != NULL){
         std::cout << temp->value << " ";
         temp = temp->next;}
      std::cout << "\n";}
    int Peek(){
      if (Empty()){
         return -1;}
      return head->value;}
   void Enqueue(int value){
      Node* newNode = new Node(value, NULL);
```

```
        if (Empty()){
            head = newNode;
            tail = newNode;}
        else{
            tail->next = newNode;
            tail = newNode;}
        count++;}};
    int main(){
        Queue myQueue;
        myQueue.Enqueue(10);
        myQueue.Enqueue(20);
        myQueue.Enqueue(30);
        myQueue.Print();
        std::cout << "Peek: " << myQueue.Peek() << "\n";
        std::cout << "Size: " << myQueue.Size() << "\n";}
```

- Rangkuman: Program ini adalah implementasi dari struktur data queue yang menggunakan konsep FIFO atau First In First Out. Tujuan dari program ini adalah untuk menunjukkan penggunaan queue yang memiliki metode untuk menambahkan elemen ke dalam antrian (Enqueue), mengetahui ukuran antrian (Size), mengecek apakah antrian kosong (Empty), dan mencetak hasil queue dengan std::cout

2. (9.3) Queue Using linked list, using Add.

- C# ke C++:

```
void add(int value) {
    Node* temp = new Node(value, NULL);
    if (head == NULL) {
        head = tail = temp;
    } else {
        tail->next = temp;
        tail = temp;
```

```
    }
    count++;
}
```

- Rangkuman: Dalam kode tersebut, sebuah node baru dibuat dengan nilai yang diberikan dan pointer next diatur menjadi null. Kemudian, dilakukan pengecekan apakah antrian kosong. Jika iya, head dan tail diatur sebagai node baru. Jika tidak, node baru tersebut ditambahkan di belakang atau tail.

3. (9.4) Queue Using linked list, using Remove

- C# ke C++:

```
int Remove() {
        if (Empty()) {
            throw
std::runtime_error("QueueEmptyException");}
        int value = head->value;
        Node* temp = head;
        head = head->next;
        delete temp;
        count--;
        return value;}};
int main() {
    Queue q;
    for (int i = 1; i <= 100; i++) {
        q.Enqueue(i);}
    for (int i = 1; i <= 50; i++) {
        q.Remove();}
```

- Rangkuman: Potongan kode tersebut adalah fungsi yang terdapat metode remove() yang bisa digunakan untuk menghapus elemen terdepan dari antrian. Jika antrian kosong, maka akan menghasilkan pengecualian, berupa QueueEmptyException. Fungsi

remove() mengembalikan nilai dari elemen yang dihapus. Kemudian, di dalam metode Main, dilakukan inisialisasi objek queue dan pengisian antrian dengan angka dari 1 hingga 100 menggunakan perulangan. Setelah itu, dilakukan penghapusan 50 elemen terdepan dari antrian menggunakan perulangan kedua.

4. (9.5) Queue using a stack

- C# ke C++:

```
#include <iostream>
#include <stack>
class QueueUsingStack {
private:
   std::stack<int> stk1;
   std::stack<int> stk2;
public:
   void Add(int value) {
      stk1.push(value);}
   int Remove() {
      int value;
      if (!stk2.empty()) {
         value = stk2.top();
         stk2.pop();
         return value;}
      while (!stk1.empty()) {
         value = stk1.top();
         stk1.pop();
         stk2.push(value);}
      value = stk2.top();
      stk2.pop();
      return value;}};
int main() {
```

```
QueueUsingStack que;
que.Add(1);
que.Add(11);
que.Add(111);
std::cout << que.Remove() << "\n";
que.Add(2);
que.Add(21);
que.Add(211);
std::cout << que.Remove() << "\n";
std::cout << que.Remove() << "\n";}
```

- Rangkuman: Kode diatas adalah implementasi queue menggunakan dua stack. Queue disimpan dalam dua stack, yaitu stk1 dan stk2. Fungsi “add” digunakan untuk menambahkan elemen ke queue dengan menambahkan elemen ke stk1. Sedangkan, fungsi “remove” digunakan untuk menghapus elemen dari antrian. Jika stk2 tidak kosong, elemen dihapus dari stk2. Jika kosong, semua elemen akan dipindahkan dari stk1 ke stk2, kemudian elemen terdepan diambil dari stk2. Dalam fungsi utama “Main”, beberapa angka ditambahkan ke antrian menggunakan fungsi “add”, kemudian beberapa elemen dihapus menggunakan fungsi remove, dan hasilnya dicetak dengan std::cout.

5. Pemecahan Masalah: Reverse Stack

- C# ke C++:

```
#include <iostream>
#include <stack>
#include <queue>
void reverseStack(std::stack<int>& stk) {
  std::queue<int> q;
  while (!stk.empty()) {
```

```
        q.push(stk.top());
        stk.pop();
    }
    while (!q.empty()) {
        stk.push(q.front());
        q.pop();}}
int main() {
    std::stack<int> stk;
    stk.push(1);
    stk.push(2);
    stk.push(3);
    std::cout << "Original stack:" << "\n";
    std::stack<int> stk_copy = stk;
    while (!stk_copy.empty()) {
        std::cout << stk_copy.top() << " ";
        stk_copy.pop();}
    std::cout << "\n";
    reverseStack(stk);
    std::cout << "Reversed stack:" << "\n";
    while (!stk.empty()) {
        std::cout << stk.top() << " ";
        stk.pop();}
    std::cout << "\n";}
```

- Rangkuman: program ini bertujuan untuk membalikkan isi dari sebuah stack menggunakan sebuah queue. Fungsi add(int value) bertugas untuk menambahkan elemen ke dalam queue, sedangkan fungsi reverseStack() bekerja untuk pembalikan stack. Cara kerjanya adalah dengan mempop semua elemen dari stack satu per satu dan menambahkannya ke dalam queue. Setelah itu, elemen-elemen tersebut dihapus dari queue dan dimasukkan kembali ke

dalam stack, sehingga urutan elemen dalam stack menjadi terbalik. Akhirnya, fungsi printReversedStack() digunakan untuk mencetak isi stack.

6. Pemecahan Masalah: Stack using a Queue

- C# ke C++:

```
#include <iostream>
#include <queue>
class StackUsingQueue {
private:
   std::queue<int> q1;
   std::queue<int> q2;
public:
   void push(int value) {
      q1.push(value);}
   int pop() {
      while (q1.size() > 1) {
         q2.push(q1.front());
         q1.pop();}
      int popped = q1.front();
      q1.pop();
      std::swap(q1, q2);
      return popped;}};
int main() {
   StackUsingQueue stack;
   stack.push(1), stack.push(2), stack.push(3);
   std::cout << stack.pop() << stack.pop() <<
stack.pop() << "\n";}
```

- Rangkuman: Kode di atas merupakan implementasi dari stack menggunakan dua queue. Tujuan dari implementasi ini adalah untuk memberi contoh pemakaian push dan pop pada stack menggunakan

O(1) untuk push dan O(n) untuk pop. Ketika operasi push dilakukan, elemen baru ditambahkan ke queue pertama. Sedangkan pada operasi pop, elemen-elemen dari queue pertama dipindahkan ke queue kedua kecuali elemen terakhir, dan kemudian nama queue pertama dan queue kedua ditukar.