**Laporan Praktikum**

**Algoritma Struktur Data**

Queue

**Oleh:**

Oscar Javier Abdullah/ 5223600001

**Program Studi STr Teknologi Game**

**Departemen Teknologi Multimedia Kreatif**

**Politeknik Elektronika**

**Negeri Surabaya**

**2024**

**Implementasi Queue menggunakan linked list**

Berikut Programnya :

#include <iostream>

using namespace std;

class Queue {

private:

struct Node {

int value;

Node\* next;

Node(int v, Node\* n) : value(v), next(n) {}

};

Node\* head = nullptr;

Node\* tail = nullptr;

int count = 0;

public:

int size() { return count; }

bool empty() { return count == 0; }

void print() {

Node\* temp = head;

while (temp != nullptr) {

cout << temp->value << " ";

temp = temp->next;

}

cout << endl;

}

int peek() {

if (empty()) {

throw invalid\_argument("QueueEmptyException");

}

return head->value;

}

void enqueue(int value) {

Node\* temp = new Node(value, nullptr);

if (head == nullptr) {

head = tail = temp;

} else {

tail->next = temp;

tail = temp;

}

count++;

}

int dequeue() {

if (empty()) {

throw invalid\_argument("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 <= 70; i++) {

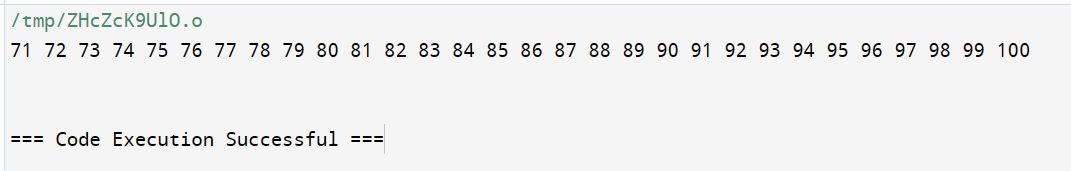
q.dequeue();

}

q.print();

return 0;

}

Hasil Output :

**Enqueue Pada Queue menggunakan linked list**

Programnya:

void Queue::enqueue(int value) {

Node\* temp = new Node(value, nullptr);

if (head == nullptr) {

head = tail = temp;

} else {

tail->next = temp;

tail = temp;

}

count++;

}

**Dequeue pada queue linked list**

int Queue::dequeue() {

if (empty()) {

throw invalid\_argument("QueueEmptyException");

}

int value = head->value;

Node\* temp = head;

head = head->next;

delete temp;

count--;

return value;

}

**Implementasi queue menggunakan 2 Stack**

#include <iostream>

#include <stack>

using namespace std;

class QueueUsingStack {

private:

stack<int> stk1, stk2;

public:

void enqueue(int value) {

stk1.push(value);

}

int dequeue() {

int value;

if (stk2.empty()) {

while (!stk1.empty()) {

value = stk1.top();

stk1.pop();

stk2.push(value);

}

}

if (stk2.empty()) {

throw invalid\_argument("QueueEmptyException");

}

value = stk2.top();

stk2.pop();

return value;

}

};

int main() {

QueueUsingStack que;

que.enqueue(1);

que.enqueue(11);

que.enqueue(111);

cout << que.dequeue() << endl; // Output: 1

que.enqueue(2);

que.enqueue(21);

que.enqueue(211);

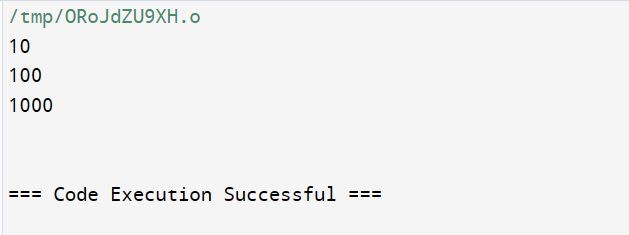
cout << que.dequeue() << endl; // Output: 11

cout << que.dequeue() << endl; // Output: 111

return 0;

}

**Hasil Output:**



Tugas:

**Stack Using Queue**

Programnya :

#include <iostream>

#include <queue>

using namespace std;

class StackUsingQueue {

private:

queue<int> q;

public:

void push(int value) {

int size = q.size();

q.push(value);

// Memindahkan elemen-elemen lain ke belakang

for (int i = 0; i < size; i++) {

q.push(q.front());

q.pop();

}

}

void pop() {

if (q.empty()) {

throw out\_of\_range("Stack is empty");

}

q.pop();

}

int top() {

if (q.empty()) {

throw out\_of\_range("Stack is empty");

}

return q.front();

}

bool empty() {

return q.empty();

}

};

int main() {

StackUsingQueue stack;

stack.push(1);

stack.push(2);

stack.push(3);

cout << "Top: " << stack.top() << endl;

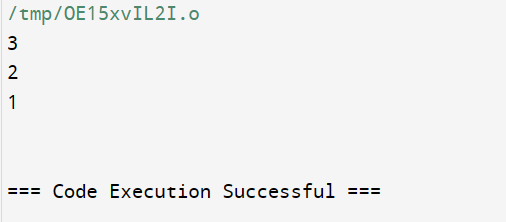
stack.pop();

cout << "Top: " << stack.top() << endl;

return 0;

}

Hasil Output :

****

**Reverse Queue**

#include <iostream>

#include <queue>

#include <stack>

void reverseQueue(std::queue<int> &que) {

std::stack<int> stk;

while (!que.empty()) {

stk.push(que.front());

que.pop();

}

while (!stk.empty()) {

que.push(stk.top());

stk.pop();

}

}

int main() {

std::queue<int> que;

que.push(1);

que.push(2);

que.push(3);

reverseQueue(que);

while (!que.empty()) {

std::cout << que.front() << '\n';

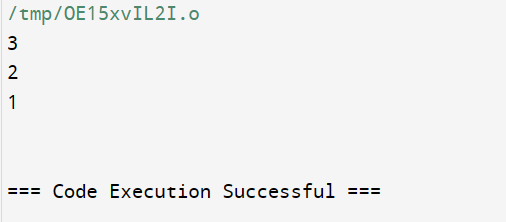
que.pop();

}

return 0;

}

Hasil Output :

****