1. 9.2 Struktur queue

Pada bagian kita akan melihat struktur dasar queue dengan menggunakan linked list

struct Node {

int value;

Node\* next;

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

Di dalamnya terdapat function size yang akan mengembalikan panjang queue

int size() {

return count;

Ada juga function Is Empty yang mengecek apakah panjang queue adalah 0

bool isEmpty() {

return count == 0;

Lalu ada function print yang menampilkan semua isi queue dari head hingga tail

void print() {

Node\* temp = head;

while (temp != nullptr) {

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

temp = temp->next;

}

cout << endl;

Terakhir ada function peek yang akan melihat nilai dari head

int peek() {

if (isEmpty()) {

throw "QueueEmptyException";

}

return head->value;

1. 9.3 Menambah enqueue pada Queue dengan linked list

Bagian ini sama dengan Kode sebelumnya hanya saja di tambahkan dengan function add yang berfungsi sebagai enqueue

Fungsi add adalah mengecek apakah head kosong jika iya maka node baru akan menjadi head dan tail

void add(int value) {

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

if (head == nullptr) {

head = tail = temp;

Jika tidak node baru akan di taruh di belakang tail dan menjadi tail baru

} else {

tail->next = temp;

tail = temp;

Lalu value count akan di tambah setelah salah satu dari 2 aksi di atas berjalan

count++;

1. 9.4 Menambah dequeue pada Queue dengan Linked list

Sama dengan sebelumnya Pada bagian ini hanya menambahkan function remove yang berperan sebagai dequeue, fungsi tersebut bekerja dengan mengecek apakah node kosong terlebih dahulu jika iya maka kode akan berakhir error dan memberikan pesan bahwa queue kosong

int remove() {

if (isEmpty()) {

throw "QueueEmptyException";

Jika queue tidak kosong maka kode akan membuat integer baru dengan mengambil value milik head queue lalu head akan di pindah ke belakang dan count akan di kurangi dan integer sebelumnya di return sebagai nilai

int value = head->value;

head = head->next;

count--;

return value;

1. 9.5 Queue dengan stack

Pada bagian ini kita akan melakukan queue dengan stack yang perlu dilakukan adalah meng include stack agar kita bisa melakukan stack dengan cepat

#include <stack>

Lalu kita perlu membuat 2 stack karena kita butuh 2 stack untuk queue

stack<int> stk1;

stack<int> stk2;

Lalu kita tambahkan fungsi add/enqueue yang akan mengpush data pada stk1

void add(int value) {

stk1.push(value);

}

Lalu kita perlu membuat fungsi enqueue nya dan fungsi tersebut di awali dengan membuat int value dan mengecek apakah stack 2 ada isinya jika iya maka kita akan mengambil nilai dari stack kedua sebagai value dan mereturn nya

int remove() {

int value;

if (!stk2.empty()) {

value = stk2.top();

stk2.pop();

return value;

Lalu jika stack kedua kosong maka kode akan mulai memindah semua data pada stack 1 ke stack kedua agar data yang masuk pertama (yang berada pada paling bawah) bisa berada di atas, begitu pula sebaliknya

while (!stk1.empty()) {

value = stk1.top();

stk1.pop();

stk2.push(value);

Setelah stack pertama kosong maka value akan megambil nilai dari top stack kedua dan melakukan pop pada stack kedua dan value akan di return

value = stk2.top();

stk2.pop();

return value;

}

1. Melakukan Stack dengan queue

Sama seperti sebelumnya pertama kita harus melakukan import queue lalu membuat 2 queue karena kita memerlukan 2 queue

#include <queue>

queue<int>que1;

queue<int>que2;

Lalu kita akan membuat fungsi add/push yang akan melakukan push ke dalam que1

void add(int value) {

que1.push(value);

Dan untuk fungsi remove/pop, fungsi tersebut akan mengecek apakah que1 kosong atau tidak jika kosong maka akan muncul pesan queue kosong dan di return

void remove(){

if(que1.empty()){ cout<<"Stack Kosong\n"; return;}

Lalu kode tersebut mulai mendeklarasi 2 int yang akan di gunakan selagi iterasi yaitu adalah v sebagai value dan pjg sebagai panjang queue yang di pilih,

lalu akan terjadi for loop yang akan melakukan dequeue pada que1 dan datanya akan di pindah kedalam v lalu di push kedalam que2 hingga tersisa 1 karena sudah di atur dalam for loop, dan ketika sudah sampai terakhir datanya akan di tampilkan dan di pop

int v, pjg = que1.size();

for (int i = 1; i < pjg ; i++){

v = que1.front();

que1.pop();

que2.push(v);

} v = que1.front();

cout <<"Data yang di pop: " << v << endl;

que1.pop();

Lalu semua data yang tadi di pindah kedalam que2 akan di kembalikan ke que1 dengan cara yang sama tetapi untuk kali ini sampai habis

pjg = que2.size();

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

v = que2.front();

que2.pop();

que1.push(v);

1. Reverse Stack

Kode ini bekerja dengan cara yang sangat simpel dan hanya perlu membuat 1 function, tetapi seperti sebelumnya yaitu kita perlu menginclude stack dahulu, Fungsi reverse stack ini memerlukan sebuah stack sebagai trigger, lalu fungsi tersebut akan membuat stack temporary dan panjang dari stack yang di input lalu memindahkan semua isi stak input kedalam stack temporary dan ketika selesai maka stack sudah menjadi reverse (terbalik)

void reverseStack(stack<int>& stk){

stack<int> temp; int pjg = stk.size();

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

temp.push(stk.top());

stk.pop();

}

stk = temp;

Lalu sisa dari kode tersebut adalah menu input yang terbuat dari while loop simpel

#include <iostream>

#include <stack>

using namespace std;

void reverseStack(stack<int>& stk){

stack<int> temp; int pjg = stk.size();

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

temp.push(stk.top());

stk.pop();

}

stk = temp;

}

int main(){

stack<int> stk,temp; int x,v=1;

while (v == 1){

cout << "\n\tMenu Stack & Reverse Stacm\n1.Push\n2.Pop\n3.Reverse Stack\n4.Exit\nInput: ";

cin >> x;

switch (x){

case 1:

cout << "Masukan data: ";

cin >> x; stk.push(x); break;

case 2:if (stk.empty()){cout << "Stack Kosong";break;}

cout << "Data yang di pop: " << stk.top();

stk.pop(); break;

case 3:if (stk.empty()){cout << "Stack Kosong";break;}

temp = stk; x = temp.size(); cout << "Stack awal: ";

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

cout << temp.top() << " ";

temp.pop();

}reverseStack(stk);

cout << "\nStack setelah di reverse: "; temp = stk;

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

cout << temp.top() << " ";

temp.pop();

}

break;

case 4:

v = 2; break;

default: cout << "Input salah! masukan 1 sampai 4";

break;

}

}

}