Tugas Evaluasi II Makassar, 14 Oktober 2024

**PRAKTIKUM**

**STRUKTUR DATA**

Nama : Andi Ikhlas Mallomo

Stambuk : 13020230251

Frekuensi : TI\_SD–7 (A7)

Dosen : Syariful Mujaddid, S.Kom., M.T.

Asisten 1 : Annisa Pratama Putri

Asisten 2 : Rahma Puspitasari

**PROGRAM STUDI TEKNIK INFORMATIKA**

**FAKULTAS ILMU KOMPUTER**

**UNIVERSITAS MUSLIM INDONESIA**

**MAKASSAR**

**2024**

1. Buatlah sebuah program geometri (Segitiga, Kotak, Persegi Panjang, Lingkaran) menggunakan QUEUE atau STACK (pilih salah satunya)

#include <iostream> // Mengimpor pustaka iostream untuk input dan output

using namespace std; // Menggunakan ruang nama standar

#define MAKS 1000 // Mendefinisikan makro MAKS dengan nilai 1000 sebagai ukuran maksimum antrian

// Struktur data untuk antrian

struct Queue {

int data[MAKS]; // Array untuk menyimpan elemen antrian

int front, rear; // Indeks untuk elemen terdepan dan terakhir

};

// Fungsi untuk menginisialisasi antrian

void inisialisasiQueue(Queue &q) {

q.front = -1; // Mengatur indeks depan antrian ke -1 (menunjukkan antrian kosong)

q.rear = -1; // Mengatur indeks belakang antrian ke -1 (menunjukkan antrian kosong)

}

// Fungsi untuk memeriksa apakah antrian kosong

bool kosong(Queue &q) {

return q.front == -1; // Mengembalikan true jika antrian kosong

}

// Fungsi untuk menambahkan elemen ke antrian

void enqueue(Queue &q, int nilai) {

if (q.rear == MAKS - 1) { // Memeriksa apakah antrian penuh

cout << "Antrian penuh!" << endl; // Menampilkan pesan bahwa antrian penuh

} else {

if (q.front == -1) q.front = 0; // Jika antrian kosong, set front ke 0

q.rear++; // Menaikkan indeks rear

q.data[q.rear] = nilai; // Menambahkan nilai ke posisi rear

}

}

// Fungsi untuk menghapus elemen dari antrian

int dequeue(Queue &q) {

if (kosong(q)) { // Memeriksa apakah antrian kosong

cout << "Antrian kosong!" << endl; // Menampilkan pesan bahwa antrian kosong

return -1; // Mengembalikan -1 sebagai indikator antrian kosong

} else {

int nilai = q.data[q.front]; // Mengambil nilai dari posisi front

// Memeriksa apakah antrian hanya memiliki satu elemen

if (q.front >= q.rear) {

q.front = -1; // Set front ke -1 (antrian kosong)

q.rear = -1; // Set rear ke -1 (antrian kosong)

} else {

q.front++; // Menaikkan indeks front

}

return nilai; // Mengembalikan nilai yang dihapus

}

}

// Fungsi untuk mencetak segitiga bintang

void cetakSegitiga(int tinggi) {

Queue q; // Membuat objek antrian

inisialisasiQueue(q); // Menginisialisasi antrian

for (int i = 1; i <= tinggi; i++) { // Loop untuk setiap baris segitiga

enqueue(q, i); // Menambahkan tinggi baris ke antrian

for (int j = 1; j <= q.data[q.rear]; j++) { // Loop untuk mencetak bintang

cout << "\* "; // Mencetak bintang

}

cout << endl; // Pindah ke baris berikutnya

}

}

// Fungsi untuk mencetak kotak bintang

void cetakKotak(int sisi) {

Queue q; // Membuat objek antrian

inisialisasiQueue(q); // Menginisialisasi antrian

for (int i = 0; i < sisi; i++) { // Loop untuk setiap baris kotak

enqueue(q, sisi); // Menambahkan sisi kotak ke antrian

for (int j = 0; j < q.data[q.rear]; j++) { // Loop untuk mencetak bintang

cout << "\* "; // Mencetak bintang

}

cout << endl; // Pindah ke baris berikutnya

}

}

// Fungsi untuk mencetak persegi panjang bintang

void cetakPersegiPanjang(int panjang, int lebar) {

Queue q; // Membuat objek antrian

inisialisasiQueue(q); // Menginisialisasi antrian

for (int i = 0; i < lebar; i++) { // Loop untuk setiap baris persegi panjang

enqueue(q, panjang); // Menambahkan panjang persegi panjang ke antrian

for (int j = 0; j < q.data[q.rear]; j++) { // Loop untuk mencetak bintang

cout << "\* "; // Mencetak bintang

}

cout << endl; // Pindah ke baris berikutnya

}

}

// Fungsi untuk mencetak lingkaran bintang

void cetakLingkaran(int radius) {

Queue q; // Membuat objek antrian

inisialisasiQueue(q); // Menginisialisasi antrian

// Loop untuk menentukan titik-titik pada lingkaran

for (int i = 0; i <= 2 \* radius; ++i) {

for (int j = 0; j <= 2 \* radius; ++j) {

// Menghitung jarak dari titik pusat

float jarak = sqrt((i - radius) \* (i - radius) + (j - radius) \* (j - radius));

// Enqueue 1 jika berada pada tepi lingkaran, 0 jika tidak

enqueue(q, (jarak > radius - 0.5 && jarak < radius + 0.5) ? 1 : 0);

}

}

// Loop untuk mencetak lingkaran

for (int i = 0; i <= 2 \* radius; ++i) {

for (int j = 0; j <= 2 \* radius; ++j) {

int isBintang = dequeue(q); // Mengambil nilai dari antrian

cout << (isBintang ? "\* " : " "); // Mencetak bintang atau spasi

}

cout << endl; // Pindah ke baris berikutnya

}

}

// Fungsi utama program

int main() {

system("clear"); // Menghapus layar (hanya berfungsi di terminal Unix/Linux)

int pilihan, dimensi1, dimensi2; // Mendeklarasikan variabel untuk pilihan dan dimensi

// Loop untuk menu program

while (true) {

cout << "NAMA : Andi Ikhlas Mallomo" << endl; // Menampilkan nama

cout << "Stambuk : 13020230251" << endl; // Menampilkan stambuk

cout << "\nMenu Bentuk Geometri:\n"; // Menampilkan menu

cout << "1. Cetak Segitiga\n"; // Pilihan 1

cout << "2. Cetak Kotak\n"; // Pilihan 2

cout << "3. Cetak Persegi Panjang\n"; // Pilihan 3

cout << "4. Cetak Lingkaran\n"; // Pilihan 4

cout << "5. Keluar\n"; // Pilihan 5

cout << "Masukkan pilihan: "; // Meminta input pilihan

cin >> pilihan; // Menerima input dari pengguna

if (pilihan == 5) { // Jika pilihan adalah 5, keluar dari program

cout << "Keluar dari program.\n"; // Menampilkan pesan keluar

break; // Keluar dari loop

}

// Menggunakan switch untuk memilih bentuk yang akan dicetak

switch (pilihan) {

case 1: // Kasus untuk mencetak segitiga

cout << "Masukkan tinggi segitiga: "; // Meminta tinggi segitiga

cin >> dimensi1; // Menerima input tinggi

cetakSegitiga(dimensi1); // Memanggil fungsi cetakSegitiga

break;

case 2: // Kasus untuk mencetak kotak

cout << "Masukkan sisi kotak: "; // Meminta sisi kotak

cin >> dimensi1; // Menerima input sisi

cetakKotak(dimensi1); // Memanggil fungsi cetakKotak

break;

case 3: // Kasus untuk mencetak persegi panjang

cout << "Masukkan panjang persegi panjang: "; // Meminta panjang

cin >> dimensi1; // Menerima input panjang

cout << "Masukkan lebar persegi panjang: "; // Meminta lebar

cin >> dimensi2; // Menerima input lebar

cetakPersegiPanjang(dimensi1, dimensi2); // Memanggil fungsi cetakPersegiPanjang

break;

case 4: // Kasus untuk mencetak lingkaran

cout << "Masukkan radius lingkaran: "; // Meminta radius lingkaran

cin >> dimensi1; // Menerima input radius

cetakLingkaran(dimensi1); // Memanggil fungsi cetakLingkaran

break;

default: // Jika pilihan tidak valid

cout << "Pilihan tidak valid!\n"; // Menampilkan pesan pilihan tidak valid

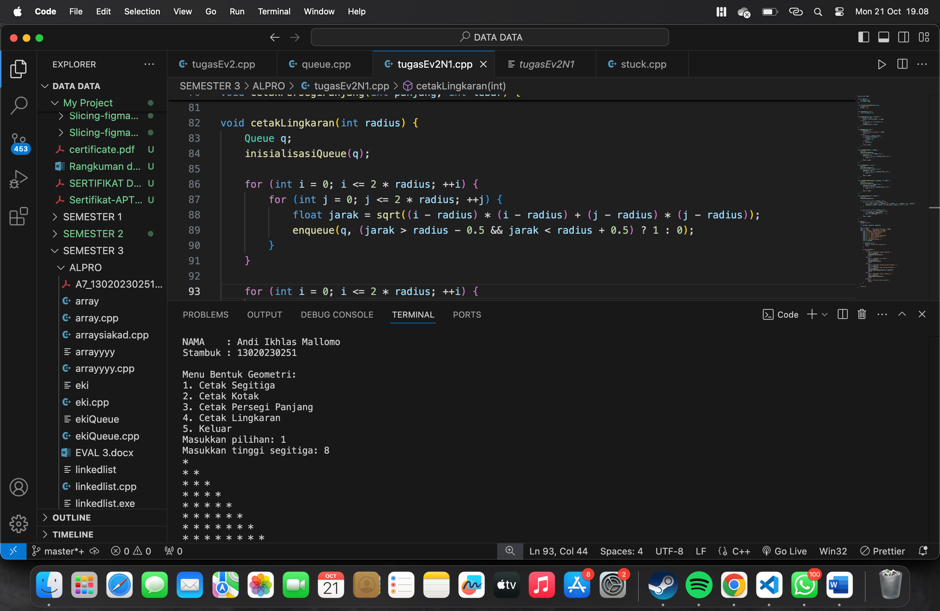
break;

}

}

return 0; // Mengembalikan 0 menandakan program berakhir dengan sukses

}



A screenshot of a computer

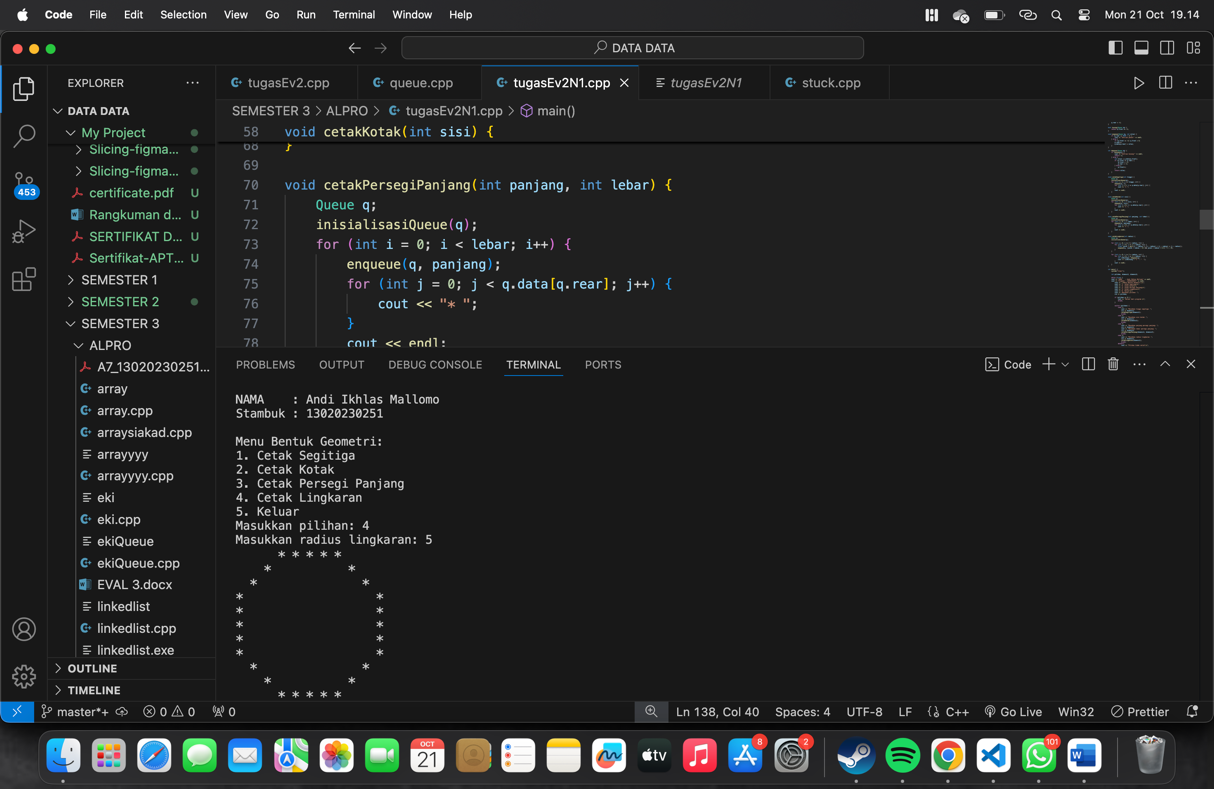
Description automatically generated

A screenshot of a computer

Description automatically generated

A black and white logo

Description automatically generatedTop of Form



1. Buatlah sebuah program UMI mart menggunakan STACK dan QUEUE dimana terdapat beberapa opsi sebagai berikut - Menambah pelanggan ke dalam antrian (Menggunakan QUEUE) - Melihat daftar antrian pelanggan (Menggunakan QUEUE) - Menambah barang ke dalam Belanjaan pelanggan (Menggunakan STACK) - Menyelesaikan Belanjaan pelanggan (Menggunakan Queue)

#include <iostream> // Mengimpor library untuk input dan output.

using namespace std; // Menggunakan namespace standar agar tidak perlu menulis std::.

const int MAKS = 100; // Batas maksimal jumlah elemen dalam antrian dan stack.

struct Antrian { // Mendefinisikan struktur Antrian.

int depan, belakang; // Indeks untuk menunjukkan elemen depan dan belakang dari antrian.

string pelanggan[MAKS]; // Array untuk menyimpan nama pelanggan.

Antrian() { // inisialisasi objek Antrian.

depan = -1; // Inisialisasi indeks depan ke -1 (antrian kosong).

belakang = -1; // Inisialisasi indeks belakang ke -1 (antrian kosong).

}

bool penuh() { // Fungsi untuk memeriksa apakah antrian penuh.

return belakang == MAKS - 1; // Mengembalikan true jika belakang mencapai indeks maksimal.

}

bool kosong() { // Fungsi untuk memeriksa apakah antrian kosong.

return depan == -1 || depan > belakang; // Mengembalikan true jika antrian kosong.

}

void tambah(string pelangganBaru) { // Fungsi untuk menambahkan pelanggan baru ke antrian.

if (penuh()) { // Memeriksa apakah antrian penuh.

cout << "Antrian penuh, tidak dapat menambah pelanggan baru.\n"; // Pesan jika antrian penuh.

} else {

if (depan == -1) depan = 0; // Jika antrian kosong, set depan ke 0.

pelanggan[++belakang] = pelangganBaru; // Tambah pelanggan ke belakang

cout << "Pelanggan " << pelangganBaru << " ditambahkan ke dalam antrian.\n";

}

}

void hapus() { // Fungsi untuk menghapus pelanggan dari antrian.

if (kosong()) { // Memeriksa apakah antrian kosong.

cout << "Antrian kosong, tidak ada pelanggan untuk diselesaikan.\n"; // Pesan jika antrian kosong.

} else {

cout << "Pelanggan " << pelanggan[depan] << " telah menyelesaikan belanja.\n";

depan++; // Pindahkan depan ke pelanggan berikutnya.

}

}

void tampilkan() { // Fungsi untuk menampilkan semua pelanggan dalam antrian.

if (kosong()) { // Memeriksa apakah antrian kosong.

cout << "Antrian kosong.\n"; // Pesan jika antrian kosong.

} else {

cout << "Daftar antrian pelanggan:\n"; // Pesan utk menampilkan daftar pelanggan.

for (int i = depan; i <= belakang; i++) { // Loop untuk menampilkan semua pelanggan.

cout << i - depan + 1 << ". " << pelanggan[i] << endl; // Tampilkan pelanggan dengan nomor urut.

}

}

}

};

struct Stack { // Mendefinisikan struktur Stack.

int atas; // Indeks untuk elemen paling atas dari stack.

string barang[MAKS]; // Array untuk menyimpan nama barang.

string pelanggan[MAKS]; // Array untuk menyimpan nama pelanggan yang sesuai dengan barang.

int jumlahBarang[MAKS]; // Array untuk menyimpan jumlah barang berdasarkan pelanggan.

Stack() { // inisialisasi objek Stack.

atas = -1; // Inisialisasi stack kosong.

for (int i = 0; i < MAKS; i++) { // Loop untuk menginisialisasi jumlahBarang.

jumlahBarang[i] = 0; //Inisialisasi jumlahBarang untuk setiap pelanggan.

}

}

bool penuh() { // Fungsi untuk memeriksa apakah stack penuh.

return atas == MAKS - 1; // Mengembalikan true jika atas mencapai indeks maksimal.

}

bool kosong() { // Fungsi untuk memeriksa apakah stack kosong.

return atas == -1; // Mengembalikan true jika stack kosong.

}

void dorong(string item, string pelangganBaru) { // Fungsi untuk menambahkan barang ke stack.

if (penuh()) { // Memeriksa apakah stack penuh.

cout << "Keranjang belanja penuh, tidak dapat menambah barang.\n"; // Pesan jika keranjang penuh.

} else {

atas++; // Increment atas.

barang[atas] = item; // Tambahkan barang ke stack.

pelanggan[atas] = pelangganBaru; // Simpan nama pelanggan sesuai dengan barang.

jumlahBarang[atas] = atas; // Simpan indeks barang.

cout << "Barang " << item << " ditambahkan ke dalam keranjang belanja untuk " << pelangganBaru << ".\n"; // Pesan konfirmasi.

}

}

void tampilkanBarangPertama(string pelangganBaru) { // Fungsi untuk menampilkan barang pertama untuk pelanggan tertentu.

if (kosong()) { // Memeriksa apakah stack kosong.

cout << "Keranjang belanja kosong untuk " << pelangganBaru << ".\n"; // Pesan jika keranjang kosong.

} else {

// Mencari barang pertama yang ditambahkan untuk pelanggan ini.

for (int i = 0; i <= atas; i++) { // Loop untuk mencari barang untuk pelanggan.

if (pelanggan[i] == pelangganBaru) { // Jika pelanggan cocok.

cout << "Barang di keranjang pelanggan " << pelangganBaru << ": " << barang[i] << endl; // Tampilkan barang pertama.

return; // Keluar dari fungsi setelah menemukan barang.

}

}

cout << "Tidak ditemukan barang untuk pelanggan " << pelangganBaru << ".\n"; // Pesan jika tidak ditemukan barang.

}

}

void kosongkan() { // Fungsi untuk mengosongkan stack.

atas = -1; // Kosongkan stack dengan mengatur atas ke -1.

cout << "Keranjang belanja telah dikosongkan.\n"; // Pesan konfirmasi.

}

};

int main() { // Fungsi utama program.

system("clear"); // Membersihkan layar terminal (hanya berfungsi di beberapa sistem).

Antrian ikhlas; // Membuat objek ikhlas dari struktur Antrian.

Stack ikhlass; // Membuat objek ikhlass dari struktur Stack.

int pilihan; // Menyimpan pilihan menu dari pengguna.

string nama, barang; // Variabel untuk menyimpan nama pelanggan dan barang.

do {

// Menampilkan informasi program dan menu.

cout << "Nama : Andi Ikhlas Mallomo\n"; // Menampilkan nama pemilik program.

cout << "Nim : 13020230251\n"; // Menampilkan NIM pemilik program.

cout << "\nUMI Mart\n"; // Judul program.

cout << "1. Tambah pelanggan ke antrian\n"; // Opsi 1: Tambah pelanggan.

cout << "2. Tambah barang ke keranjang pelanggan\n"; // Opsi 2: Tambah barang.

cout << "3. Tampilkan antrian pelanggan\n"; // Opsi 3: Tampilkan antrian.

cout << "4. Tampilkan barang di keranjang pelanggan\n"; // Opsi 4: Tampilkan barang.

cout << "5. Selesaikan belanja dan hapus pelanggan dari antrian\n"; // Opsi 5: Selesaikan belanja.

cout << "6. Keluar\n"; // Opsi 6: Keluar dari program.

cout << "Pilih opsi: "; // Pesan untuk memilih opsi.

cin >> pilihan; // Membaca input pilihan dari pengguna.

cin.ignore(); // Mengabaikan karakter newline agar input berikutnya bisa menggunakan 'getline'.

switch (pilihan) { // Memilih aksi berdasarkan pilihan pengguna.

case 1: // Kasus untuk menambah pelanggan.

system("clear"); // Membersihkan layar.

cout << "Masukkan nama pelanggan: "; // Pesan untuk memasukkan nama pelanggan.

getline(cin, nama); // Membaca nama pelanggan.

ikhlas.tambah(nama); // Menambahkan pelanggan ke antrian.

break;

case 2: // Kasus untuk menambah barang ke keranjang.

system("clear"); // Membersihkan layar.

cout << "Masukkan nama barang: "; // Pesan untuk memasukkan nama barang.

getline(cin, barang); // Membaca nama barang.

ikhlass.dorong(barang, nama); // Menambahkan barang ke keranjang (stack) dengan nama pelanggan saat ini.

break;

case 3: // Kasus untuk menampilkan antrian pelanggan.

system("clear"); // Membersihkan layar.

ikhlas.tampilkan(); // Menampilkan semua pelanggan dalam antrian.

break;

case 4: // Kasus untuk menampilkan barang di keranjang pelanggan.

system("clear"); // Membersihkan layar.

if (!ikhlas.kosong()) { // Memeriksa apakah antrian tidak kosong.

ikhlass.tampilkanBarangPertama(ikhlas.pelanggan[ikhlas.depan]); // Menampilkan barang pertama dalam keranjang untuk pelanggan di depan antrian.

} else {

cout << "Tidak ada pelanggan dalam antrian.\n"; // Jika antrian kosong.

}

break;

case 5: // Kasus untuk menyelesaikan belanja pelanggan.

system("clear"); // Membersihkan layar.

if (!ikhlas.kosong()) { // Memeriksa apakah antrian tidak kosong.

ikhlas.hapus(); // Menghapus pelanggan pertama dari antrian.

// Mengosongkan keranjang setelah pelanggan selesai belanja.

} else {

cout << "Antrian kosong, tidak ada pelanggan untuk diselesaikan.\n"; // Pesan jika antrian kosong.

}

break;

case 6: // Kasus untuk keluar dari program.

system("clear"); // Membersihkan layar.

ikhlass.kosongkan(); // Mengosongkan keranjang sebelum keluar.

cout << "Terima kasih telah menggunakan UMI Mart!\n"; // Pesan saat keluar dari program.

break;

default: // Kasus jika pilihan tidak valid.

cout << "Pilihan tidak valid, coba lagi.\n"; // Pesan jika pilihan tidak sesuai

}

} while (pilihan != 6); // Loop terus hingga pengguna memilih untuk keluar.

return 0; // Mengakhiri program.

A screen shot of a computer

Description automatically generated}

A screen shot of a computer

Description automatically generated

A black and white logo

Description automatically generated

A screen shot of a computer

Description automatically generated

A black and white logo

Description automatically generated

A screen shot of a computer

Description automatically generated

A black and white logo

Description automatically generated

A screen shot of a computer

Description automatically generated

A screen shot of a computer

Description automatically generated

A screen shot of a computer

Description automatically generated

1. Jelaskan konsep LIFO (Last In First Out) yang menjadi dasar dari stack dan konsep FIFO (First In First Out) yang menjadi dasar dari queue? Berikan contoh bagaimana stack dan queue diimplementasikan.Konsep LIFO (Last In First Out)

LIFO adalah suatu prinsip di mana elemen terakhir yang dimasukkan ke dalam struktur data adalah elemen pertama yang dikeluarkan. Dengan kata lain, elemen yang terakhir ditambahkan akan menjadi elemen pertama yang diambil.

Implementasi Stack:

Stack adalah struktur data yang menerapkan prinsip LIFO. Operasi utama yang dilakukan pada stack adalah:

Push: Menambahkan elemen ke atas stack.

Pop: Mengeluarkan elemen dari atas stack.

Contoh Implementasi Stack dalam C++:

A black and white logo

Description automatically generated

#include <iostream>

using namespace std;

struct Stack {

int data[5];

int top;

} tumpukan;

void inisialisasi() {

tumpukan.top = -1;

}

bool IsEmpty() {

return tumpukan.top == -1;

}

bool IsFull() {

return tumpukan.top == 4;

}

void push(int data) {

if (IsFull()) {

cout << "Stack sudah penuh!" << endl;

} else {

A black and white logo

Description automatically generated tumpukan.top++;

tumpukan.data[tumpukan.top] = data;

cout << "Data " << data << " berhasil dimasukkan..." << endl;

}

}

void pop() {

if (IsEmpty()) {

cout << "Stack kosong, pop dibatalkan..." << endl;

} else {

cout << "Data " << tumpukan.data[tumpukan.top] << " berhasil dikeluarkan..." << endl;

tumpukan.top--;

}

}

void tampilkan() {

if (IsEmpty()) {

cout << "Stack masih kosong..." << endl;

} else {

cout << "Isi stack mulai dari posisi top: ";

for (int i = 0; i <= tumpukan.top; i++) {

cout << tumpukan.data[i] << " ";

}

cout << endl;

}

A black and white logo

Description automatically generated}

void kosongkanStack() {

inisialisasi();

cout << "Stack berhasil dikosongkan..." << endl;

}

int main() {

system("clear");

int pilih, input;

inisialisasi();

do {

cout << "Implementasi Stack Dengan Array" << endl;

cout << "==================================" << endl;

cout << "Menu : " << endl;

cout << "1. Push data" << endl;

cout << "2. Pop data" << endl;

cout << "3. Tampil isi stack" << endl;

cout << "4. Kosongkan stack" << endl;

cout << "5. Keluar" << endl;

cout << "Masukkan pilihan [1..5] : ";

cin >> pilih;

switch (pilih) {

case 1:

A black and white logo

Description automatically generated cout << "Masukkan data: ";

cin >> input;

push(input);

break;

case 2:

pop();

break;

case 3:

tampilkan();

break;

case 4:

kosongkanStack();

break;

case 5:

cout << "Keluar dari program." << endl;

break;

default:

cout << "Pilihan tidak valid!" << endl;

}

cout << endl;

} while (pilih != 5);

return 0;

}

A screen shot of a computer

Description automatically generatedA black and white logo

Description automatically generated

Konsep FIFO (First In First Out)FIFO adalah prinsip di mana elemen pertama yang dimasukkan ke dalam struktur data adalah elemen pertama yang dikeluarkan. Dengan kata lain, elemen yang pertama ditambahkan akan menjadi elemen pertama yang diambil.

Implementasi Queue:

Queue adalah struktur data yang menerapkan prinsip FIFO. Operasi utama yang dilakukan pada queue adalah:

Enqueue: Menambahkan elemen ke belakang queue.

Dequeue: Mengeluarkan elemen dari depan queue.

Contoh Implementasi Queue dalam C++:

#include <iostream>

using namespace std;

const int MAX = 5;

A black and white logo

Description automatically generatedstruct queue {

string data[MAX];

int front;

int rear = -1;

}antrian;

void enQueue(string data) {

antrian.rear++;

antrian.data[antrian.rear] = data;

}

void deQueue() {

antrian.front++;

}

void tampilkan() {

for(int i = antrian.front; i <= antrian.rear; i++) {

cout << "Data Di Dalam Antrian :" << antrian.data[i] << endl;

}

}

int main() {

system("clear");

string nama;

int pil;

A black and white logo

Description automatically generated

do {

cout << "Menu:\n";

cout << "1. enQueue\n";

cout << "2. deQueue\n";

cout << "3. Tampilkan\n";

cout << "4. Keluar\n";

cout << "Pilih opsi (1/2/3/4): ";

cin >> pil;

switch (pil) {

case 1:

cin.ignore();

cout << "Masukkan Nama: ";

getline(cin, nama);

enQueue(nama);

break;

case 2:

deQueue();

break;

case 3:

tampilkan();

break;

case 4:

A black and white logo

Description automatically generated cout << "Keluar dari program." << endl;

break;

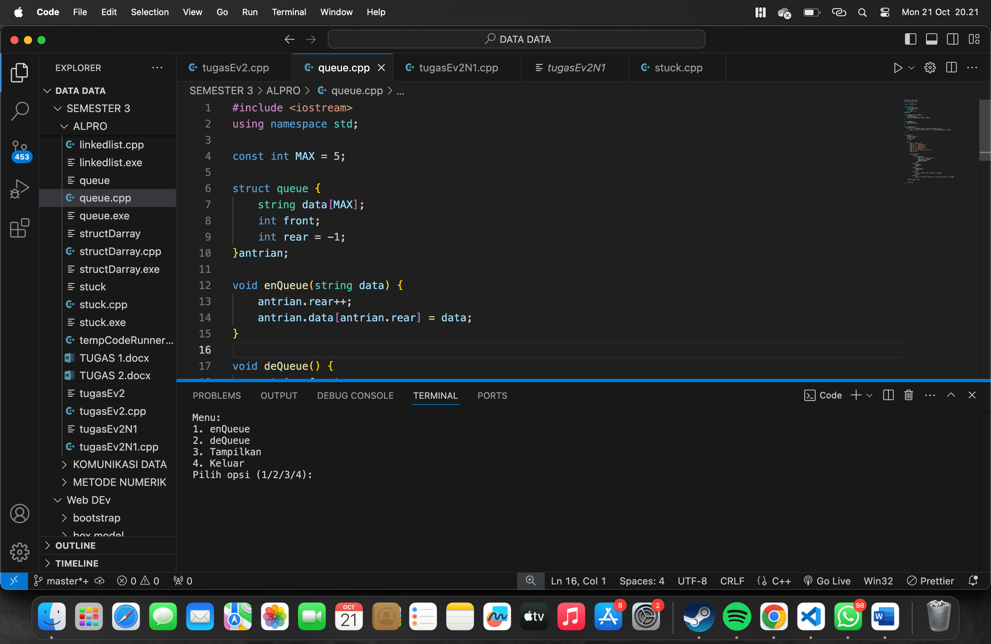
default:

cout << "Pilihan tidak valid. Silakan coba lagi." << endl;

}

} while (pil != 4);

return 0;

}