# LAPORAN PRAKTIKUM STRUKTUR DATA DAN ALGORITMA MODUL 7

**“Queue”**



# DISUSUN OLEH:

# Farrell Edric Kelvianto

# 2311102079

**S1 IF-11-B DOSEN:**

**Pak Wahyu Andi Saputra**, **S.Pd., M.Eng.**

# PROGRAM STUDI S1 INFORMATIKA FAKULTAS INFORMATIKA

**INSTITUT TEKNOLOGI TELKOM PURWOKERTO PURWOKERTO**

**2024**

1. **Dasar Teori**
2. **Pengertian Queue**

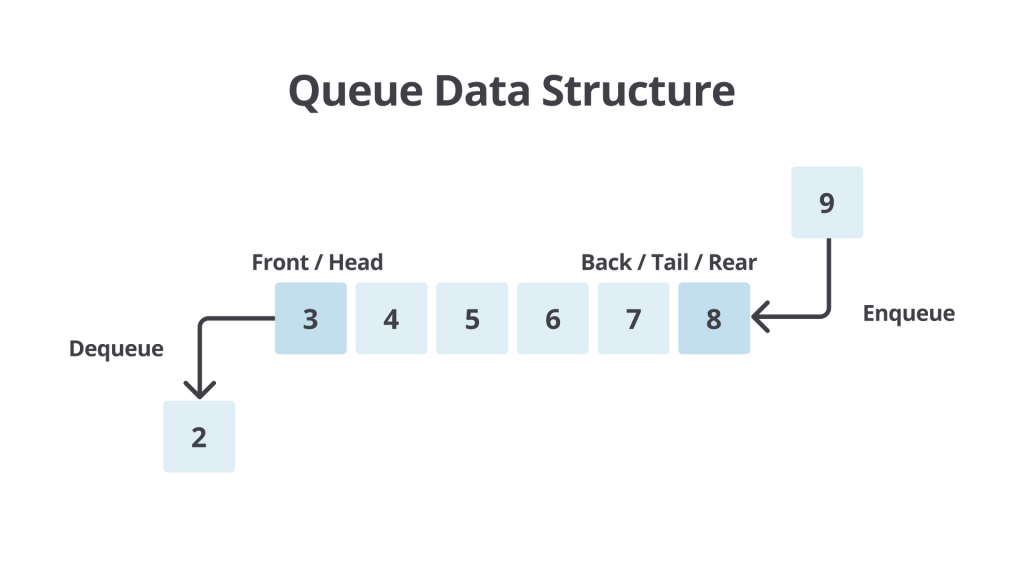
Queue adalah struktur untuk menyimpan data dengan cara FIFO yaitu First-in Fitst-Out atau LILO Last-In Last-Out. Data pertama akan dimasukan kedalam queue akan menjadi data yang pertama untuk dikeluarkan ari queue.queue ini mirip dengan konsep antrian jadi analoginya seperti kita mengantri membeli Mie Gacoan atau makanan lain, dan pasti didalam antrian yang paling pertama itu pasti dilayani terlebih dahulu. Untuk melakukan implementasi queue ini dapat dilakukan dengan menggunakan array atau linked list. Untuk perbedaan antara stack dan queue ini terdapat pada penghapusan elemen dan aturan penambahan. Pada stack ini operasi penambahan dan penghapusan elemen ini dilakukan di satu ujung. Elemen yang terakhir diinputkan berada paling dengan ujung atau dipaling atas maka akan dihapus paling awal, karena itulah sifatnya dikenal dengan LIFO. Pada queue ini ditempat yang berbeda yaitu melalui salah satu ujung jadi hanya satu insert saja maupun delete.

1. **Operasi Basic Queue**

Dalam operasi eneque ini berbeda dengan teknik hash table seperti kemarin disini ada beberapa operasi khusus untuk stack sesuai

1. Enqueue( ) : berfungsi untuk menambahkan data kedalam antrian.
2. Dequeue ( ) : berfungsi untuk mengeluarkan data dari antrian.
3. Peek ( ) : berfungsi untuk mengambil data didalam antrian tetapi tidak menghapusnya.
4. isEmpty ( ) : berfungsi untuk mengecek diantrian kosong atau tidak.
5. isFull ( ) : Berfungsi untuk mengecek antrian penuh atau tidak.
6. Size ( ) : ini berfungsi untuk menghitung jumlah elemen didalam queue.

* Sebagai berikut adalah gambaran dari suatu queue yang berisikan sebuah antrian:



**Guided**

Guided 1

|  |
| --- |
| #include <iostream>  using namespace std;  const int maksimalQueue = 5; // Maksimal antrian  int front = 0;               // Penanda antrian  int back = 0;                // Penanda  string queueTeller[5];       // Fungsi pengecekan  bool isFull()  { // Pengecekan antrian penuh atau tidak      return back == maksimalQueue;  }  bool isEmpty()  { // Antriannya kosong atau tidak      return back == 0;  }  void enqueueAntrian(string data)  {      if (isFull())      {          cout << "Antrian Penuh " << endl;      }      else      {          queueTeller[back] = data;          back++;      }  }  void dequeueAntrian()  {      if (isEmpty())      {          cout << "Antrian Kosong " << endl;      }      else      {          for (int i = 0; i < back - 1; i++)          {              queueTeller[i] = queueTeller[i + 1];          }          queueTeller[back - 1] = " ";          back--;      }  }  int countQueue()  { // Fungsi menghitung banyak antrian      return back;  }  void clearQueue()  { // Fungsi menghapus semua antrian      for (int i = 0; i < back; i++)      {          queueTeller[i] = "";      }      back = 0;      front = 0;  }  void viewQueue()  {      cout << "Data antrian teller " << endl;      for (int i = 0; i < maksimalQueue; i++)      {          if (queueTeller[i] != " ")          {              cout << i + 1 << queueTeller[i] << endl;          }          else          {              cout << i + 1 << ". (Kosong)" << endl;          }      }  }  int main()  {      enqueueAntrian("Andi");      enqueueAntrian("Maya");      viewQueue();      cout << "Jumlah antrian = " << countQueue() << endl;      dequeueAntrian();      viewQueue();      cout << "Jumlah antrian = " << countQueue() << endl;      clearQueue();      viewQueue();      cout << "Jumlah antrian = " << countQueue() << endl;      return 0;  } |

Screenshot Output

Deskripsi Program

1. **Tugas**

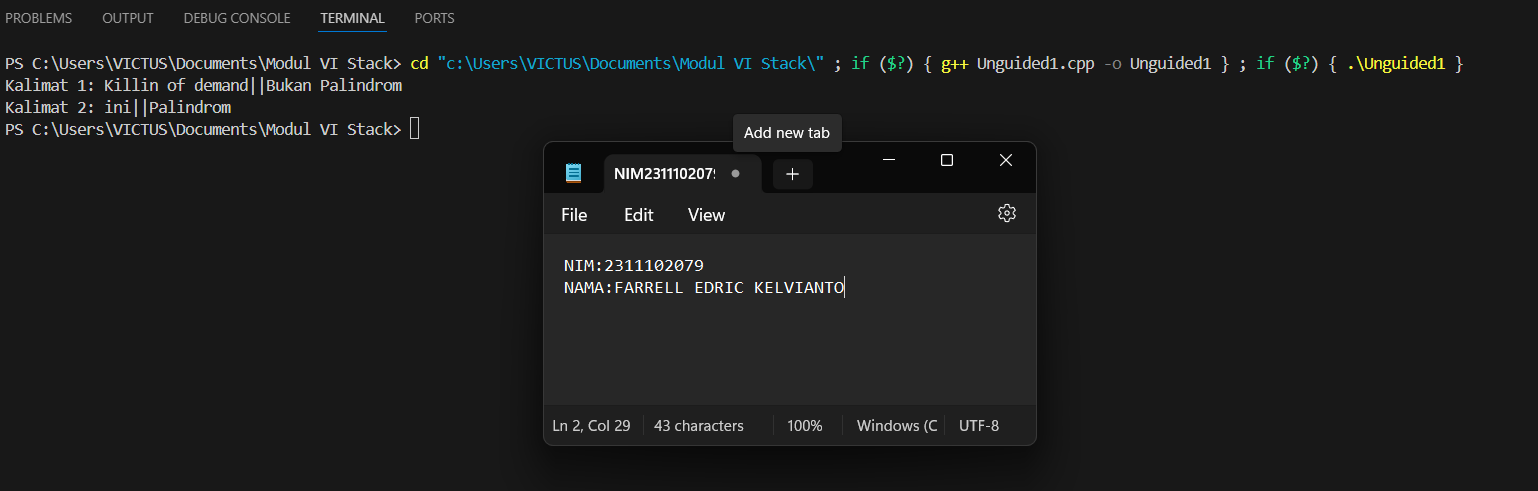
**Unguided 1**

|  |
| --- |
| #include <iostream>  #include <string>  using namespace std;  string arrayPalindrom[5];  int maksimal = 5, top = 0;  // Mengecek apakah stack penuh  bool isFull()  {      return (top == maksimal);  }  // Mengecek apakah stack kosong  bool isEmpty()  {      return (top == 0);  }  // Menambahkan elemen ke stack  void pushPalindrom(string Palindrom)  {      if (isFull())      {          cout << "Data Telah Penuh" << endl;      }      else      {          arrayPalindrom[top] = Palindrom;          top++;      }  }  // Menghapus elemen dari stack  void popPalindrom()  {      if (isEmpty())      {          cout << "Tidak ada data yang dihapus" << endl;      }      else      {          arrayPalindrom[top - 1] = "";          top--;      }  }  // Melihat elemen pada posisi tertentu dari stack  void peekPalindrom(int posisi)  {      if (isEmpty())      {          cout << "Suku Kata Palindrom Tidak Ditemukan" << endl;      }      else      {          int index = top;          for (int i = 1; i <= posisi; i++)          {              index--;          }          cout << "Posisi ke " << posisi << " adalah " << arrayPalindrom[index] << endl;      }  }  // Menghitung jumlah elemen dalam stack  int countstack()  {      return top;  }  // Mengubah elemen pada posisi tertentu dari stack  void changePalindrome(int posisi, string palindrome)  {      if (posisi > top)      {          cout << "Posisi melebihi jangkauan " << endl;      }      else      {          int index = top;          for (int i = 1; i <= posisi; i++)          {              index--;          }          arrayPalindrom[index] = palindrome;      }  }  // Menghapus semua elemen dalam stack  void destroyArraybuku()  {      for (int i = top; i >= 0; i--)      {          arrayPalindrom[i] = "";      }      top = 0;  }  bool isPalindrome(const string& word)  {      int left = 0;      int right = word.length() - 1;      while (left < right)      {          if (word[left] != word[right])          {              return false;          }          left++;          right--;      }      return true;  }  bool Palindrome(const string& sentence)  {      string pembersihKata;        for (char c : sentence)      {          if (isalpha(c))          {              pembersihKata += tolower(c);          }      }      return isPalindrome(pembersihKata);  }  int main()  {        string kalimat1 = "Killin of demand";      string kalimat2 = "ini";        cout << "Kalimat 1: " << kalimat1 << "||"<< (Palindrome(kalimat1) ? "Palindrom" : "Bukan Palindrom") << endl;      cout << "Kalimat 2: " << kalimat2 << "||" << (Palindrome(kalimat2) ? "Palindrom" : "Bukan Palindrom") << endl;      return 0;  } |

Screenshot Output

Soal No. 1 menentukan apakah kalimat tersebut yang diinputkan

dalam program stack adalah palindrom/tidak:



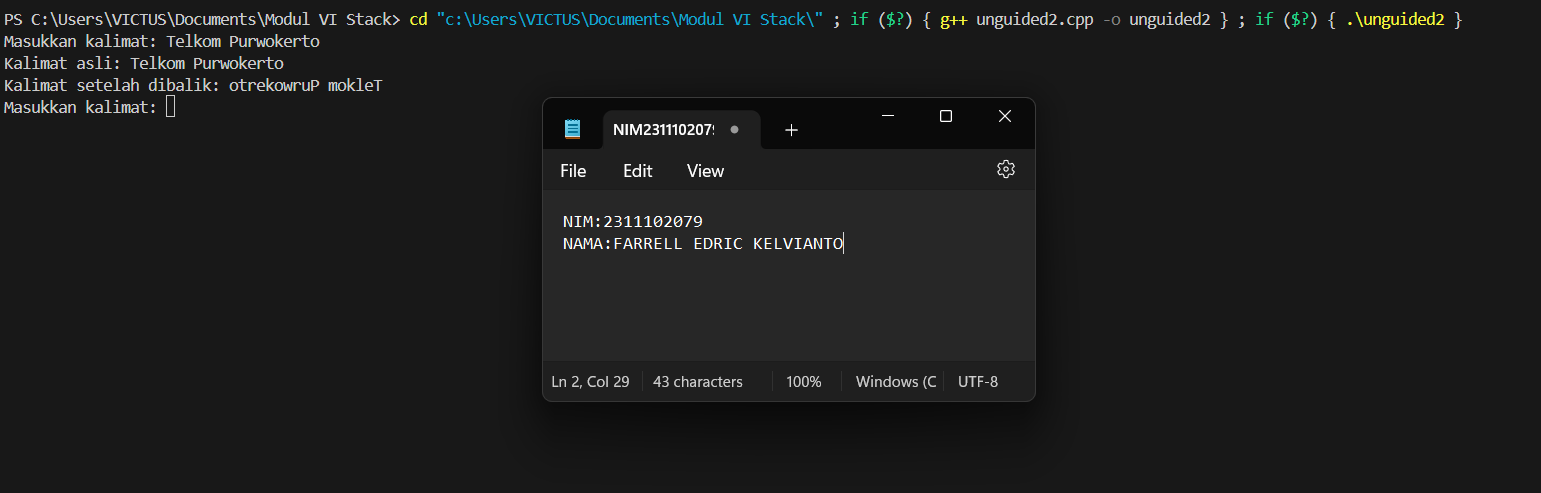
Deskripsi Program

Cara kerja program ini adalah seperti program pada umumnya sebenarnya hanya disini memaksakan untuk melakukan implementasi stack dari beberapa proses stacknya disini yang menjadi stacknya itu adalah tumpukan katanya. Dan disini saya akan jelaskan bagaimana setiap operasi di atas:

* isFull( ) berfungsi untuk memberikan batas maksimal dan jika ujung sudah maksimal maka akan terhenti.
* isEmpty ( ) ini berfungsi untuk mengindikasikan apakah stack masih kosong dan jika kosong maka dia akan mereturn.
* pushPalindrom ( ) untuk menumpuk kata – kata yang diindikasikan sebagai kata palindrom atau bukan.
* popPalindrom ( ) ini berfungsi untuk memnghapus data paling atas.
* peekPalindrom( ) berfungsi untuk mencari kata – kata yang ditumpuk tadi dengan posisi tertentu.
* ChangePalindrom ( ) ini berfungsi untuk mengubah kata palindrom sebelumnya menjadi kata baru.
* DestroyArrayBuku ( ) ini berfungsi untuk menghapus semua data didalam array.
* IsPalindrom ( ) berfungsi untuk mengecek apakah mengandung palindrome atau bukan dengan cara melihat dari kekiri ke kanan.
* Palindrome( ) ini berfungsi untuk memeriksa dari karakter ke karakter dan dipaling bawah itu fungsinya untuk mengabaikan seperti huruf kapital simbol ataupun angka.

**Unguided 2**

|  |
| --- |
| #include <iostream>  #include <cstring>  using namespace std;  const int MAX = 100;  void pembalikKata(char\* sentence) {      char stack[MAX];      int top = -1;      int length = strlen(sentence);        for (int i = 0; i < length; ++i) {          if (top >= (MAX - 1)) {              cout << "Stack Overflow\n";              return;          } else {              stack[++top] = sentence[i];          }      }      for (int i = 0; i < length; ++i) {          if (top < 0) {              cout << "Stack Underflow\n";              return;          } else {              sentence[i] = stack[top--];          }      }  }  int main() {      while (true){      char kalimat[MAX];        cout << "Masukkan kalimat: ";      cin.getline(kalimat, sizeof(kalimat));      int word\_count = 1;      char temp[MAX];      strcpy(temp, kalimat);      char\* token = strtok(temp, " ");      while (token != nullptr) {          ++word\_count;          token = strtok(nullptr, " ");      }      if (word\_count <= 3 && word\_count <= 2) {          cerr << "Kalimat harus mengandung minimal 3 kata." << endl;          return 1;      }      cout << "Kalimat asli: " << kalimat << endl;      pembalikKata(kalimat);      cout << "Kalimat setelah dibalik: " << kalimat << endl;      }      return 0;  } |

**Screenshot Output**

**Deskripsi Program**

Nah untuk program yang ini berbeda dengan sebelumnya tetapi untuk implementasi stacknya tetap masuk. Disini terdapat sebuah 1 prosedur yaitu untuk implementasi stacknya dan beserta membaca length dari kata yang akan dibalik. Untuk membatasi maksnya disini menggunakan char Max yang bersikan 100 untuk batasnya disertai const supaya tidak terjadi manipulasi data. Disini yang sebagai pembalik katanya terdapat pada fungsi mainnya mneggunakan sizeof untuk mengukur size kata dari inputan lalu mengindikasikan melalui char dan strtok ini berfungsi untuk membagi – bagi kata yang akan dibuat untuk reverse katanya, nah disini saya membatasi 3 kata seperti pada soalnya.

**Kesimpulan**

Kesimpulannya adalah stack merupakan tumpukan seperti dengan analogi bola yang

ditumpukan kedalam ranjang bola dan pasti setiap ranjang memiliki kapasitas untuk diisi nah

sama seperti stack jika stack telah mencapai maksimum maka dia tidak bisa diisi kembali ini

berguna untuk mencegah alokasi yang tidak terkendali. Kesimpulan di nomor 1 dan 2 adalah

walaupun temanya beda tetapi untuk susunan program untuk menggunakan stacknya tetap sama.

**Refrensi**

Asisten Praktikum. “Modul 6 Stack”. Learning Management System. 2024