**Laporan Tugas Besar**

**Algoritma & Struktur Data**

**“SI Pembayaran Listrik”**

****

Oleh:

Faathir Akbar Nugroho

4522210033

ASD (A)

Dosen:

**Sri Rezeki Candra Nursari, Dra., M.Kom**

**S1-Teknik Informatika**

**Fakultas Teknik Universitas Pancasila**

**2022/2023**

# KATA PENGANTAR

Puji syukur penulis panjatkan ke hadirat Tuhan yang maha esa karena dengan izin-Nya, penulis dapat menyelesaikan laporan praktikum ini dengan baik dan benar serta dapat terselesaikan tepat sesuai dengan waktunya.

Laporan praktikum yang bertemakan “SI Pembayaran Listrik” ini disusun untuk memenuhi tugas besar dari mata kuliah Algoritma dan Struktur Data semester 2 Teknik Informatika Universitas Pancasila tahun 2022/2023.

Pada kesempatan ini, penulis ingin mengucapkan terima kasih kepada semua pihak yang telah berkontribusi dan memberikan semangat, bantuan, dan motivasi kepada penulis dalam pembuatan laporan praktikum ini.

Penulis pun menyadari dalam laporan praktikum ini masih terdapat banyak kesalahan, kekurangan, dan masih jauh dari kesempurnaan. Oleh karena itu, segala kritik serta saran yang bersifat membangun sangat penulis harapakan untuk penyempurnaan laporan praktikum mendatang. Semoga laporan praktikum ini dapat bermanfaat bagi penulis maupun pembaca.

Jakarta, 28 Juni 2023

Penulis

**DAFTAR ISI**

KATA PENGANTAR 1

DAFTAR ISI 2

BAB 1 PENDAHULUAN 3

1.1 Latar Belakang 3

1.2 Tujuan 3

1.3 Pembahasan Teori 4

1.3.1 Struktur 4

1.3.2 Pointer 4

1.3.3 ADT SDL Linear Single Linked List 5

1.3.4 SDL Stack 5

1.3.5 Bubble Sort Ascending 6

1.3.6 Sequential Search 6

1.4 Tugas Praktikum 7

BAB 2 TUGAS BESAR 38

2.1 Pembahasan Tugas Besar 38

2.1.1 Pseudocode 38

2.1.2 Algoritma 41

2.1.3 Program (Source Code) 44

2.1.4 Command Prompt Capture 52

BAB 3 PENUTUP 54

3.1 Kesimpulan 54

3.2 Kritik dan Saran 54

DAFTAR PUSTAKA 55

**BAB 1**

**PENDAHULUAN**

## Latar Belakang

Pada era modern seperti saat ini yang sudah memasuki revolusi industri 4.0 diiringi dengan kemajuan dan perkembangan zaman, manusia dituntut untuk mengikuti perkembangan ilmu pengetahuan serta teknologi. Komputer merupakan alat yang sangat populer untuk membantu dan menyelesaikan suatu pekerjaan dengan cepat pada zaman sekarang, dilengkapi dengan perangkat lunak dan perangkat keras yang semakin banyak jenisnya serta fungsinya. Mayoritas dari kehidupan kita sehari-hari pada era globalisasi seperti ini yaitu tidak bisa dijauhkan dari teknologi dan informasi.

Adanya Bahasa pemrograman komputer C++ yang sudah mendorong penulis untuk membuat program SI Pembayaran Listrik secara sederhana. Karena hampir dari seluruh masyarakat sudah menggunakan smartphone atau komputer, yang dimana salah satu penggunaannya adalah untuk membayar tagihan listrik atau membeli token listrik. Maka dari itu penulis membuat program SI Pembayaran Listrik secara sederhana agar mempermudah masyarakat dalam transaksi perihal kebutuhan listrik.

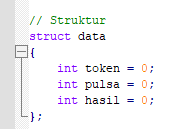
* 1. **Tujuan**  
     Adapun tujuan disusunnya laporan tugas besar ini yaitu:

1. Untuk memenuhi tugas besar mata kuliah Algoritma dan Struktur Data pada semester 2 tahun ajaran 2022/2023.
2. Untuk memberikan pemahaman terhadap materi-materi yang sudah di pelajari dengan cara mengaplikasikannya menjadi suatu program yang terdapat semua materi tersebut.
3. Isi dari laporan tugas besar ini bertujuaan untuk mempermudah masyarakat dalam membayar Tagihan Listrik tanpa harus datang secara langsung.

## Pembahasan Teori

## Struktur

Struktur merujuk pada cara suatu objek, sistem, atau entitas terorganisir dan diatur. Ini melibatkan pengaturan komponen-komponen atau elemen-elemen menjadi sebuah kerangka yang memiliki pola atau tata letak tertentu. Struktur menentukan bagaimana komponen-komponen tersebut saling terhubung, berinteraksi, dan bekerja bersama untuk mencapai tujuan atau fungsi yang diinginkan.



Gambar 1 contoh Struktur dalam program tugas besar

* + 1. **Pointer**

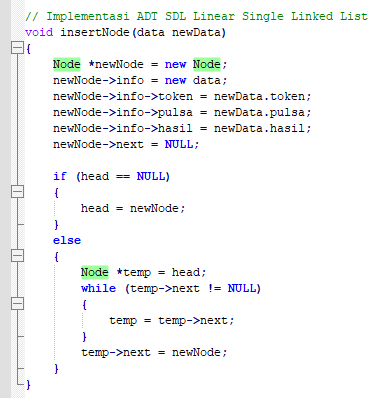
Pointer adalah sebuah variabel yang berisi alamat memori dari suatu nilai atau objek dalam bahasa pemrograman. Dengan menggunakan pointer, kita dapat mengakses dan memanipulasi nilai atau objek yang disimpan dalam alamat memori tersebut. Pointer memungkinkan kita untuk melakukan referensi atau referensi terhadap data atau objek yang lebih besar atau kompleks secara efisien.



Gambar 2 contoh Pointer dalam program tugas besar

* + 1. **ADT-SDL-Linear Single Linked List**

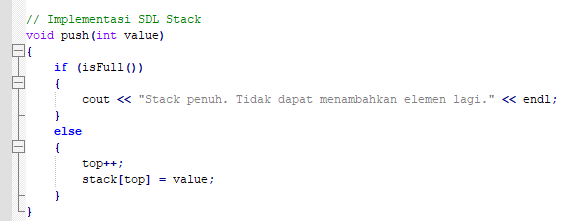
Linear Single Linked List adalah struktur data yang terdiri dari serangkaian simpul (node) yang saling terhubung satu demi satu melalui tautan (link) berupa alamat memori. Setiap simpul menyimpan data dan tautan ke simpul berikutnya. ADT-SDL-Linear Single Linked List memungkinkan pengguna untuk mengimplementasikan operasi-operasi dasar pada struktur data ini, seperti penambahan (insertion), penghapusan (deletion), dan pencarian (searching) elemen-elemen dalam daftar tautan tunggal secara efisien dalam pengembangan aplikasi yang menggunakan SDL.



Gambar 3 contoh ADT-SDL-Linear Single Linked List dalam program tugas besar

### SDL-Stack

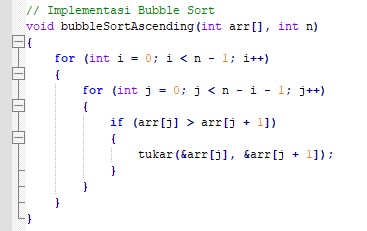
SDL Stack merujuk pada SDL (Simple DirectMedia Layer) Stack, yang merupakan sebuah library atau pustaka perangkat lunak yang digunakan dalam pengembangan permainan dan aplikasi multimedia. SDL Stack menyediakan fungsi dan alat yang memudahkan pengguna untuk mengakses dan mengontrol elemen-elemen grafis, suara, jendela, dan input pada aplikasi yang dikembangkan menggunakan SDL. SDL Stack membantu pengembang dalam mengelola tumpukan (stack) berbagai fungsi dan fitur yang dibutuhkan dalam pengembangan aplikasi multimedia dengan SDL.



Gambar 4 contoh SDL-Stack dalam program tugas besar

* + 1. **Bubble Sort-Ascending**

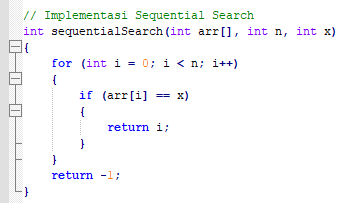
Bubble sort ascending merupakan salah satu algoritma pengurutan sederhana yang digunakan untuk mengurutkan elemen-elemen dalam suatu rangkaian data secara terurut meningkat (ascending). Algoritma ini bekerja dengan membandingkan pasangan-pasangan elemen secara berurutan dan menukar posisinya jika ditemukan elemen yang lebih besar di depannya. Selama iterasi, elemen terbesar akan "menggelembung" ke posisi yang tepat, sehingga pada akhirnya, urutan yang dihasilkan akan berupa rangkaian data dengan elemen-elemen yang terurut secara terurut meningkat.



Gambar 5 contoh Bubble Sort-Ascending dalam program tugas besar

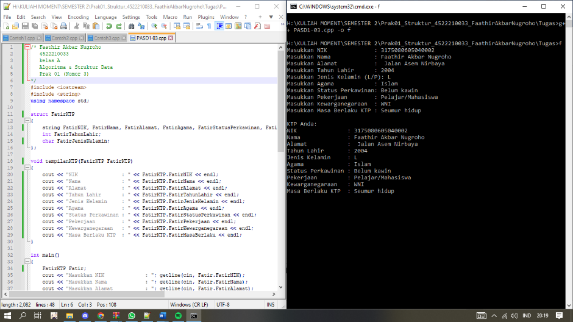
* + 1. **Sequential Search**

Sequential search, juga dikenal sebagai linear search, adalah sebuah metode pencarian dalam struktur data atau rangkaian data yang melakukan pencarian secara berurutan dari awal hingga akhir elemen.

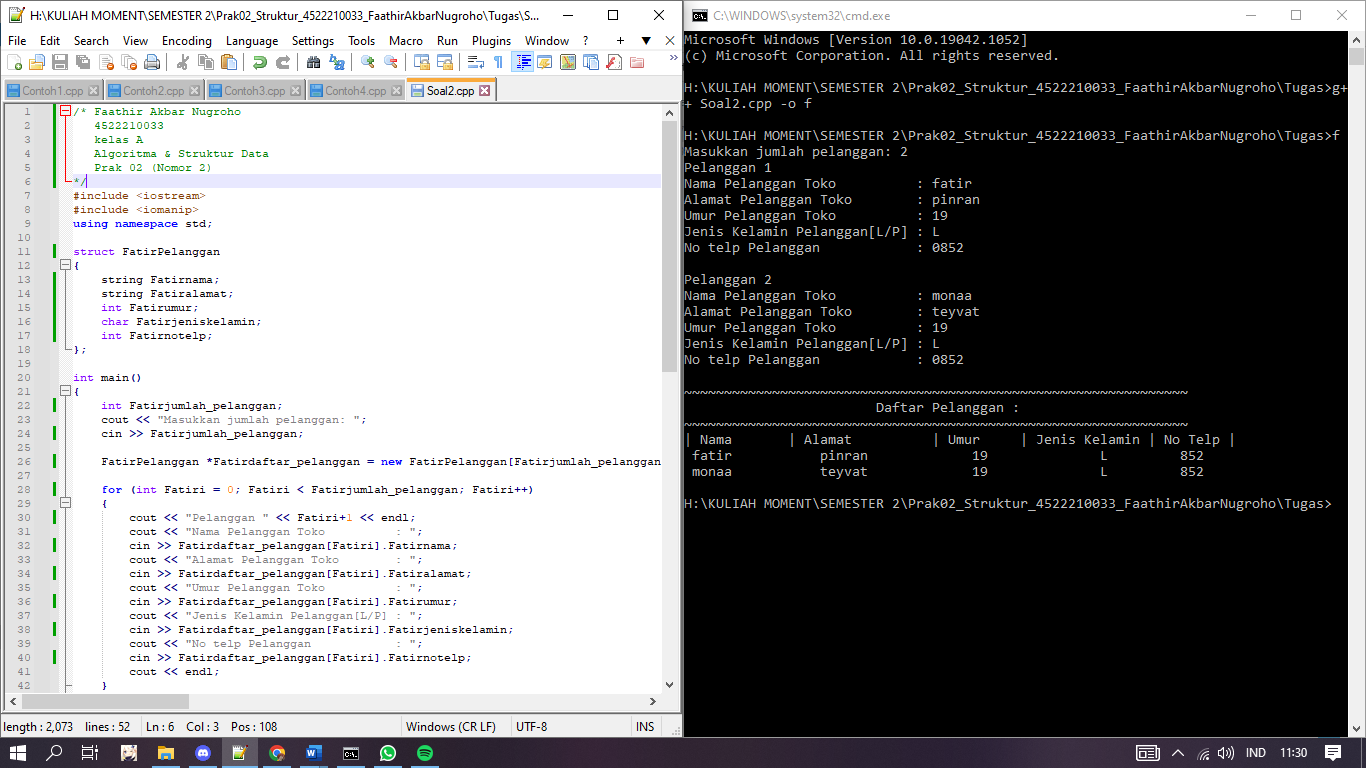
  
Gambar 6 contoh Sequential Search dalam program tugas besar

* 1. **Tugas Praktikum**

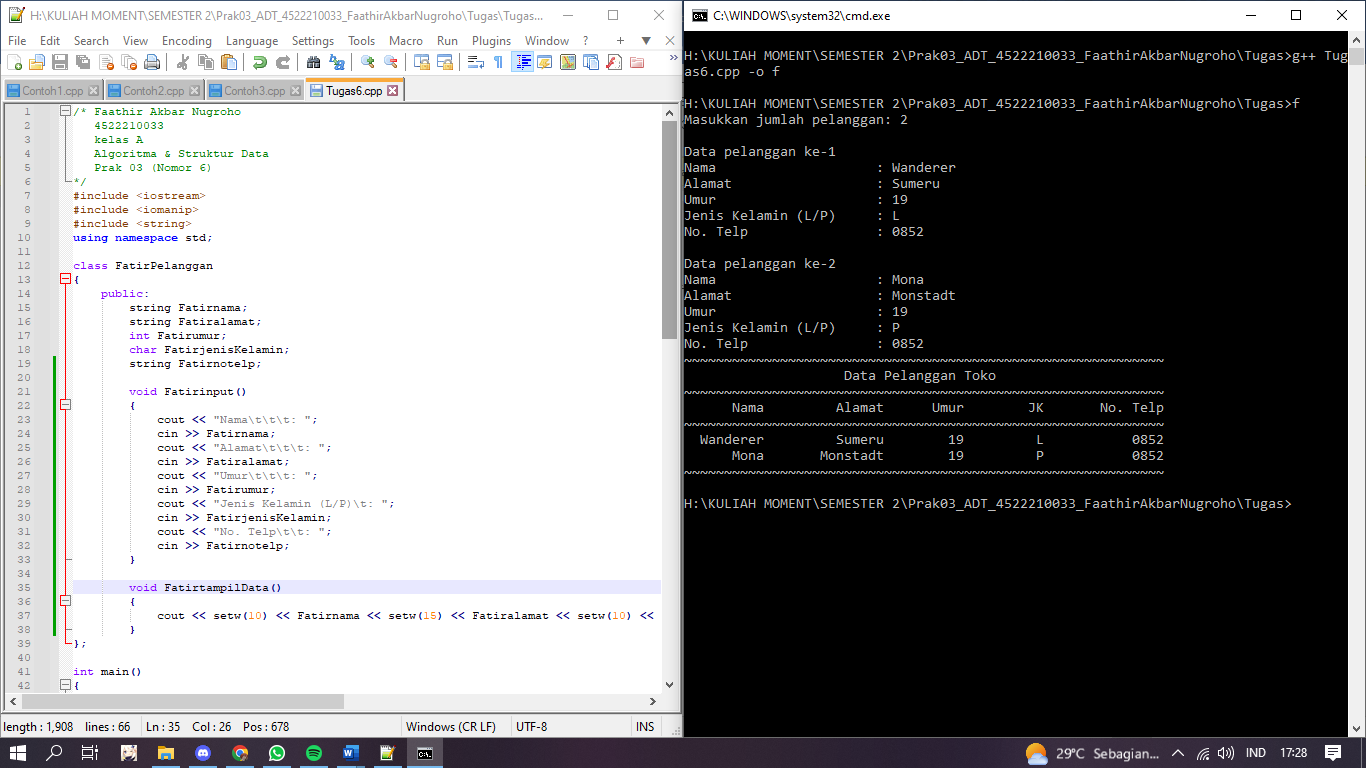
|  |
| --- |
| **Praktikum 1** |
| **Pseudocode (Nomor 3)**  **Kamus/Deklarasi Variabel Function**  **-**  **Kamus/Deklarasi Variabel**  FatirNIK, FatirNama, FatirAlamat, FatirAgama, FatirStatusPerkawinan, FatirPekerjaan, FatirKewarganegaraan, FatirMasaBerlaku = string  FatirJenisKelamin = char  FatirTahunLahir = int  **Algoritma/Deskripsi Function tampilanKTP (FatirKTP FatirKTP)**  print(FatirKTP.FatirNIK)  print(FatirKTP.FatirNama)  print(FatirKTP.FatirAlamat)  print(FatirKTP.FatirTahunLahir)  print(FatirKTP.FatirJenisKelamin)  print(FatirKTP.FatirAgama)  print(FatirKTP.FatirStatusPerkawinan)  print(FatirKTP.FatirPekerjaan)  print(FatirKTP.FatirKewarganegaraan)  print(FatirKTP.FatirMasaBerlaku)  **Algoritma/Deskripsi**  struct FatirKTP{FatirNIK, FatirNama, FatirAlamat, FatirAgama, FatirStatusPerkawinan, FatirPekerjaan, FatirKewarganegaraan, FatirMasaBerlaku, FatirTahunLahir, FatirJenisKelamin}  FatirKTP Fatir  input(Fatir.FatirNIK)  input(Fatir.FatirNama)  input(Fatir.FatirAlamat)  input(Fatir.FatirTahunLahir)  input(Fatir.FatirJenisKelamin)  input(Fatir.FatirAgama)  input(Fatir.FatirStatusPerkawinan)  input(Fatir.FatirPekerjaan)  input(Fatir.FatirKewarganegaraan)  input(Fatir.FatirMasaBerlaku)  tampilanKTP(Fatir) |
| **Algoritma/Bahasa Natural (Nomor 3)**   1. Membuat fungsi tampilanKTP(FatirKTP FatirKTP) 2. Mencetak/menampilkan pengaksesan anggota struktur (FatirKTP.FatirNIK) 3. Mencetak/menampilkan pengaksesan anggota struktur (FatirKTP.FatirNama) 4. Mencetak/menampilkan pengaksesan anggota struktur (FatirKTP.FatirAlamat) 5. Mencetak/menampilkan pengaksesan anggota struktur (FatirKTP.FatirTahunLahir) 6. Mencetak/menampilkan pengaksesan anggota struktur (FatirKTP.FatirJenisKelamin) 7. Mencetak/menampilkan pengaksesan anggota struktur (FatirKTP.FatirAgama) 8. Mencetak/menampilkan pengaksesan anggota struktur (FatirKTP.FatirStatusPerkawinan) 9. Mencetak/menampilkan pengaksesan anggota struktur (FatirKTP.FatirPekerjaan) 10. Mencetak/menampilkan pengaksesan anggota struktur (FatirKTP.FatirKewarganegaraan) 11. Mencetak/menampilkan pengaksesan anggota struktur (FatirKTP.FatirMasaBerlaku) 12. Deklarasi struktur (struct FatirKTP{FatirNIK, FatirNama, FatirAlamat, FatirAgama, FatirStatusPerkawinan, FatirPekerjaan, FatirJenisKelamin FatirKewarganegaraan,   FatirMasaBerlaku, FatirTahunLahir})   1. Mendefinisikan struktur {FatirKTP Fatir} 2. Memasukkan isi/nilai pengaksesan anggota struktur (FatirKTP.FatirNIK) 3. Memasukkan isi/nilai pengaksesan anggota struktur (FatirKTP.FatirNama) 4. Memasukkan isi/nilai pengaksesan anggota struktur (FatirKTP.FatirAlamat) 5. Memasukkan isi/nilai pengaksesan anggota struktur (FatirKTP.FatirTahunLahir) 6. Memasukkan isi/nilai pengaksesan anggota struktur (FatirKTP.FatirJenisKelamin) 7. Memasukkan isi/nilai pengaksesan anggota struktur (FatirKTP.FatirAgama) 8. Memasukkan isi/nilai pengaksesan anggota struktur (FatirKTP.FatirStatusPerkawinan) 9. Memasukkan isi/nilai pengaksesan anggota struktur (FatirKTP.FatirPekerjaan) 10. Memasukkan isi/nilai pengaksesan anggota struktur (FatirKTP.FatirKewarganegaraan) 11. Memasukkan isi/nilai pengaksesan anggota struktur (FatirKTP.FatirMasaBerlaku) 12. Memanggil fungsi tampilanKTP(Fatir) 13. Selesai |



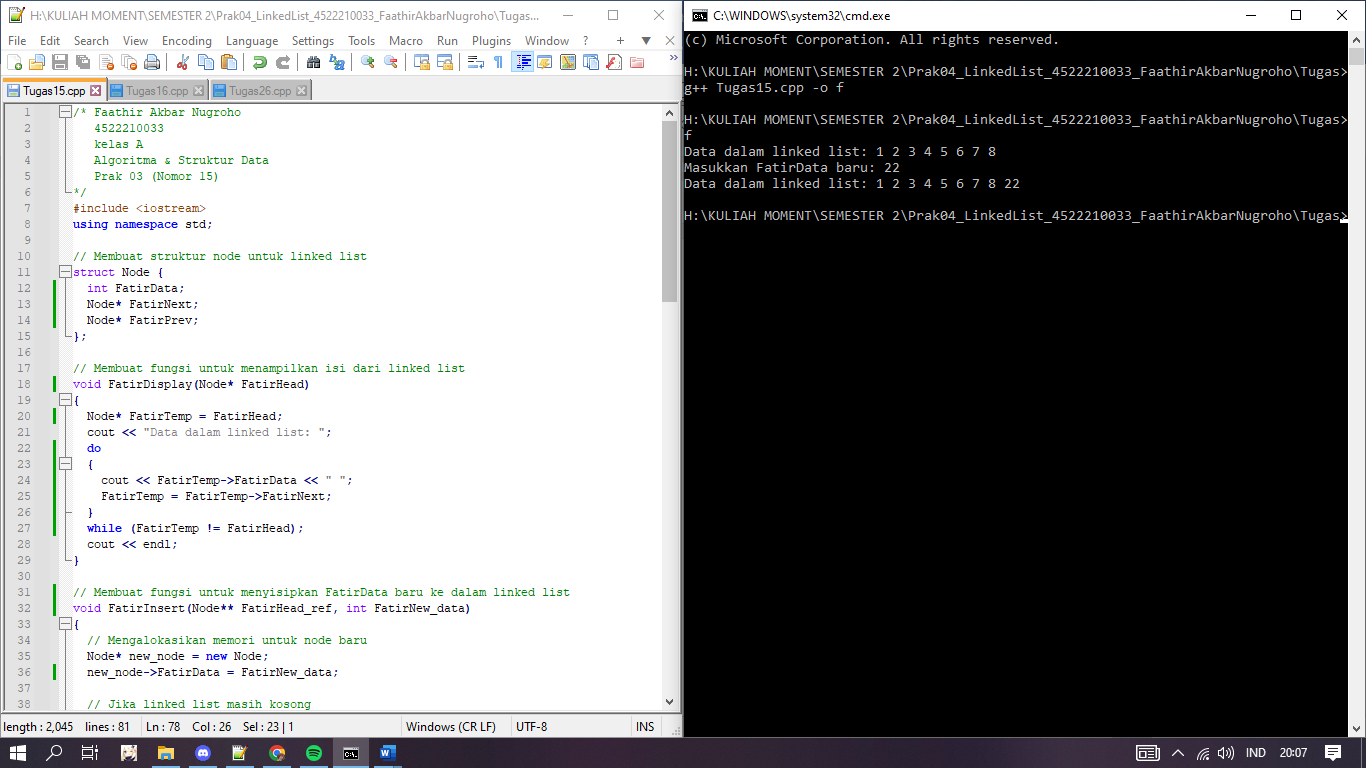
|  |
| --- |
| **Praktikum 2** |
| **Pseudocode (Nomor 2)**  **Kamus/Deklarasi Variabel**  Fatirnama, Fatiralamat = char  Fatirumur, Fatirnotelp = int  Fatirjeniskelamin = char  **Algoritma/Deskripsi**  struct FatirPelanggan {Fatirnama, Fatiralamat, Fatirumur, Fatirjeniskelamin, Fatirnotelp}  input(Fatirjumlah\_pelanggan)  FatirPelanggan \*Fatirdaftar\_pelanggan = new FatirPelanggan[Fatirjumlah\_pelanggan]  for (int Fatiri = 0; Fatiri < Fatirjumlah\_pelanggan; Fatiri++)  print(Fatiri+1)  input(Fatirdaftar\_pelanggan[Fatiri].Fatirnama)  input(Fatirdaftar\_pelanggan[Fatiri].Fatiralamat)  input(Fatirdaftar\_pelanggan[Fatiri].Fatirumur)  input(Fatirdaftar\_pelanggan[Fatiri].Fatirjeniskelamin)  input(Fatirdaftar\_pelanggan[Fatiri].Fatirnotelp)  endfor  for (int Fatiri = 0; Fatiri < Fatirjumlah\_pelanggan; Fatiri++)  print(Fatirdaftar\_pelanggan[Fatiri].Fatirnama, Fatirdaftar\_pelanggan[Fatiri].Fatiralamat, Fatirdaftar\_pelanggan[Fatiri].Fatirumur, Fatirdaftar\_pelanggan[Fatiri].Fatirjeniskelamin, Fatirdaftar\_pelanggan[Fatiri].Fatirnotelp)  endfor |
| **Algoritma/Bahasa Natural (Nomor 2)**   1. Deklarasikan struktur (FatirPelanggan {Fatirnama, Fatiralamat, Fatirumur, Fatirjeniskelamin, Fatirnotelp}) 2. Memasukkan isi/nilai variabel Fatirjumlah\_pelanggan 3. Mendefinisikan struktur {FatirPelanggan \*Fatirdaftar\_pelanggan = new FatirPelanggan[Fatirjumlah\_pelanggan]} 4. Fatiri = 0 5. Selama (Fatiri< Fatirjumlah\_pelanggan), maka kerjakan baris 6 s.d 12, kalau tidak kerjakan baris 13 6. Mencetak isi/nilai variabel (Fatiri+1) 7. Memasukkan isi/nilai pengaksesan anggota struktur (Fatirdaftar\_pelanggan[Fatiri].Fatirnama) 8. Memasukkan isi/nilai pengaksesan anggota struktur Fatirdaftar\_pelanggan[Fatiri].Fatiralamat 9. Memasukkan isi/nilai pengaksesan anggota struktur Fatirdaftar\_pelanggan[Fatiri].Fatirumur 10. Memasukkan isi/nilai pengaksesan anggota struktur Fatirdaftar\_pelanggan[Fatiri].Fatirjeniskelamin 11. Memasukkan isi/nilai pengaksesan anggota struktur Fatirdaftar\_pelanggan[Fatiri].Fatirnotelp 12. Fatiri++ 13. Fatiri = 0 14. Selama (Fatiri < Fatirjumlah\_pelanggan), maka kerjakan baris 15 s.d 16, kalau tidak kerjakan baris 17 15. Mencetak/menampilkan pengaksesan anggota struktur (Fatirdaftar\_pelanggan[Fatiri].Fatirnama, Fatirdaftar\_pelanggan[Fatiri].Fatiralamat, Fatirdaftar\_pelanggan[Fatiri].Fatirumur, Fatirdaftar\_pelanggan[Fatiri].Fatirjeniskelamin, Fatirdaftar\_pelanggan[Fatiri].Fatirnotelp) 16. Fatiri++ 17. Selesai |



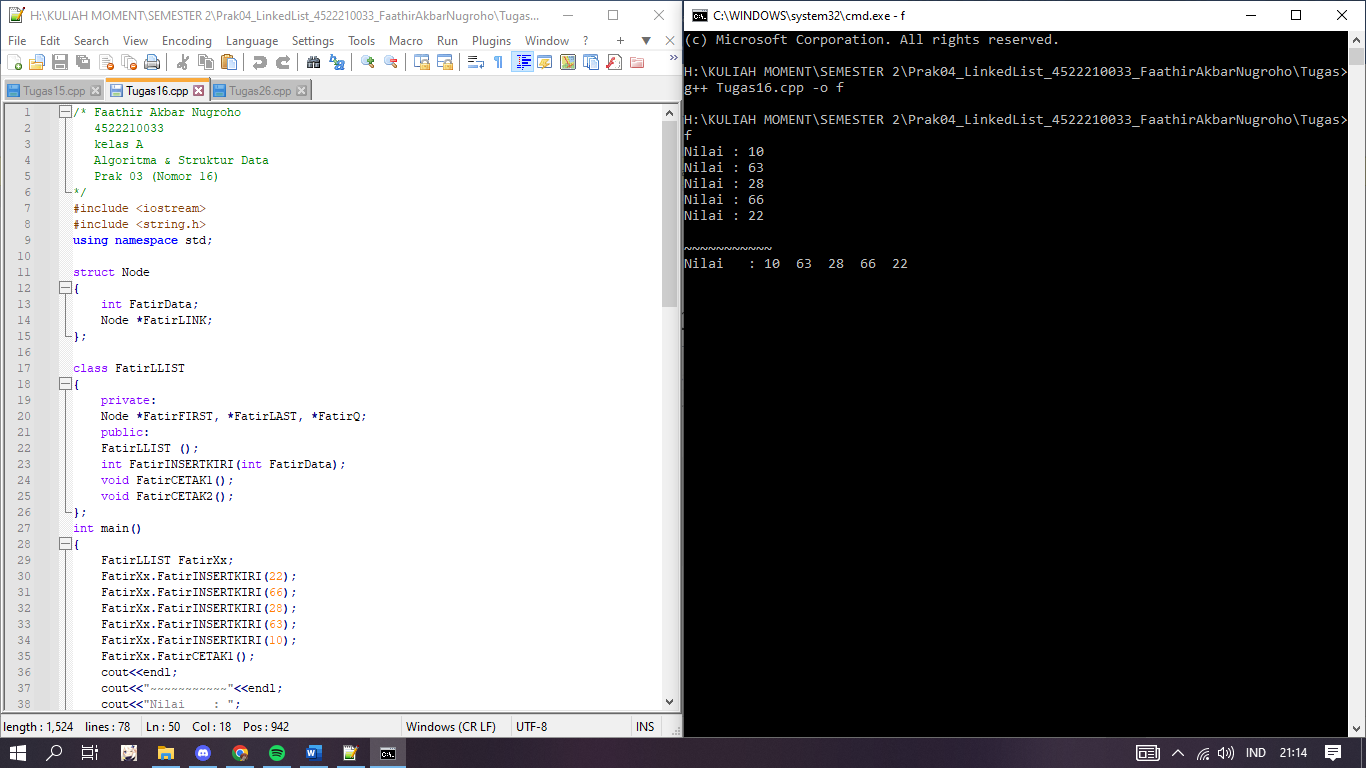
|  |
| --- |
| **Praktikum 3** |
| **Pseudocode (Nomor 6)**  **Kamus/Deklarasi Variabel Function Fatirinput()**  -  **Algoritma/Deskripsi Function Fatirinput()**  input(Fatirnama)  input(Fatiralamat)  input(Fatirumur)  input(FatirjenisKelamin)  input(Fatirnotelp)  **Kamus/Deklarasi Variabel Function FatirtampilData()**  -  **Algoritma/Deskripsi Function FatirtampilData()**  print(Fatirnama, Fatiralamat, Fatirumur, FatirjenisKelamin, Fatirnotelp)  **Kamus/Deklarasi Variabel**  FatirjumlahPelanggan, Fatiri : int  **Algoritma/Deskripsi**  Public: class FatirPelanggan (Fatirnama, Fatiralamat, Fatirumur, FatirjenisKelamin, Fatirnotelp, Fatirinput(), FatirtampilData())  input(FatirjumlahPelanggan)  FatirPelanggan daftarPelanggan[FatirjumlahPelanggan]  for(Fatiri = 0; Fatiri < FatirjumlahPelanggan; Fatiri++)  print(Fatiri+1)  daftarPelanggan[Fatiri].Fatirinput()  endfor  for(Fatiri = 0; Fatiri < FatirjumlahPelanggan; Fatiri++)  daftarPelanggan[Fatiri].FatirtampilData()  endfor |
| **Algoritma/Bahasa Natural (Nomor 6)**   1. Membuat function Fatirinput() 2. Memasukkan isi/nilai pengaksesan anggota class Fatirnama 3. Memasukkan isi/nilai pengaksesan anggota class Fatiralamat 4. Memasukkan isi/nilai pengaksesan anggota class Fatirumur 5. Memasukkan isi/nilai pengaksesan anggota class FatirjenisKelamin 6. Memasukkan isi/nilai pengaksesan anggota class Fatirnotelp 7. Membuat function FatirtampilData() 8. Menampilkan isi/nilai pengaksesan anggota class (Fatirnama, Fatirumur, Fatiralamat, FatirjenisKelamin, Fatirnotelp) 9. Mendeklarasikan class (class FatirPelanggan (Fatirnama, Fatiralamat, Fatirumur, FatirjenisKelamin, Fatirnotelp, Fatirinput(), FatirtampilData()) dengan penentu akses secara public 10. Memasukkan isi/nilai variabel FatirjumlahPelanggan 11. Mendefinisikan class (FatirPelanggan daftarPelanggan[FatirjumlahPelanggan]) 12. Fatiri = 0 13. Selama (Fatiri<0), maka kerjakan baris 14 s.d 16, kalau tidak baris 17 14. Menampilkan isi/nilai variabel (Fatiri+1) 15. Memanggil hasil deklarasi class yaitu daftarPelanggan[Fatiri] dan function Fatirinput() 16. Fatiri++ 17. Fatiri = 0 18. Selama (Fatiri< FatirjumlahPelanggan), maka kerjakan baris 19 s.d 20, kalau tidak baris 21 19. Memanggil hasil deklarasi class yaitu daftarPelanggan[Fatiri] dan function FatirtampilData () 20. Fatiri++ 21. Selesai |



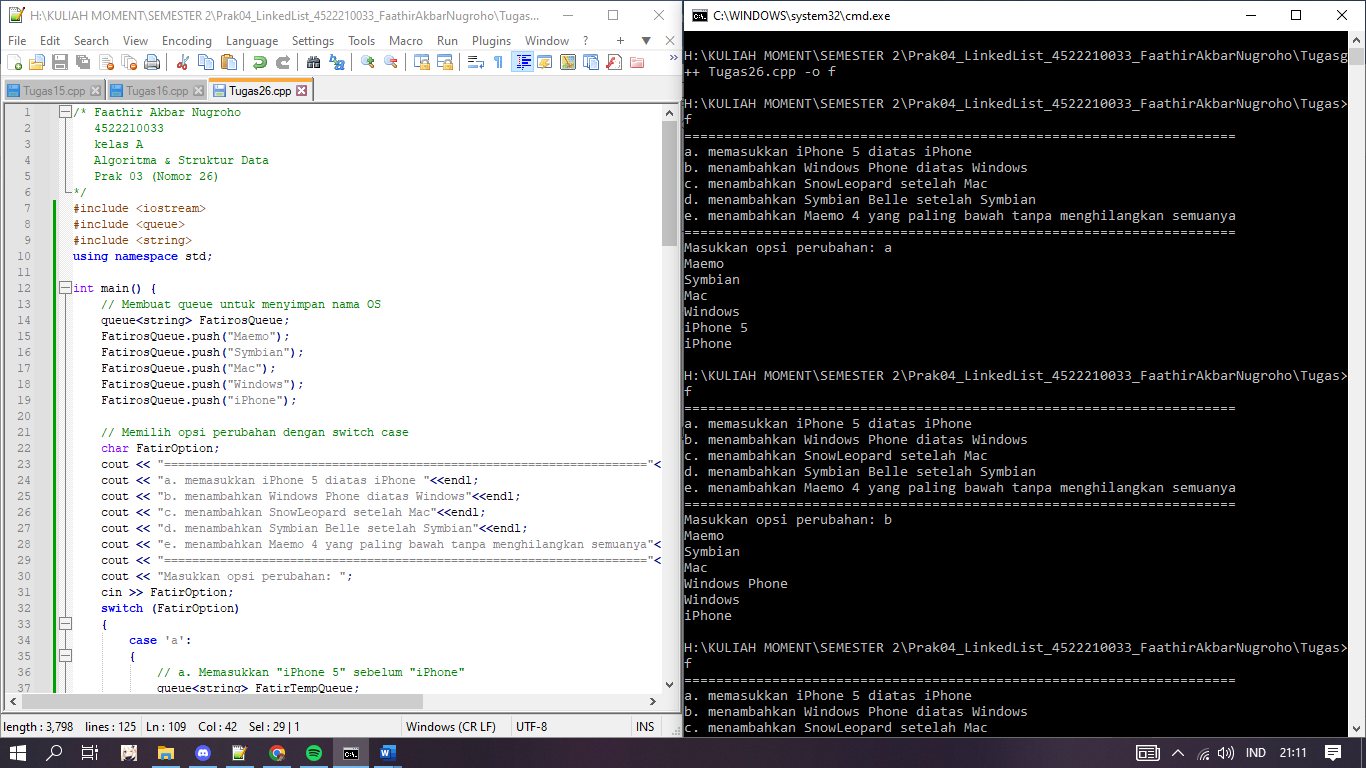
|  |
| --- |
| **Praktikum 4** |
| **Pseudocode (Nomor 15)**  **Kamus/Deklarasi Variabel Function FatirDisplay(Node\* FatirHead)**  -  **Algoritma/Deskripsi Function FatirDisplay(Node\* FatirHead)**  Node\* FatirTemp = FatirHead  do  print(FatirTemp->FatirData)  FatirTemp = FatirTemp->FatirNext  while (FatirTemp != FatirHead)  endwhile  **Kamus/Deklarasi Variabel Function** **FatirInsert(Node\*\* FatirHead\_ref, int FatirNew\_data)**  -  **Algoritma/Deskripsi Function FatirInsert(Node\*\* FatirHead\_ref, int FatirNew\_data)**  Node\* new\_node = new Node  new\_node->FatirData = FatirNew\_data  if (\*FatirHead\_ref == NULL)  new\_node->FatirNext = new\_node  new\_node->FatirPrev = new\_node  \*FatirHead\_ref = new\_node  endif  Node\* FatirLast = (\*FatirHead\_ref)->FatirPrev  new\_node->FatirNext = \*FatirHead\_ref  (\*FatirHead\_ref)->FatirPrev = new\_node  new\_node->FatirPrev = FatirLast  FatirLast->FatirNext = new\_node  **Kamus/Deklarasi Variabel**  FatirData, FatirNew\_data = int  **Algoritma/Deskripsi**  struct Node(FatirData, Node\* FatirNext, Node\* FatirPrev)  Node\* FatirHead = NULL  FatirInsert(&FatirHead, 1)  FatirInsert(&FatirHead, 2)  FatirInsert(&FatirHead, 3)  FatirInsert(&FatirHead, 4)  FatirInsert(&FatirHead, 5)  FatirInsert(&FatirHead, 6)  FatirInsert(&FatirHead, 7)  FatirInsert(&FatirHead, 8)  FatirDisplay(FatirHead)  input(FatirNew\_data)  FatirInsert(&FatirHead, FatirNew\_data)  FatirDisplay(FatirHead) |
| **Algoritma/Bahasa Natural (Tugas 15)**   1. Membuat function FatirDisplay(Node\* FatirHead) 2. Node\* FatirTemp = FatirHead 3. Mencetak hasil pengaksesan anggota struktur (FatirTemp->FatirData) 4. FatirTemp = FatirTemp->FatirNext 5. Selama (FatirTemp != FatirHead), maka kerjakan baris 3 s.d 4, kalau tidak kerjakan baris 6 6. Membuat function FatirInsert(Node\*\* FatirHead\_ref, int FatirNew\_data) 7. Node\* new\_node = new Node 8. new\_node->FatirData = FatirNew\_data 9. Selama (\*FatirHead\_ref == NULL), maka kerjakan baris 10 s.d 12, kalau tidak kerjakan baris 13 10. new\_node->FatirNext = new\_node 11. new\_node->FatirPrev = new\_node 12. \*FatirHead\_ref = new\_node 13. Node\* FatirLast = (\*FatirHead\_ref)->FatirPrev 14. new\_node->FatirNext = \*FatirHead\_ref 15. (\*FatirHead\_ref)->FatirPrev = new\_node 16. new\_node->FatirPrev = FatirLast 17. FatirLast->FatirNext = new\_node 18. Node\* FatirHead = NULL 19. FatirInsert(&FatirHead, 1) 20. FatirInsert(&FatirHead, 2) 21. FatirInsert(&FatirHead, 3) 22. FatirInsert(&FatirHead, 4) 23. FatirInsert(&FatirHead, 5) 24. FatirInsert(&FatirHead, 6) 25. FatirInsert(&FatirHead, 7) 26. FatirInsert(&FatirHead, 8) 27. Memanggil function FatirDisplay(FatirHead) 28. Memasukkan isi/nilai variabel FatirNew\_data 29. Memanggil function FatirInsert(&FatirHead, FatirNew\_data) 30. Memanggil function FatirDisplay(FatirHead) 31. Selesai |



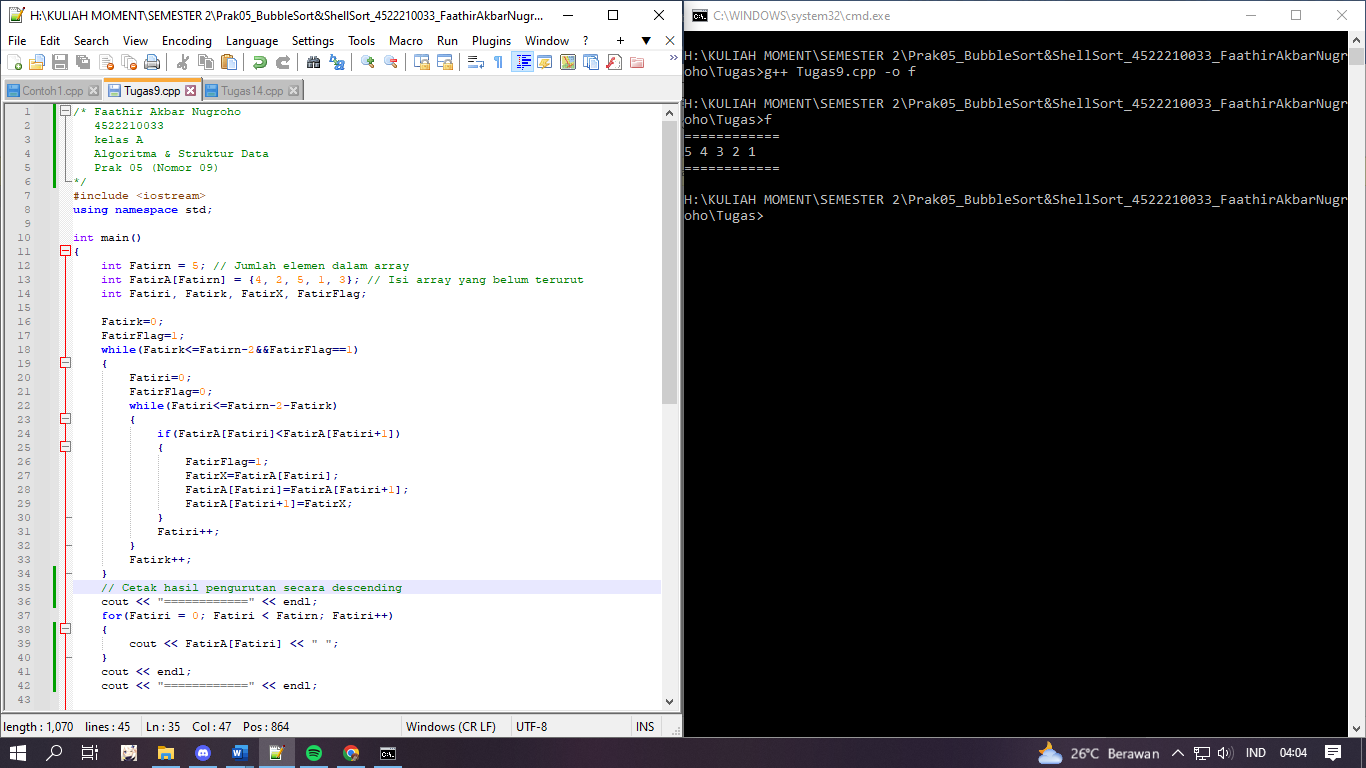
|  |
| --- |
| **Praktikum 4** |
| **Pseudocode (Nomor 16)**  **Kamus/Deklarasi Variabel Function FatirLLIST :: FatirCETAK1()**  -  **Algoritma/Deskripsi Function FatirLLIST :: FatirCETAK1()**  Node \*FatirQ  FatirQ = FatirFIRST  while (FatirQ != NULL)  print(FatirQ -> FatirData)  FatirQ = FatirQ -> FatirLINK  endwhile  **Kamus/Deklarasi Variabel Function FatirLLIST :: FatirCETAK2()**  -  **Algoritma/Deskripsi Function FatirLLIST :: FatirCETAK2()**  Node \*FatirQ  FatirQ = FatirFIRST  while (FatirQ != NULL)  print(FatirQ -> FatirData)  FatirQ = FatirQ -> FatirLINK  endwhile  **Kamus/Deklarasi Variabel int FatirLLIST :: FatirINSERTKIRI(int Fatirdt)**  Fatirdt = int  **Algoritma/Deskripsi int FatirLLIST :: FatirINSERTKIRI(int Fatirdt)**  Node \*FatirP  FatirP = new Node  FatirP -> FatirLINK = FatirFIRST  FatirP -> FatirData = Fatirdt  FatirFIRST = FatirP  **Kamus/Deklarasi Variabel**  FatirINSERTKIRI, FatirData = int  **Algoritma/Deskripsi**  Private: class FatirLLIST (Node \*FatirFIRST, \*FatirLAST, \*FatirQ)  Public: class FatirLLIST(FatirLLIST(), FatirINSERTKIRI(FatirData), FatirCETAK1(), FatirCETAK2())  FatirLLIST FatirXx  FatirXx.FatirINSERTKIRI(22)  FatirXx.FatirINSERTKIRI(66)  FatirXx.FatirINSERTKIRI(28)  FatirXx.FatirINSERTKIRI(63)  FatirXx.FatirINSERTKIRI(10)  FatirXx.FatirCETAK1()  FatirXx.FatirCETAK2() |
| **Algoritma/Bahasa Natural (Nomor 16)**   1. Membuat function FatirLLIST :: FatirCETAK1() 2. Membuat objek pada Node menjadi \*FatirQ 3. FatirQ = FatirFIRST 4. Selama (FatirQ != NULL), maka kerjakan baris 5 s.d 6, jika tidak kerjakan baris 7 5. Mencetak/menampilkan anggota struktur (FatirQ -> FatirData) 6. FatirQ = FatirQ -> FatirLINK 7. Membuat function FatirLLIST :: FatirCETAK2() 8. Membuat objek pada Node menjadi \*FatirQ 9. FatirQ = FatirFIRST 10. Selama (FatirQ != NULL), maka kerjakan baris 11 s.d 12, jika tidak kerjakan baris 13 11. Mencetak/menampilkan anggota struktur (FatirQ -> FatirData) 12. FatirQ = FatirQ -> FatirLINK 13. Membuat function FatirLLIST :: FatirINSERTKIRI(int Fatirdt) 14. Membuat objek pada Node menjadi \*FatirQ 15. FatirP = new Node 16. FatirP -> FatirLINK = FatirFIRST 17. FatirP -> FatirData = Fatirdt 18. FatirFIRST = FatirP 19. Mendefinisikan FatirLLIST menjadi FatirXx 20. FatirXx.FatirINSERTKIRI(22) 21. FatirXx.FatirINSERTKIRI(66) 22. FatirXx.FatirINSERTKIRI(28) 23. FatirXx.FatirINSERTKIRI(63) 24. FatirXx.FatirINSERTKIRI(10) 25. Memanggil function FatirXx.FatirCETAK1() 26. Memanggil function FatirXx.FatirCETAK2() 27. Selesai |



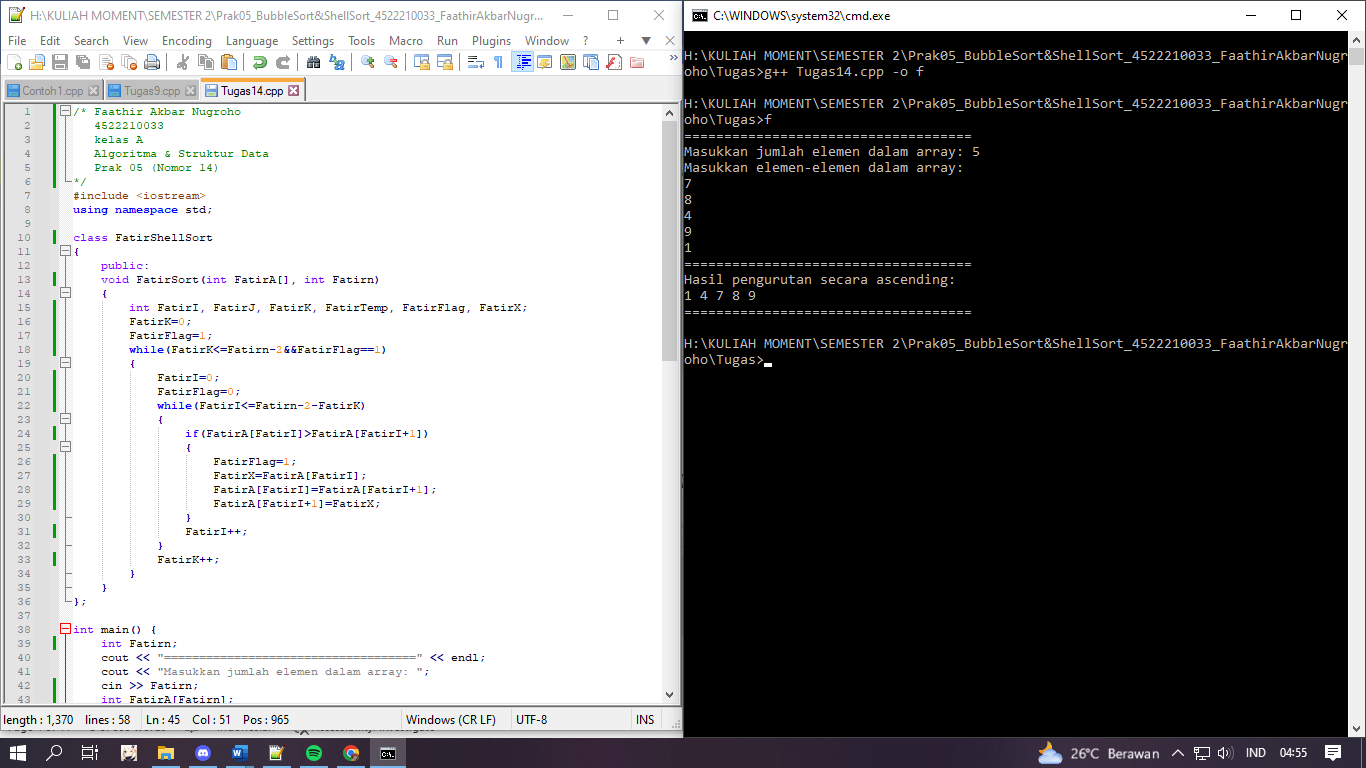
|  |
| --- |
| **Praktikum 4** |
| **Pseudocode (Nomor 26)**  **Kamus/Deklarasi Variabel**  FatirOption = char  **Algoritma/Deskripsi**  queue<string> FatirosQueue  FatirosQueue.push("Maemo")  FatirosQueue.push("Symbian")  FatirosQueue.push("Mac")  FatirosQueue.push("Windows")  FatirosQueue.push("iPhone")  input(FatirOption)  switch(FatirOption)  case 'a':  queue<string> FatirTempQueue  while (!FatirosQueue.empty())  if (FatirosQueue.front() == "iPhone")  FatirTempQueue.push("iPhone 5")  endif  FatirTempQueue.push(FatirosQueue.front())  FatirosQueue.pop()  endwhile  FatirosQueue = FatirTempQueue  break;  case 'b':  queue<string> FatirTempQueue  while (!FatirosQueue.empty())  if (FatirosQueue.front() == "Windows")  FatirTempQueue.push("Windows Phone")  endif  FatirTempQueue.push(FatirosQueue.front())  FatirosQueue.pop()  endwhile  FatirosQueue = FatirTempQueue  break;  case 'c':  queue<string> FatirTempQueue  while (!FatirosQueue.empty())  if (FatirosQueue.front() == "Mac")  FatirTempQueue.push(FatirosQueue.front())  FatirTempQueue.push("SnowLeopard")  else  FatirTempQueue.push(FatirosQueue.front())  endif  FatirosQueue.pop()  endwhile  FatirosQueue = FatirTempQueue  break;  case 'd':  queue<string> FatirTempQueue  while (!FatirosQueue.empty())  if (FatirosQueue.front() == "Symbian")  FatirTempQueue.push(FatirosQueue.front())  FatirTempQueue.push("Symbian Belle")  else  FatirTempQueue.push(FatirosQueue.front())  endif  FatirosQueue.pop()  endwhile  FatirosQueue = FatirTempQueue  break;  case 'e':  FatirosQueue.push("Maemo 4")  break;  default:  print("Opsi tidak valid.")  while(!FatirosQueue.empty())  print(FatirosQueue.front())  FatirosQueue.pop()  endwhile |
| **Algoritma/Bahasa Natural (Nomor 26)**   1. queue<string> FatirosQueue 2. FatirosQueue.push("Maemo") 3. FatirosQueue.push("Symbian") 4. FatirosQueue.push("Mac") 5. FatirosQueue.push("Windows") 6. FatirosQueue.push("iPhone") 7. Memasukkan isi/nilai variabel FatirOption 8. Jika (FatirOption = ‘a’), maka kerjakan baris 9 s.d 15, kalau tidak kerjakan baris 16 9. queue<string> FatirTempQueue 10. Selama (!FatirosQueue.empty()), maka kerjakan baris 11 s.d 14, kalau tidak kerjakan baris 15 11. Selama (FatirosQueue.front() == "iPhone"), maka kerjakan baris 12, kalau tidak kerjakan baris 13 12. FatirTempQueue.push("iPhone 5") 13. FatirTempQueue.push(FatirosQueue.front()) 14. FatirosQueue.pop() 15. FatirosQueue = FatirTempQueue 16. Jika (FatirOption = ‘b’), maka kerjakan baris 17 s.d 23, kalau tidak kerjakan baris 24 17. queue<string> FatirTempQueue 18. Selama (!FatirosQueue.empty()), maka kerjakan baris 19 s.d 22, kalau tidak kerjakan baris 23 19. Selama (FatirosQueue.front() == "Windows"), maka kerjakan baris 20, kalau tidak kerjakan baris 21 20. FatirTempQueue.push("Windows Phone") 21. FatirTempQueue.push(FatirosQueue.front()) 22. FatirosQueue.pop() 23. FatirosQueue = FatirTempQueue 24. Jika (FatirOption = ‘c’), maka kerjakan baris 25 s.d 32, kalau tidak kerjakan baris 33 25. queue<string> FatirTempQueue 26. Selama (!FatirosQueue.empty()), maka kerjakan baris 27 s.d 31, kalau tidak kerjakan baris 32 27. Selama (FatirosQueue.front() == "Mac"), maka kerjakan baris 28 s.d 29, kalau tidak kerjakan baris 30 28. FatirTempQueue.push(FatirosQueue.front()) 29. FatirTempQueue.push("SnowLeopard") 30. FatirTempQueue.push(FatirosQueue.front()) 31. FatirosQueue.pop() 32. FatirosQueue = FatirTempQueue 33. Jika (FatirOption = ‘d’), maka kerjakan baris 34 s.d 41, kalau tidak kerjakan baris 42 34. queue<string> FatirTempQueue 35. Selama (!FatirosQueue.empty()), maka kerjakan baris 36 s.d 40, kalau tidak kerjakan baris 41 36. Selama (FatirosQueue.front() == "Symbian"), maka kerjakan baris 37 s.d 38, kalau tidak kerjakan baris 39 37. FatirTempQueue.push(FatirosQueue.front()) 38. FatirTempQueue.push("Symbian Belle") 39. FatirTempQueue.push(FatirosQueue.front()). 40. FatirosQueue.pop() 41. FatirosQueue = FatirTempQueue 42. Jika (FatirOption = ‘e’), maka kerjakan baris 43, kalau tidak kerjakan baris 44 43. FatirosQueue.push("Maemo 4") 44. Mencetak/menampilkan ("Opsi tidak valid.") 45. Selama (!FatirosQueue.empty()), maka kerjakan baris 46 s.d 47, kalau tidak kerjakan baris 48 46. Mencetak/menampilkan objek (FatirosQueue.front()) 47. FatirosQueue.pop() 48. Selesai |



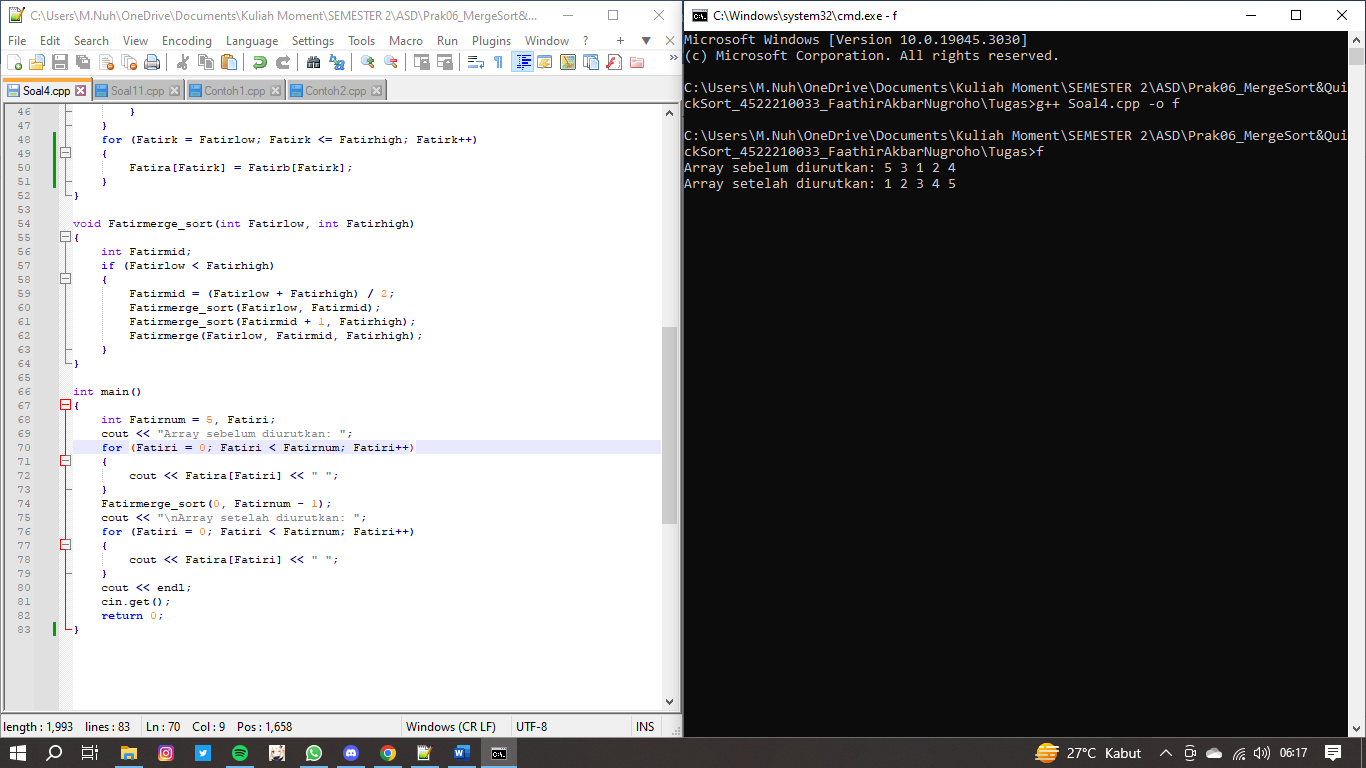
|  |
| --- |
| **Praktikum 5** |
| **Pseudocode (Nomor 9)**  **Kamus/Deklarasi**  Fatirn, FatirA[], Fatiri, Fatirk, FatirX, FatirFlag = int  **Algoritma/Deskripsi**  Fatirn = 5  FatirA[Fatirn] = {4, 2, 5, 1, 3}  Fatirk=0  FatirFlag=1  while(Fatirk<=Fatirn-2&&FatirFlag==1)  Fatiri=0  FatirFlag=0  while(Fatiri<=Fatirn-2-Fatirk)  if(FatirA[Fatiri]<FatirA[Fatiri+1])  FatirFlag=1  FatirX=FatirA[Fatiri]  FatirA[Fatiri]=FatirA[Fatiri+1]  FatirA[Fatiri+1]=FatirX  endif  FatirI++  endwhile  FatirK++  emdwhile  for(Fatiri = 0; Fatiri < Fatirn; Fatiri++)  print(FatirA[Fatiri])  endfor |
| **Algoritma/Bahasa Natural (Tugas 9)**   1. Fatirn=5 2. FatirA[Fatirn] = {4, 2, 5, 1, 3} 3. Fatirk=0 4. FatirFlag=1 5. Selama (Fatirk<=Fatirn-2&&FatirFlag==1) maka kerjakan baris 6 s.d 15, jika tidak maka kerjakan baris 16 6. Fatiri=0 7. FatirFlag=0 8. Selama (Fatiri<=Fatirn-2-Fatirk) maka kerjakan baris 9 s.d 14, jika tidak maka kerjakan baris 15 9. Jika (FatirA[Fatiri]<FatirA[Fatiri+1]) maka kerjakan baris 10 s.d 13, jika tidak maka kerjakan baris 14 10. FatirFlag=1 11. FatirX=FatirA[Fatiri] 12. FatirA[Fatiri]=FatirA[Fatiri+1] 13. FatirA[Fatiri+1]=FatirX 14. Fatiri++ 15. Fatirk++ 16. Fatiri=0 17. Selama (Fatiri < Fatirn) maka kerjakan baris 18 s.d 19, jika tidak maka kerjakan baris 20 18. Mencetak/menampilkan isi/nilai variabel FatirA[Fatiri] 19. Fatiri++ 20. Selesai |



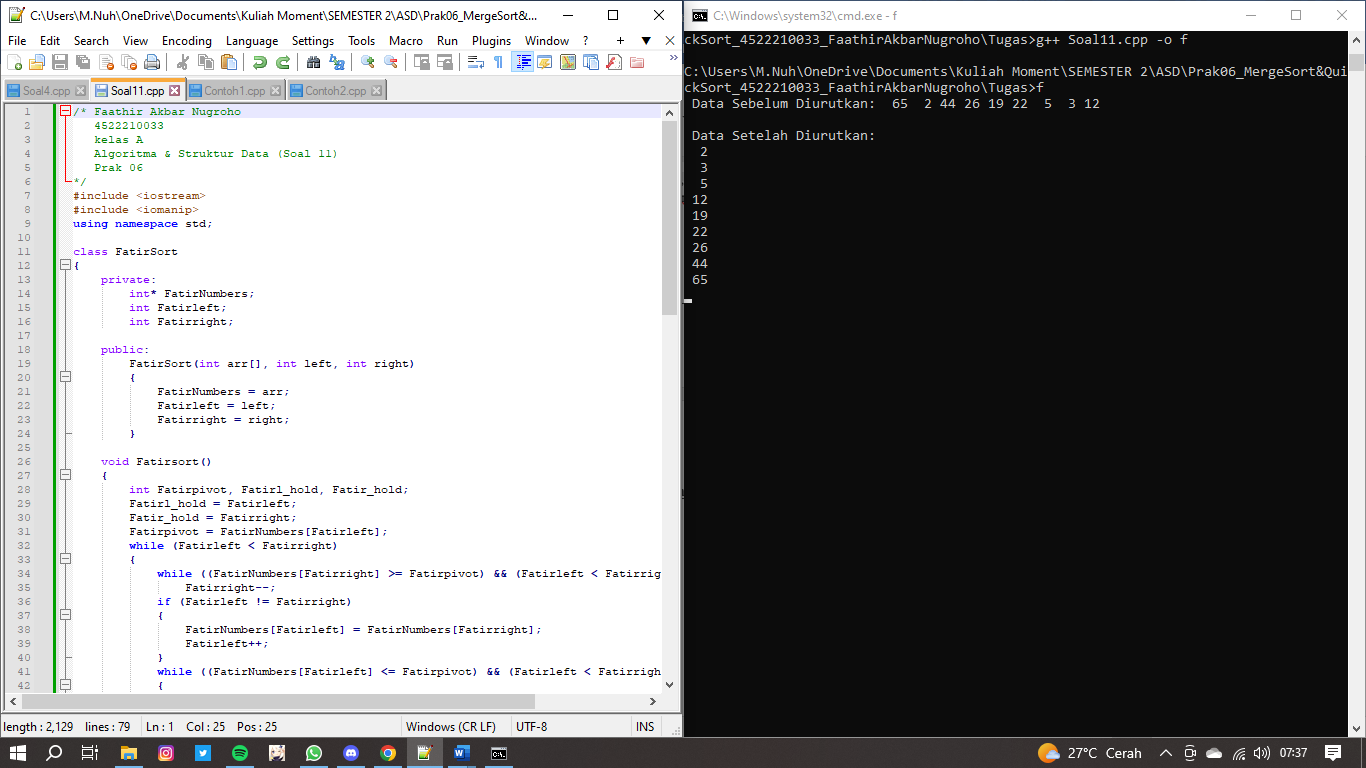
|  |
| --- |
| **Praktikum 5** |
| **Pseudocode (Nomor 14)**  **Kamus/Deklarasi Function FatirSort(int FatirA[], int Fatirn)**  FatirI, FatirJ, FatirK, FatirTemp, FatirFlag, FatirX = int  **Algoritma/Deskripsi FatirSort(int FatirA[], int Fatirn)**  FatirK=0  FatirFlag=1  while(FatirK<=Fatirn-2&&FatirFlag==1)  FatirI=0  FatirFlag=0  while(FatirI<=Fatirn-2-FatirK)  if(FatirA[FatirI]>FatirA[FatirI+1])  FatirFlag=1  FatirX=FatirA[FatirI]  FatirA[FatirI]=FatirA[FatirI+1]  FatirA[FatirI+1]=FatirX  endif  FatirI++  endwhile  FatirK++  endwhile  **Kamus/Deklarasi**  Fatirn, FatirA[] = int  **Algoritma/Deskripsi**  Class FatirShellSort  Public:  FatirSort(int FatirA[], int Fatirn)  input(Fatirn)  for(int FatirI = 0; FatirI < Fatirn; FatirI++)  input(FatirA[FatirI])  endfor  FatirShellSort obj  obj.FatirSort(FatirA, Fatirn)  for(int FatirI = 0; FatirI < Fatirn; FatirI++)  print(FatirA[FatirI])  endfor |
| **Algoritma/Bahasa Natural (Nomor 14)**   1. Membuat FatirSort(int FatirA[], int Fatirn) 2. FatirK=0 3. FatirFlag=1 4. Selama (FatirK<=Fatirn-2&&FatirFlag==1) maka kerjakan baris 5 s.d 14, jika tidak maka kerjakan baris 15 5. FatirI=0 6. FatirFlag=0 7. Selama (FatirI<=Fatirn-2-FatirK) maka kerjakan baris 8 s.d 13, jika tidak maka kerjakan baris 14 8. Jika (FatirA[FatirI]>FatirA[FatirI+1]) maka kerjakan baris 9 s.d 12, jika tidak maka kerjakan baris 13 9. FatirFlag=1 10. FatirX=FatirA[FatirI] 11. FatirA[FatirI]=FatirA[FatirI+1] 12. FatirA[FatirI+1]=FatirX 13. FatirI++ 14. FatirK++ 15. Deklarasi class FatirShellSort (FatirSort(int FatirA[], int Fatirn)) dengan penentu akses secara public 16. Memasukkan isi/nilai variabel Fatirn 17. FatirI = 0 18. Selama (FatirI < Fatirn) maka kerjakan baris 19 s.d 20, jika tidak maka kerjakan baris 21 19. Memasukkan isi/nilai variabel FatirA[FatirI] 20. FatirI++ 21. Mendefinisikan class (FatirShellSort obj) 22. Pengaksesan anggota struktur class (obj.FatirSort(FatirA, Fatirn)) 23. FatirI = 0 24. Selama (FatirI < Fatirn) maka kerjakan baris 25 s.d 26, jika tidak maka kerjakan baris 27 25. Menampilkan isi/nilai variabel FatirA[FatirI] 26. FatirI++ 27. Selesai |



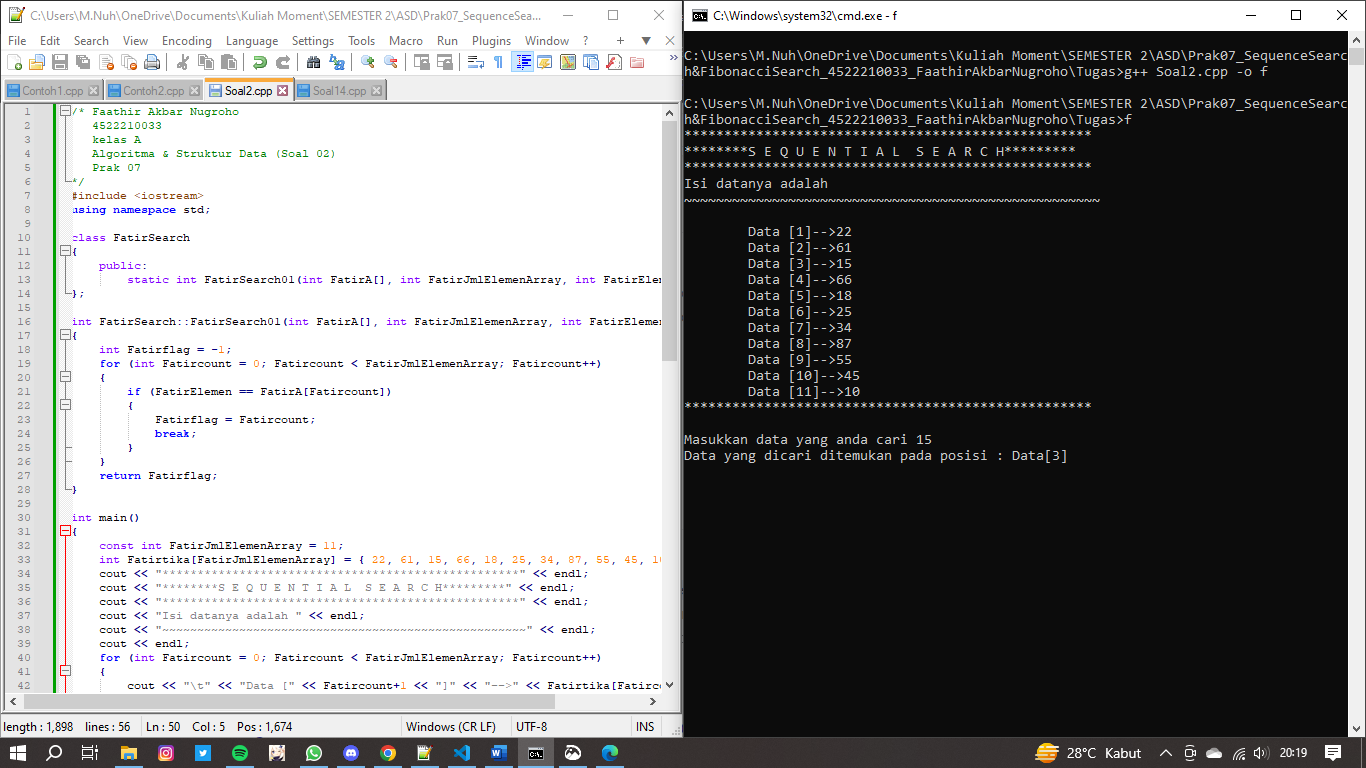
|  |
| --- |
| **Praktikum 6** |
| **Pseudocode (Nomor 04)**  **Kamus/Deklarasi Variabel Function Fatirmerge(int Fatirlow, int Fatirmid, int Fatirhigh)**  Fatirlow, Fatirmid, Fatirhigh, Fatirh, Fatiri, Fatirj, Fatirb[], Fatirk = int  **Algoritma/Deskripsi Function Fatirmerge(int Fatirlow, int Fatirmid, int Fatirhigh)**  Fatirh = Fatirlow  Fatiri = Fatirlow  Fatirj = Fatirmid + 1  while ((Fatirh <= Fatirmid) && (Fatirj <= Fatirhigh))  if (Fatira[Fatirh] <= Fatira[Fatirj])  Fatirb[Fatiri] = Fatira[Fatirh]  Fatirh++  else  Fatirb[Fatiri] = Fatira[Fatirj]  Fatirj++  endif  Fatiri++  endwhile  if (Fatirh > Fatirmid)  for (Fatirk = Fatirj; Fatirk <= Fatirhigh; Fatirk++)  Fatirb[Fatiri] = Fatira[Fatirk]  Fatiri++  endfor  else  for (Fatirk = Fatirh; Fatirk <= Fatirmid; Fatirk++)  Fatirb[Fatiri] = Fatira[Fatirk]  Fatiri++  endfor  endif  for (Fatirk = Fatirlow; Fatirk <= Fatirhigh; Fatirk++)  Fatira[Fatirk] = Fatirb[Fatirk]  endfor  **Kamus/Deklarasi Variabel Function Fatirmerge\_sort(int Fatirlow, int Fatirhigh)**  Fatirmid = int  **Algoritma/Deskripsi Function Fatirmerge\_sort(int Fatirlow, int Fatirhigh)**  if (Fatirlow < Fatirhigh)  Fatirmid = (Fatirlow + Fatirhigh) / 2  Fatirmerge\_sort(Fatirlow, Fatirmid)  Fatirmerge\_sort(Fatirmid + 1, Fatirhigh)  Fatirmerge(Fatirlow, Fatirmid, Fatirhigh)  endif  **Kamus/Deklarasi Variabel**  Fatirnum, Fatiri = int  **Algoritma/Deskripsi**  Fatirnum = 5  for (Fatiri = 0; Fatiri < Fatirnum; Fatiri++)  print(Fatira[Fatiri])  endfor  Fatirmerge\_sort(0, Fatirnum - 1)  for (Fatiri = 0; Fatiri < Fatirnum; Fatiri++)  print(Fatira[Fatiri])  endfor |
| **Algoritma/Bahasa Natural (Nomor 04)**   1. Membuat function Fatirmerge(int Fatirlow, int Fatirmid, int Fatirhigh) 2. Fatirh = Fatirlow 3. Fatiri = Fatirlow 4. Fatirj = Fatirmid + 1 5. Selama ((Fatirh <= Fatirmid) && (Fatirj <= Fatirhigh)), maka kerjakan baris 6 s.d 11, kalau tidak kerjakan baris 12 6. Jika (Fatira[Fatirh] <= Fatira[Fatirj]), maka kerjakan baris 7 s.d 8, kalau tidak kerjakan baris 9 s.d 10 7. Fatirb[Fatiri] = Fatira[Fatirh] 8. Fatirh++ 9. Fatirb[Fatiri] = Fatira[Fatirj] 10. Fatirj++ 11. Fatiri++ 12. Jika (Fatirh > Fatirmid), maka kerjakan baris 13 s.d 17, kalau tidak kerjakan baris 18 13. Fatirk = Fatirj 14. Selama (Fatirk <= Fatirhigh), maka kerjakan baris 15 s.d 17, kalau tidak kerjakan baris 18 15. Fatirb[Fatiri] = Fatira[Fatirk] 16. Fatiri++ 17. Fatirk++ 18. Fatirk = Fatirh 19. Selama (Fatirk <= Fatirmid), maka kerjakan baris 20 s.d 22, kalau tidak kerjakan baris 23 20. Fatirb[Fatiri] = Fatira[Fatirk] 21. Fatiri++ 22. Fatirk++ 23. Fatirk = Fatirlow 24. Selama (Fatirk <= Fatirhigh), maka kerjakan baris 25 s.d 26, kalau tidak kerjakan baris 27 25. Fatira[Fatirk] = Fatirb[Fatirk] 26. Fatirk++ 27. Membuat function Fatirmerge\_sort(int Fatirlow, int Fatirhigh) 28. Jika (Fatirlow < Fatirhigh) 29. Fatirmid = (Fatirlow + Fatirhigh) / 2 30. Fatirmerge\_sort(Fatirlow, Fatirmid) 31. Fatirmerge\_sort(Fatirmid + 1, Fatirhigh) 32. Fatirmerge(Fatirlow, Fatirmid, Fatirhigh) 33. Fatirnum = 5 34. Fatiri = 0 35. Selama (Fatiri < Fatirnum), maka kerjakan baris 36 s.d 37, kalau tidak kerjakan baris 38 36. Menampilkan isi/nilai variabel (Fatira[Fatiri]) 37. Fatiri++ 38. Memanggil function Fatirmerge\_sort(0, Fatirnum - 1) 39. Fatiri = 0 40. Selama (Fatiri < Fatirnum), maka kerjakan baris 41 s.d 42, kalau tidak kerjakan baris 43 41. Menampilkan isi/nilai variabel (Fatira[Fatiri]) 42. Fatiri++ 43. Selesai |



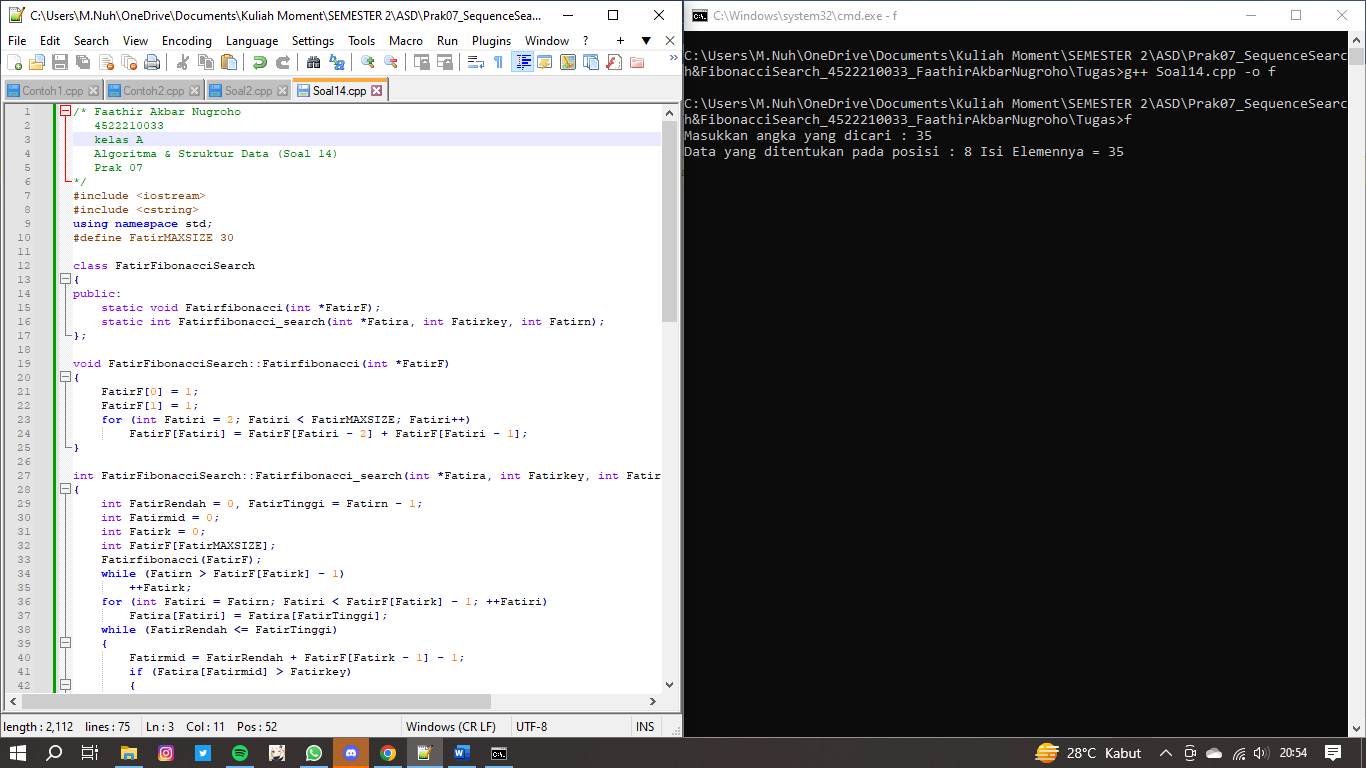
|  |
| --- |
| **Praktikum 6** |
| **Pseudocode (Nomor 11)**  **Kamus/Deklarasi Variabel Function Fatirsort()**  arr[], left, right, Fatirpivot, Fatirl\_hold, Fatir\_hold = int  **Algoritma/Deskripsi Function Fatirsort()**  Fatirl\_hold = Fatirleft  Fatir\_hold = Fatirright  Fatirpivot = FatirNumbers[Fatirleft]  while (Fatirleft < Fatirright)  while ((FatirNumbers[Fatirright] >= Fatirpivot) && (Fatirleft < Fatirright))  Fatirright--  endwhile  if (Fatirleft != Fatirright)  FatirNumbers[Fatirleft] = FatirNumbers[Fatirright];  Fatirleft++  endif  while ((FatirNumbers[Fatirleft] <= Fatirpivot) && (Fatirleft < Fatirright))  Fatirleft++  endwhile  if (Fatirleft != Fatirright)  FatirNumbers[Fatirright] = FatirNumbers[Fatirleft]  Fatirright--  endif  endwhile  FatirNumbers[Fatirleft] = Fatirpivot  Fatirpivot = Fatirleft  Fatirleft = Fatirl\_hold  Fatirright = Fatir\_hold  if (Fatirleft < Fatirpivot)  FatirSort(FatirNumbers, Fatirleft, Fatirpivot - 1).Fatirsort()  endif  if (Fatirright > Fatirpivot)  FatirSort(FatirNumbers, Fatirpivot + 1, Fatirright).Fatirsort()  endif  **Kamus/Deklarasi Variabel**  FatirNumList[], FatirNumbers, Fatirleft, Fatirright = int  **Algoritma/Deskripsi**  class FatirSort  private: int FatirNumbers, int Fatirleft, int Fatirright  public: FatirSort(int arr[], int left, int right)  FatirNumbers = arr  Fatirleft = left  Fatirright = right  FatirNumList[9] = { 65,2,44,26,19,22,5,3,12}  for (int Fatird = 0; Fatird < 9; Fatird++)  print(FatirNumList[Fatird])  endfor  FatirSort(FatirNumList, 0, 9).Fatirsort()  for (int Fatiri = 0; Fatiri < 9; Fatiri++)  print(FatirNumList[Fatiri])  endfor |
| **Algoritma/Bahasa Natural (Nomor 11)**   1. Membuat function Fatirsort() 2. Fatirl\_hold = Fatirleft 3. Fatir\_hold = Fatirright 4. Fatirpivot = FatirNumbers[Fatirleft] 5. Selama (Fatirleft < Fatirright), maka kerjakan baris 6 s.d 15, kalau tidak kerjakan baris 16 6. Selama ((FatirNumbers[Fatirright] >= Fatirpivot) && (Fatirleft < Fatirright)), maka kerjakan baris 7, kalau tidak kerjakan baris 8 7. Fatirright-- 8. Jika (Fatirleft != Fatirright), maka kerjakan baris 9 s.d 10, kalau tidak kerjakan baris 11 9. FatirNumbers[Fatirleft] = FatirNumbers[Fatirright] 10. Fatirleft++ 11. Selama ((FatirNumbers[Fatirleft] <= Fatirpivot) && (Fatirleft < Fatirright)), maka kerjakan baris 12, kalau tidak kerjakan baris 13 12. Fatirleft++ 13. Jika (Fatirleft != Fatirright), maka kerjakan baris 14 s.d 15, kalau tidak kerjakan baris 16 14. FatirNumbers[Fatirright] = FatirNumbers[Fatirleft] 15. Fatirright-- 16. FatirNumbers[Fatirleft] = Fatirpivot 17. Fatirpivot = Fatirleft 18. Fatirleft = Fatirl\_hold 19. Fatirright = Fatir\_hold 20. Jika (Fatirleft < Fatirpivot), maka kerjakan baris 21, kalau tidak kerjakan baris 22 21. FatirSort(FatirNumbers, Fatirleft, Fatirpivot - 1).Fatirsort() 22. Jika (Fatirright > Fatirpivot), maka kerjakan baris 23, kalau tidak kerjakan baris 24 23. FatirSort(FatirNumbers, Fatirpivot + 1, Fatirright).Fatirsort() 24. Mendeklarasikan class (class FatirSort (FatirNumbers, Fatirleft, Fatirright)) dengan penentu akses private 25. Mendeklarasikan class (class FatirSort (FatirSort(int arr[], int left, int right))) dengan penentu akses public 26. FatirNumList[9] = {65,2,44,26,19,22,5,3,12} 27. Fatird = 0 28. Selama (Fatird < 9), maka kerjakan baris 29 s.d 30, kalau tidak kerjakan baris 31 29. Menampilkan isi/nilai variabel (FatirNumList[Fatird]) 30. Fatird++ 31. Memanggil function FatirSort(FatirNumList, 0, 9).Fatirsort() 32. Fatiri = 0 33. Selama (Fatiri < 9), maka kerjakan baris 34 s.d 35, kalau tidak kerjakan baris 36 34. Menampilkan isi/nilai variabel (FatirNumList[Fatiri]) 35. Fatiri++ 36. Selesai |



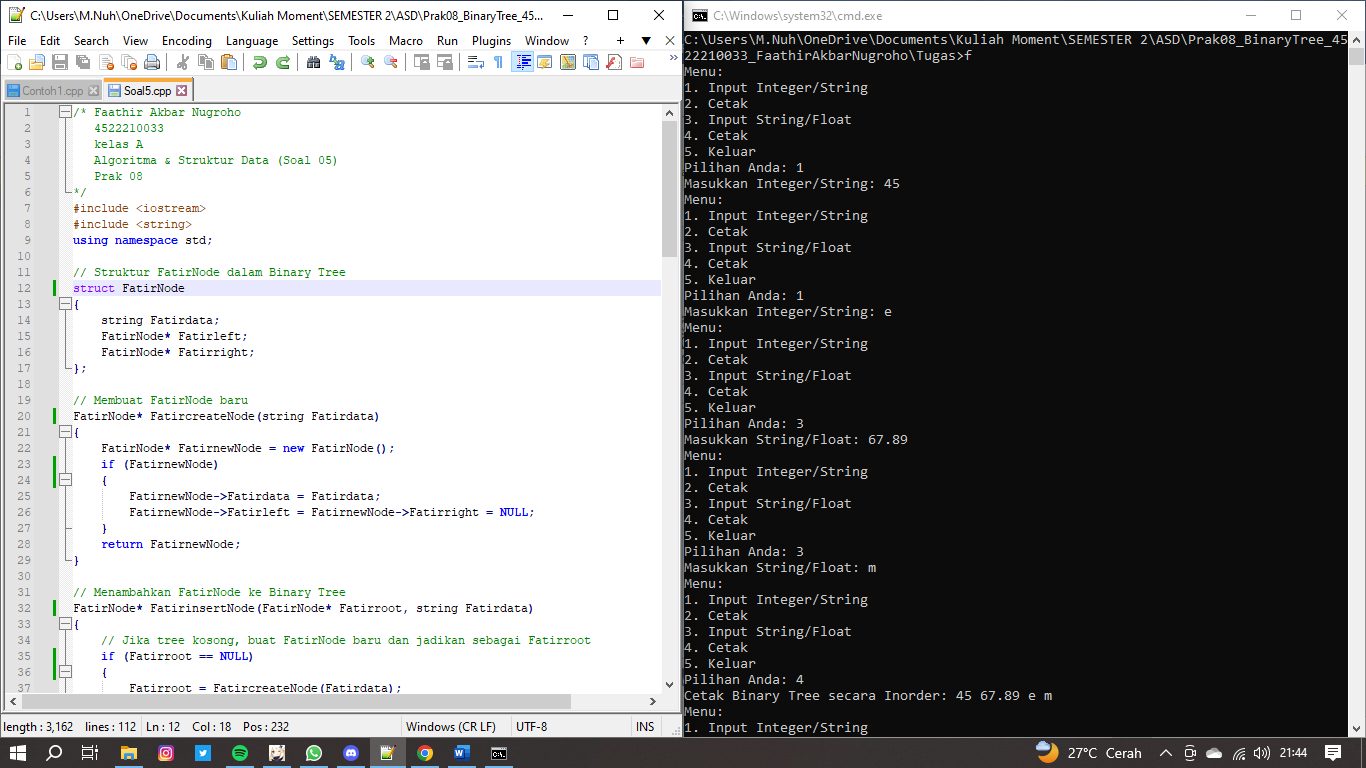
|  |
| --- |
| **Praktikum 7** |
| **Pseudocode (Nomor 02)**  **Kamus/Deklarasi Variabel FatirSearch::FatirSearch01(int FatirA[], int FatirJmlElemenArray, int FatirElemen)**  -  **Algoritma/Deskripsi FatirSearch::FatirSearch01(int FatirA[], int FatirJmlElemenArray, int FatirElemen)**  int Fatirflag = -1  for (int Fatircount = 0; Fatircount < FatirJmlElemenArray; Fatircount++)  if (FatirElemen== FatirA[Fatircount])  Fatirflag = Fatircount  break  endif  endfor  return Fatirflag  **Kamus/Deklarasi Variabel**  -  **Algoritma/Deskripsi**  class FatirSearch  public:  static int FatirSearch01(int FatirA[], int FatirJmlElemenArray, int FatirElemen)  const int FatirJmlElemenArray = 11  int Fatirtika[FatirJmlElemenArray] = { 22, 61, 15, 66, 18, 25, 34, 87, 55, 45, 10 }  for (int Fatircount = 0; Fatircount < FatirJmlElemenArray; Fatircount++)  print((Fatircount+1),( Fatirtika[Fatircount]))  endfor  int FatirSearchElemen = 15  int Fatirflag = 0  print(FatirSearchElemen)  Fatirflag = FatirSearch::FatirSearch01(Fatirtika, FatirJmlElemenArray, FatirSearchElemen)  if (Fatirflag != -1)  print(Fatirflag+1)  else  print(“Data yang anda cari tidak ditemukan”)  endif |
| **Algoritma/Bahasa Natural (Nomor 02)**   1. Membuat function FatirSearch::FatirSearch01(int FatirA[], int FatirJmlElemenArray, int FatirElemen) 2. Fatirflag = -1 3. Fatircount = 0 4. Selama (Fatircount < FatirJmlElemenArray), maka kerjakan baris 5 s.d 7, kalau tidak kerjakan baris 8 5. Jika ((FatirElemen == FatirA[Fatircount])), maka kerjakan baris 6, kalau tidak kerjakan baris 7 6. Fatirflag = Fatircount 7. Fatircount++ 8. Kembali ke Fatirflag 9. Mendeklarasikan class (class FatirSearch(static int FatirSearch01(int FatirA[], int FatirJmlElemenArray, int FatirElemen))) dengan tipe akses secara public 10. FatirJmlElemenArray = 11 11. Fatirtika[FatirJmlElemenArray] = { 22, 61, 15, 66, 18, 25, 34, 87, 55, 45, 10 } 12. Fatircount = 0 13. Selama (Fatircount < FatirJmlElemenArray), maka kerjakan baris 14 s.d 15, kalau tidak kerjakan baris 16 14. Menampilkan isi/nilai variabel ((Fatircount+1),( Fatirtika[Fatircount])) 15. Fatircount++ 16. FatirSearchElemen = 15 17. Fatirflag = 0 18. Menampilkan isi/nilai variabel FatirSearchElemen 19. Fatirflag = FatirSearch::FatirSearch01(Fatirtika, FatirJmlElemenArray, FatirSearchElemen) 20. Jika (Fatirflag != -1), maka kerjakan baris 21, kalau tidak kerjakan baris 22 21. Menampilkan isi/nilai variabel (Fatirflag+1) 22. Menampilkan (“Data yang anda cari tidak ditemukan”) 23. Selesai |



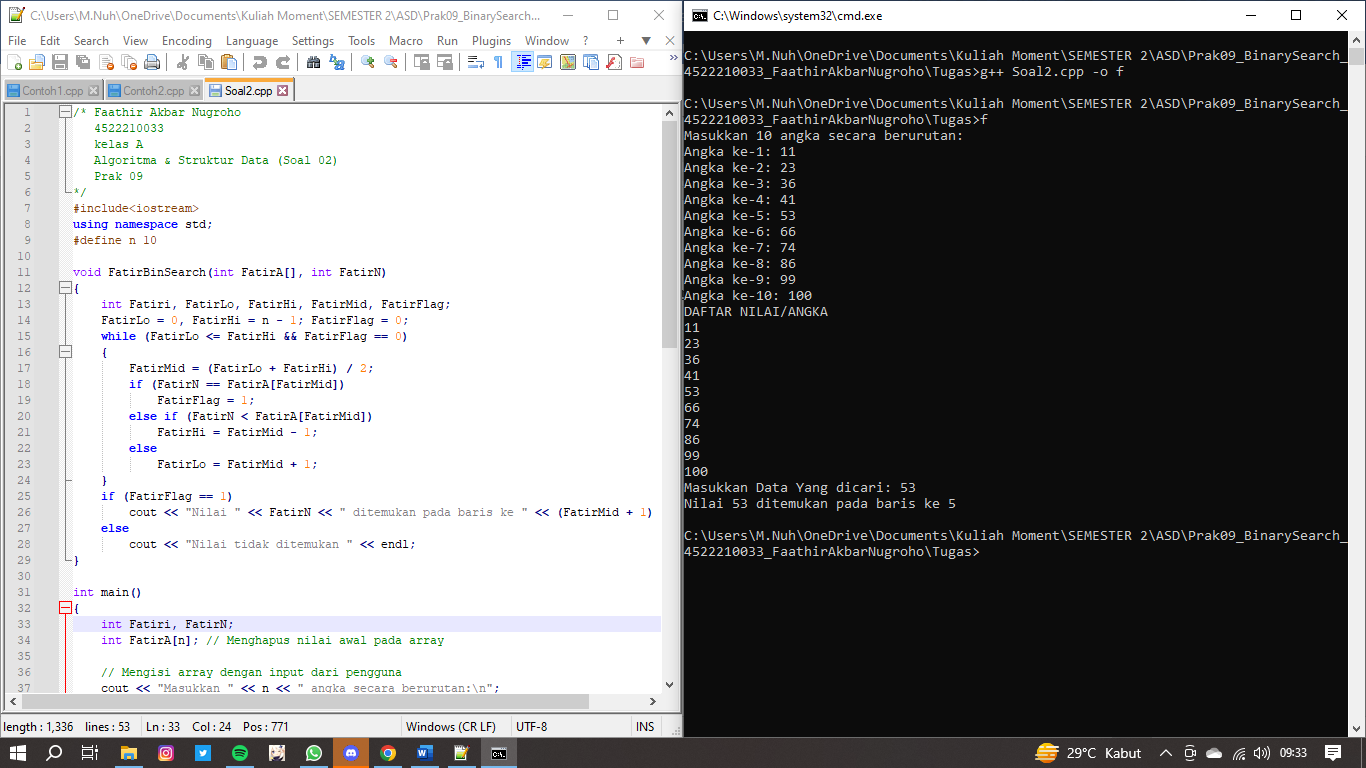
|  |
| --- |
| **Praktikum 7** |
| **Pseudocode (Nomor 14)**  **Kamus/Deklarasi Variabel Function FatirFibonacciSearch::Fatirfibonacci(int \*FatirF)**  -  **Algoritma/Deskripsi** **Function FatirFibonacciSearch::Fatirfibonacci(int \*FatirF)**  FatirF[0] = 1  FatirF[1] = 1  for (int Fatiri = 2; Fatiri < FatirMAXSIZE; Fatiri++)  FatirF[Fatiri] = FatirF[Fatiri - 2] + FatirF[Fatiri - 1]  endfor  **Kamus/Deklarasi Variabel FatirFibonacciSearch::Fatirfibonacci\_search(int \*Fatira, int Fatirkey, int Fatirn)**  FatirF[] = int  **Algoritma/Deskripsi FatirFibonacciSearch::Fatirfibonacci\_search(int \*Fatira, int Fatirkey, int Fatirn)**  int FatirRendah = 0, FatirTinggi = Fatirn - 1  int Fatirmid = 0  int Fatirk = 0  Fatirfibonacci(FatirF)  while (Fatirn > FatirF[Fatirk] - 1)  ++Fatirk  endwhile  for (int Fatiri = Fatirn; Fatiri < FatirF[Fatirk] - 1; ++Fatiri)  Fatira[Fatiri] = Fatira[FatirTinggi]  endfor  while (FatirRendah <= FatirTinggi)  Fatirmid = FatirRendah + FatirF[Fatirk - 1] - 1  if (Fatira[Fatirmid] > Fatirkey)  FatirTinggi = Fatirmid - 1  Fatirk = Fatirk - 1  else if (Fatira[Fatirmid] < Fatirkey)  FatirRendah = Fatirmid + 1  Fatirk = Fatirk - 2  else  if (Fatirmid <= FatirTinggi)  return Fatirmid  else  return -1  endif  endif  endwhile  return -1  **Kamus/Deklarasi Variabel**  **-**  **Algoritma/Deskripsi**  class FatirFibonacciSearch  public:  static void Fatirfibonacci(int \*FatirF)  static int Fatirfibonacci\_search(int \*Fatira, int Fatirkey, int Fatirn)  int Fatira[FatirMAXSIZE] = {1, 4, 6, 8, 9, 11, 23, 35, 47, 51, 55, 63, 64, 78, 88, 95, 99}  int Fatirk = 35  print(Fatirk)  int Fatirpos = FatirFibonacciSearch::Fatirfibonacci\_search(Fatira, Fatirk, 17)  if (Fatirpos != -1)  print((Fatirpos + 1),( Fatirk))  else  print("Data tidak ditemukan")  endif |
| **Algoritma/Bahasa Natural (Nomor 14)**   1. Membuat function FatirFibonacciSearch::Fatirfibonacci(int \*FatirF) 2. FatirF[0] = 1 3. FatirF[1] = 1 4. Fatiri = 2 5. Selama (Fatiri < FatirMAXSIZE), maka kerjakan baris 6 s.d 7, kalau tidak kerjakan baris 8 6. FatirF[Fatiri] = FatirF[Fatiri - 2] + FatirF[Fatiri - 1] 7. Fatiri++ 8. Membuat function FatirFibonacciSearch::Fatirfibonacci\_search(int \*Fatira, int Fatirkey, int Fatirn) 9. FatirRendah = 0 10. FatirTinggi = Fatirn - 1 11. Fatirmid = 0 12. Fatirk = 0 13. FatirF[FatirMAXSIZE] 14. Memanggil function Fatirfibonacci(FatirF) 15. Selama (Fatirn > FatirF[Fatirk] - 1), maka kerjakan baris 16, kalau tidak kerjakan baris 17 16. ++Fatirk 17. Fatiri = Fatirn 18. Selama (Fatiri < FatirF[Fatirk] - 1), maka kerjakan baris 19 s.d 20, kalau tidak kerjakan baris 21 19. Fatira[Fatiri] = Fatira[FatirTinggi] 20. ++Fatiri 21. Selama (FatirRendah <= FatirTinggi), maka kerjakan baris 22 s.d 31, kalau tidak kerjakan baris 32 22. Fatirmid = FatirRendah + FatirF[Fatirk - 1] - 1 23. Jika (Fatira[Fatirmid] > Fatirkey), maka kerjakan baris 24 s.d 25, kalau tidak kerjakan baris 26 24. FatirTinggi = Fatirmid - 1 25. Fatirk = Fatirk - 1 26. Jika (Fatira[Fatirmid] < Fatirkey), maka kerjakan baris 27 s.d 28, kalau tidak kerjakan baris 29 27. FatirRendah = Fatirmid + 1 28. Fatirk = Fatirk - 2 29. Jika (Fatirmid <= FatirTinggi), maka kerjakan baris 30, kalau tidak kerjakan baris 31 30. Kembali ke Fatirmid 31. Kembali -1 32. Kembali -1 33. Mendeklarasikan class (class FatirFibonacciSearch(static void Fatirfibonacci(int \*FatirF)), static int Fatirfibonacci\_search(int \*Fatira, int Fatirkey, int Fatirn)) 34. )) 35. Fatira[FatirMAXSIZE] = {1, 4, 6, 8, 9, 11, 23, 35, 47, 51, 55, 63, 64, 78, 88, 95, 99} 36. Fatirk = 35 37. Menampilkan isi/nilai variabel Fatirk 38. Memanggil function Fatirpos = FatirFibonacciSearch::Fatirfibonacci\_search(Fatira, Fatirk, 17) 39. Jika (Fatirpos != -1), maka kerjakan baris 40, kalau tidak kerjakan baris 41 40. Menampilkan isi/nilai variabel ((Fatirpos + 1),( Fatirk)) 41. Menampilkan ("Data tidak ditemukan") 42. Selesai |



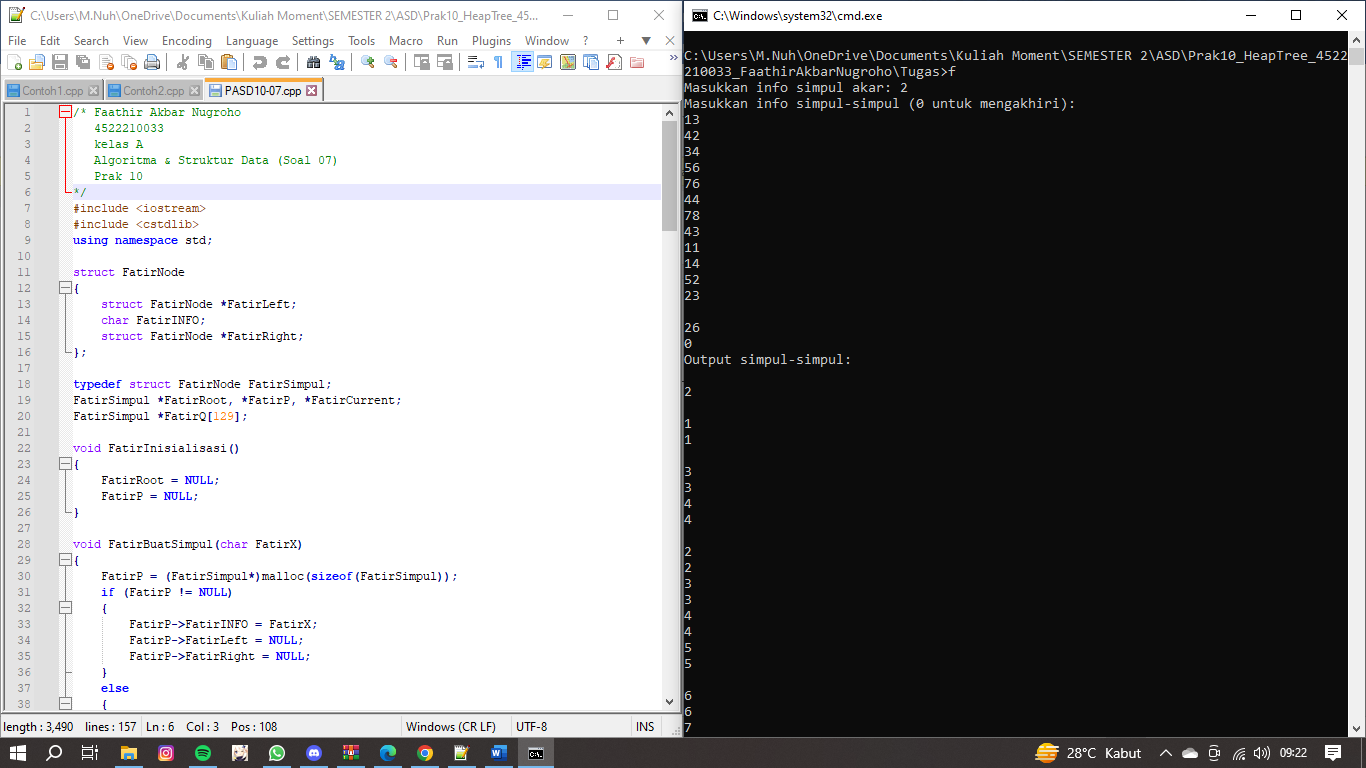
|  |
| --- |
| **Praktikum 8** |
| **Pseudocode (Nomor 05)**  **Kamus/Deklarasi Variabel Function FatirNode\* FatircreateNode(string Fatirdata)**  **-**  **Algoritma/Deskripsi Function FatirNode\* FatircreateNode(string Fatirdata)**  FatirNode\* FatirnewNode = new FatirNode()  if (FatirnewNode)  FatirnewNode->Fatirdata = Fatirdata  FatirnewNode->Fatirleft = FatirnewNode->Fatirright = NULL  endif  return FatirnewNode  **Kamus/Deklarasi Variabel Function FatirNode\* FatirinsertNode(FatirNode\* Fatirroot, string Fatirdata)**  **-**  **Algoritma/Deskripsi Function FatirNode\* FatirinsertNode(FatirNode\* Fatirroot, string Fatirdata)**  if (Fatirroot == NULL)  Fatirroot = FatircreateNode(Fatirdata)  return Fatirroot  endif  if (Fatirdata < Fatirroot->Fatirdata)  Fatirroot->Fatirleft = FatirinsertNode(Fatirroot->Fatirleft, Fatirdata)  else if (Fatirdata >= Fatirroot->Fatirdata)  Fatirroot->Fatirright = FatirinsertNode(Fatirroot->Fatirright, Fatirdata)  endif  return Fatirroot  **Kamus/Deklarasi Variabel Function FatirinorderTraversal(FatirNode\* Fatirroot)**  **-**  **Algoritma/Deskripsi Function** **FatirinorderTraversal(FatirNode\* Fatirroot)**  if (Fatirroot == NULL)  return  endif  FatirinorderTraversal(Fatirroot->Fatirleft)  print(Fatirroot->Fatirdata)  FatirinorderTraversal(Fatirroot->Fatirright)  **Kamus/Deklarasi Variabel**  Fatirchoice = int  Fatirdata = string  **Algoritma/Deskripsi**  struct FatirNode  string Fatirdata  FatirNode\* Fatirleft  FatirNode\* Fatirright  FatirNode\* Fatirroot = NULL  do  input(Fatirchoice)  switch (Fatirchoice)  case 1:  input(Fatirdata)  Fatirroot = FatirinsertNode(Fatirroot, Fatirdata)  case 2:  FatirinorderTraversal(Fatirroot)  case 3:  input(Fatirdata)  Fatirroot = FatirinsertNode(Fatirroot, Fatirdata)  case 4:  FatirinorderTraversal(Fatirroot)  case 5:  print(“Terima kasih! Program selesai.\n”)  default:  print(“Pilihan tidak valid. Silakan pilih kembali.\n”)  while (Fatirchoice != 5)  endwhile |
| **Algoritma/Bahasa Natural (Nomor 05)**   1. Membuat function FatirNode\* FatircreateNode(string Fatirdata) 2. FatirNode\* FatirnewNode = new FatirNode() 3. Jika (FatirnewNode), maka kerjakan baris 4 s.d 5, kalau tidak kerjakan baris 6 4. FatirnewNode->Fatirdata = Fatirdata 5. FatirnewNode->Fatirleft = FatirnewNode->Fatirright = NULL 6. return FatirnewNode 7. Membuat function FatirNode\* FatirinsertNode(FatirNode\* Fatirroot, string Fatirdata) 8. Jika (Fatirroot == NULL), maka kerjakan baris 9 s.d 10, kalau tidak kerjakan baris 11 9. Fatirroot = FatircreateNode(Fatirdata) 10. return Fatirroot 11. Jika (Fatirdata < Fatirroot->Fatirdata), maka kerjakan baris 12, kalau tidak kerjakan baris 13 12. Fatirroot->Fatirleft = FatirinsertNode(Fatirroot->Fatirleft, Fatirdata) 13. Jika (Fatirdata >= Fatirroot->Fatirdata), maka kerjakan baris 14, kalau tidak kerjakan baris 15 14. Fatirroot->Fatirright = FatirinsertNode(Fatirroot->Fatirright, Fatirdata) 15. return Fatirroot 16. Membuat function FatirinorderTraversal(FatirNode\* Fatirroot) 17. Jika (Fatirroot == NULL), maka kerjakan baris 18, kalau tidak kerjakan baris 19 18. return 19. FatirinorderTraversal(Fatirroot->Fatirleft) 20. Menampilkan isi/nilai variabel (Fatirroot->Fatirdata) 21. FatirinorderTraversal(Fatirroot->Fatirright) 22. Mendeklarasikan struct (struct FatirNode (string Fatirdata, FatirNode\* Fatirleft, FatirNode\* Fatirright) 23. FatirNode\* Fatirroot = NULL 24. Memasukkan isi/nilai variabel Fatirchoice 25. Jika (Fatirchoice = 1), maka kerjakan baris 25 s.d 26, kalau tidak kerjakan baris 27 26. Memasukkan isi/nilai variabel Fatirdata 27. Fatirroot = FatirinsertNode(Fatirroot, Fatirdata) 28. Jika (Fatirchoice = 2), maka kerjakan baris 28, kalau tidak kerjakan baris 29 29. FatirinorderTraversal(Fatirroot) 30. Jika (Fatirchoice = 3), maka kerjakan baris 30 s.d 31, kalau tidak kerjakan baris 32 31. Memasukkan isi/nilai variabel Fatirdata 32. Fatirroot = FatirinsertNode(Fatirroot, Fatirdata) 33. Jika (Fatirchoice = 4), maka kerjakan baris 33, kalau tidak kerjakan baris 34 34. FatirinorderTraversal(Fatirroot) 35. Jika (Fatirchoice = 5), maka kerjakan baris 35, kalau tidak kerjakan baris 36 36. Menampilkan (“Terima kasih! Program selesai.\n”) 37. Jika (Fatirchoice = default), maka kerjakan baris 37, kalau tidak kerjakan baris 38 38. Menampilkan ("Pilihan tidak valid. Silakan pilih kembali.\n") 39. Selama (Fatirchoice != 5), maka kerjakan baris 23 s,d 37, kalau tidak kerjakan baris 39 40. Selesai |



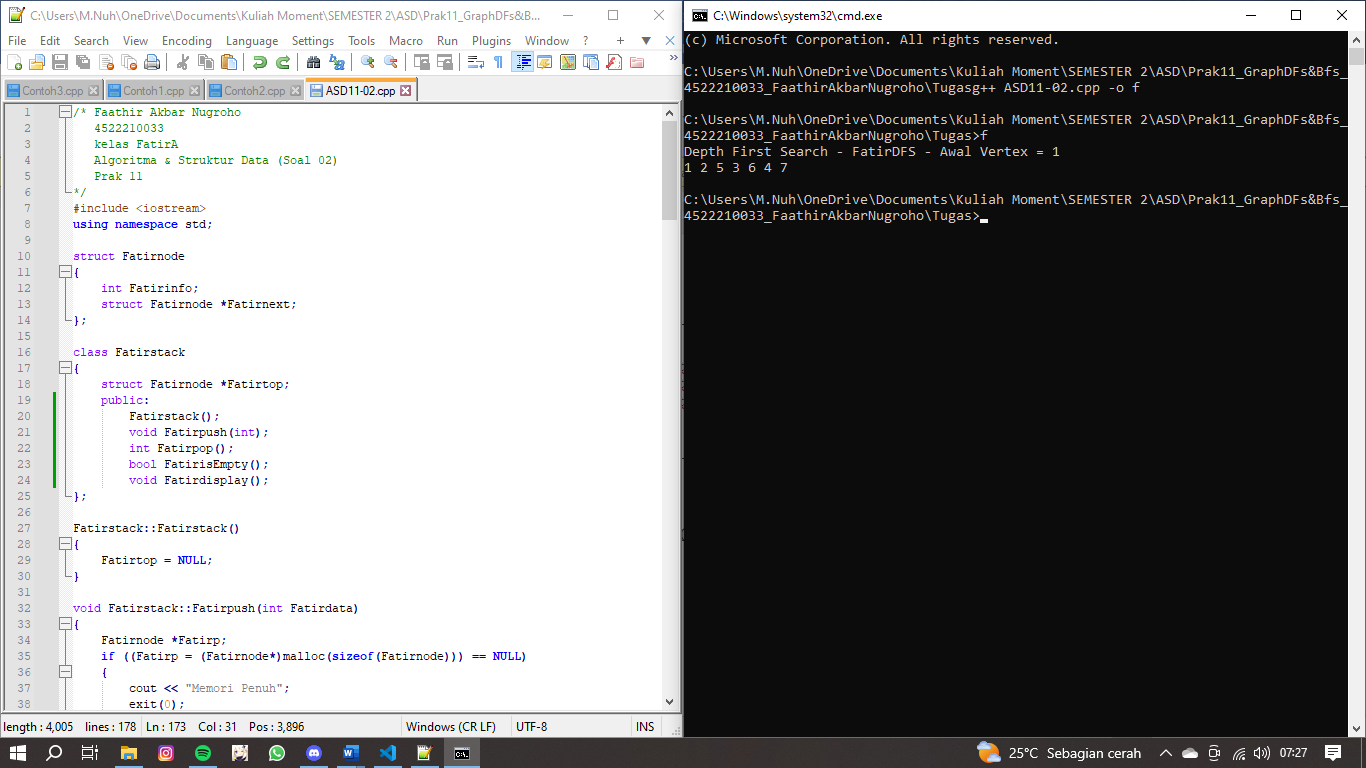
|  |
| --- |
| **Praktikum 9** |
| **Pseudocode (Nomor 02)**  **Kamus/Deklarasi Variabel Function FatirBinSearch(int FatirA[], int FatirN)**  Fatiri, FatirLo, FatirHi, FatirMid, FatirFlag=int  **Algoritma/Deskripsi Function FatirBinSearch(int FatirA[], int FatirN)**  while (FatirLo <= FatirHi && FatirFlag == 0)  FatirMid = (FatirLo + FatirHi) / 2  if (FatirN == FatirA[FatirMid])  FatirFlag = 1  else  if (FatirN < FatirA[FatirMid])  FatirHi = FatirMid - 1  else  FatirLo = FatirMid + 1  endif  endif  endwhile  if (FatirFlag == 1)  print((FatirN),( (FatirMid + 1)))  else  ("Nilai tidak ditemukan")  endif  **Kamus/Deklarasi Variabel**  Fatiri, FatirN, FatirA[] =int  **Algoritma/Deskripsi**  input(n)  for (Fatiri = 0; Fatiri < n; Fatiri++)  print(Fatiri + 1)  input(FatirA[Fatiri])  endfor  for (Fatiri = 0; Fatiri < n; Fatiri++)  print(FatirA[Fatiri])  endfor  input(FatirN)  FatirBinSearch(FatirA, FatirN) |
| **Algoritma/Bahasa Natural (Nomor 02)**   1. Membuat function FatirBinSearch(int FatirA[], int FatirN) 2. FatirLo=0, FatirHi=n-1; FatirFlag=0 3. Selama (FatirLo <= FatirHi && FatirFlag==0), maka kerjakan baris 4 s.d 9, kalau tidak kerjakan baris 10 4. FatirMid=(FatirLo+FatirHi)/2 5. Jika (FatirN==FatirA[FatirMid]), maka kerjakan baris 6, kalau tidak kerjakan baris 7 6. FatirFlag=1 7. Jika (FatirN<FatirA[FatirMid]), maka kerjakan baris 8, kalau tidak kerjakan baris 9 8. FatirHi=FatirMid-1 9. FatirLo=FatirMid+1 10. Jika (FatirFlag==1), maka kerjakan baris 11, kalau tidak kerjakan baris 12 11. Menampilkan isi/nilai variabel ((Fatirn),( Fatirmid + 1)) 12. Menampilkan ("Nilai tidak ditemukan") 13. Memasukkan isi/nilai variabel n 14. Fatiri = 0 15. Selama (Fatiri < n), maka kerjakan baris 16 s.d 18, kalau tidak kerjakan baris 19 16. Menampilkan isi/nilai variabel (Fatiri + 1) 17. Memasukkan isi/nilai variabel (FatirA[Fatiri]) 18. Fatiri++ 19. Fatiri = 0 20. Selama (Fatiri < n), maka kerjakan baris 21 s.d 22, kalau tidak kerjakan baris 23 21. Menampilkan isi/nilai variabel (FatirA[Fatiri]) 22. Fatiri++ 23. Memasukkan isi/nilai variabel FatirN 24. Memanggil function FatirBinSearch(FatirA, FatirN) 25. Selesai |



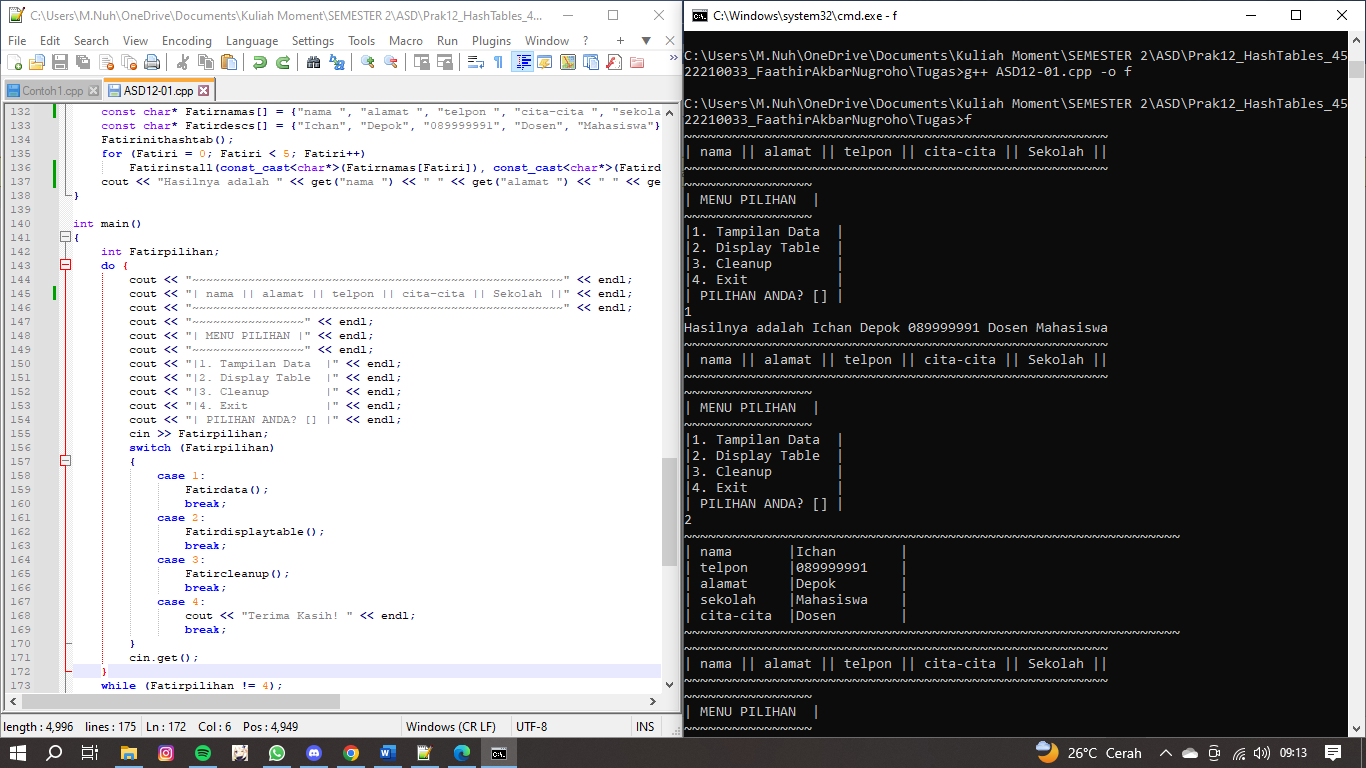
|  |
| --- |
| **Praktikum 10** |
| **Pseudocode (Nomor 07)**  **Kamus/Deklarasi Variabel Function FatirInisialisasi()**  -  **Algoritma/Deskripsi Function FatirInisialisasi()**  FatirRoot = NULL  FatirP = NULL  **Kamus/Deklarasi Variabel Function FatirBuatSimpul(char FatirX)**  -  **Algoritma/Deskripsi Function FatirBuatSimpul(char FatirX)**  FatirP = (FatirSimpul\*)malloc(sizeof(FatirSimpul))  if (FatirP != NULL)  FatirP->FatirINFO = FatirX  FatirP->FatirLeft = NULL  FatirP->FatirRight = NULL  else  print(“Memory Heap Full”)  exit(1)  endif **Kamus/Deklarasi Variabel Function** **FatirBuatSimpulAkar()**  **-**  **Algoritma/Deskripsi Function FatirBuatSimpulAkar()**  if (FatirRoot == NULL)  if (FatirP != NULL)  FatirRoot = FatirP  FatirRoot->FatirLeft = NULL  FatirRoot->FatirRight = NULL  else  print(“FatirSimpul Belum Dibuat”)  endif  else  print(“Pohon Sudah Ada!!!!”)  endif  **Kamus/Deklarasi Variabel Function FatirInsertUrutNomor()**  Fatiri, Fatirj, FatirFlag=int  FatirX=char  **Algoritma/Deskripsi Function** **FatirInsertUrutNomor()**  FatirFlag = 0  Fatiri = 1  Fatirj = 1  FatirQ[Fatiri] = FatirRoot  while (FatirFlag == 0 && Fatirj < 127)  input(FatirX)  if (FatirX != '0')  FatirBuatSimpul(FatirX)  FatirCurrent = FatirQ[Fatiri]  FatirCurrent->FatirLeft = FatirP  Fatirj++  FatirQ[Fatirj] = FatirP  else  FatirFlag = 1  Fatirj++  FatirQ[Fatirj] = NULL  endif  if (FatirFlag == 0)  if (FatirX != '0')  FatirBuatSimpul(FatirX)  FatirCurrent->FatirRight = FatirP  Fatirj++  FatirQ[Fatirj] = FatirP  else  FatirFlag = 1  Fatirj++  FatirQ[Fatirj] = NULL  endif  endif  Fatiri++  endwhile  **Kamus/Deklarasi Variabel Function FatirBacaUrutNomor()**  Fatiri, Fatirj, Fatirn, FatirCounter=int  FatirX=char  **Algoritma/Deskripsi Function FatirBacaUrutNomor()**  Fatiri = 1  Fatirj = 1  Fatirn = 1  FatirCounter = 0  while (FatirQ[Fatiri] != NULL)  FatirCurrent = FatirQ[Fatiri]  print(FatirCurrent->FatirINFO)  FatirCounter++  if (FatirCounter == Fatirn)  FatirCounter = 0  Fatirn = Fatirn \* 2  endif  if (FatirCurrent->FatirLeft != NULL)  Fatirj++  FatirQ[Fatirj] = FatirCurrent->FatirLeft  endif  if (FatirCurrent->FatirRight != NULL)  Fatirj++  FatirQ[Fatirj] = FatirCurrent->FatirRight  endif  Fatiri++  **Kamus/Deklarasi Variabel**  FatirX=char  **Algoritma/Deskripsi**  struct FatirNode  struct FatirNode \*FatirLeft  char FatirINFO  struct FatirNode \*FatirRight  FatirInisialisasi()  input(FatirX)  FatirBuatSimpul(FatirX)  FatirBuatSimpulAkar()  FatirInsertUrutNomor()  FatirBacaUrutNomor() |
| **Algoritma/Bahasa Natural (Nomor 07)**   1. Membuat function FatirInisialisasi() 2. FatirRoot = NULL 3. FatirP = NULL 4. Membuat function FatirBuatSimpul(char FatirX) 5. FatirP = (FatirSimpul\*)malloc(sizeof(FatirSimpul)) 6. Jika (FatirP != NULL), maka kerjakan baris 7 s.d 9, kalau tidak kerjakan baris 10 s.d 11 7. FatirP->FatirINFO = FatirX 8. FatirP->FatirLeft = NULL 9. FatirP->FatirRight = NULL 10. Menampilkan (“Memory Heap Full”) 11. exit(1) 12. Membuat function FatirBuatSimpulAkar() 13. Jika (FatirRoot == NULL), maka kerjakan baris 14 s.d 18, kalau tidak kerjakan baris 19 14. Jika (FatirP != NULL), maka kerjakan baris 15 s.d 17, kalau tidak kerjakan baris 18 15. FatirRoot = FatirP 16. FatirRoot->FatirLeft = NULL 17. FatirRoot->FatirRight = NULL 18. Menampilkan (“FatirSimpul Belum Dibuat”) 19. Menampilkan (“Pohon Sudah Ada!!!!”) 20. Membuat function FatirInsertUrutNomor() 21. FatirFlag = 0 22. Fatiri = 1 23. Fatirj = 1 24. FatirQ[Fatiri] = FatirRoot 25. Selama (FatirFlag == 0 && Fatirj < 127), maka kerjakan baris 26 s.d 45, kalau tidak kerjakan baris 46 26. Memasukkan isi/nilai variabel FatirX 27. Jika (FatirX != '0'), maka kerjakan baris 28 s.d 32, kalau tidak kerjakan baris 33 s.d 35 28. FatirBuatSimpul(FatirX) 29. FatirCurrent = FatirQ[Fatiri] 30. FatirCurrent->FatirLeft = FatirP 31. Fatirj++ 32. FatirQ[Fatirj] = FatirP 33. FatirFlag = 1 34. Fatirj++ 35. FatirQ[Fatirj] = NULL 36. Jika (FatirFlag == 0), maka kerjakan baris 37 s.d 44, kalau tidak kerjakan baris 45 37. Jika (FatirX != '0'), maka kerjakan baris 38 s.d 41, kalau tidak kerjakan baris 42 s.d 44 38. FatirBuatSimpul(FatirX) 39. FatirCurrent->FatirRight = FatirP 40. Fatirj++ 41. FatirQ[Fatirj] = FatirP 42. FatirFlag = 1 43. Fatirj++ 44. FatirQ[Fatirj] = NULL 45. Fatiri++ 46. Membuat function FatirBacaUrutNomor() 47. Fatiri = 1 48. Fatirj = 1 49. Fatirn = 1 50. FatirCounter = 0 51. Selama (FatirQ[Fatiri] != NULL), maka kerjakan baris 52 s.d 64, kalau tidak kerjakan baris 65 52. FatirCurrent = FatirQ[Fatiri] 53. Menampilkan isi/nilai variabel (FatirCurrent->FatirINFO) 54. FatirCounter++ 55. Jika (FatirCounter == Fatirn), maka kerjakan baris 56 s.d 57, kalau tidak kerjakan baris 58 56. FatirCounter = 0 57. Fatirn = Fatirn \* 2 58. Jika (FatirCurrent->FatirLeft != NULL), maka kerjakan baris 59 s.d 60, kalau tidak kerjakan baris 61 59. Fatirj++ 60. FatirQ[Fatirj] = FatirCurrent->FatirLeft 61. Jika (FatirCurrent->FatirRight != NULL), maka kerjakan baris 62 s.d 63, kalau tidak kerjakan baris 64 62. Fatirj++ 63. FatirQ[Fatirj] = FatirCurrent->FatirRight 64. Fatiri++ 65. Mendeklarasikan struct (struct FatirNode(struct FatirNode \*FatirLeft, char FatirINFO, struct FatirNode \*FatirRight)) 66. Mendefinisikan struct (typedef struct FatirNode FatirSimpul, FatirSimpul \*FatirRoot, \*FatirP, \*FatirCurrent, FatirSimpul \*FatirQ[129]) 67. Memanggil function FatirInisialisasi() 68. Memasukkan isi/nilai variabel FatirX 69. Memanggil function FatirBuatSimpul(FatirX) 70. Memanggil function FatirBuatSimpulAkar() 71. Memanggil function FatirInsertUrutNomor() 72. Memanggil function FatirBacaUrutNomor() 73. Selesai |



|  |
| --- |
| **Praktikum 11** |
| **Pseudocode (Nomor 02)**  **Kamus/Deklarasi Variabel Function Fatirstack::Fatirstack()**  -  **Algoritma/Deskripsi Function Fatirstack::Fatirstack()**  Fatirtop = NULL  **Kamus/Deklarasi Variabel Function Fatirstack::Fatirpush(int Fatirdata)**  **-**  **Algoritma/Deskripsi Function Fatirstack::Fatirpush(int Fatirdata)**  Fatirnode \*Fatirp  if ((Fatirp = (Fatirnode\*)malloc(sizeof(Fatirnode))) == NULL)  print("Memori Penuh")  exit(0)  endif  Fatirp = new Fatirnode  Fatirp->Fatirinfo = Fatirdata  Fatirp->Fatirnext = NULL  if (Fatirtop != NULL)  Fatirp->Fatirnext = Fatirtop  endif  Fatirtop = Fatirp  **Kamus/Deklarasi Variabel Function Fatirstack::Fatirpop()**  Fatirvalue = int  **Algoritma/Deskripsi Function Fatirstack::Fatirpop()**  struct Fatirnode \*Fatirtemp  if (Fatirtop == NULL)  print("Stack Kosong") else  Fatirtemp = Fatirtop  Fatirtop = Fatirtop->Fatirnext  Fatirvalue = Fatirtemp->Fatirinfo  delete Fatirtemp  endif  return Fatirvalue  **Kamus/Deklarasi Variabel Function** **Fatirstack::FatirisEmpty()**  **-**  **Algoritma/Deskripsi Function Fatirstack::FatirisEmpty()**  return (Fatirtop == NULL)  **Kamus/Deklarasi Variabel Function** **Fatirstack::Fatirdisplay()**  **-**  **Algoritma/Deskripsi Function Fatirstack::Fatirdisplay()**  struct Fatirnode \*Fatirp = Fatirtop  if (Fatirtop == NULL)  print("Tidak ada tampilan”)  else  while (Fatirp != NULL)  print(Fatirp->Fatirinfo)  Fatirp = Fatirp->Fatirnext  endwhile  **Kamus/Deklarasi Variabel Function FatirGraph::FatirGraph(int Fatirsize)**  Fatiri, Fatirj = int  **Algoritma/Deskripsi Function FatirGraph::FatirGraph(int Fatirsize)**  if (Fatirsize < 2)  Fatirn = 2  else  Fatirn = Fatirsize  endif  FatirA = new int \*[Fatirn]  for (Fatiri = 0; Fatiri < Fatirn; ++Fatiri)  FatirA[Fatiri] = new int[Fatirn]  endfor  for (Fatiri = 0; Fatiri < Fatirn; ++Fatiri)  for (Fatirj = 0; Fatirj < Fatirn; ++Fatirj)  FatirA[Fatiri][Fatirj] = 0  endfor  endfor  **Kamus/Deklarasi Variabel Function** **FatirGraph::~FatirGraph()**  **-**  **Algoritma/Deskripsi Function** **FatirGraph::~FatirGraph()**  for (int Fatiri = 0; Fatiri < Fatirn; ++Fatiri)  delete[] FatirA[Fatiri]  delete[] FatirA  endfor  **Kamus/Deklarasi Variabel Function FatirGraph::FatirisConnected(int Fatirx, int Fatiry)**  **-**  **Algoritma/Deskripsi Function FatirGraph::FatirisConnected(int Fatirx, int Fatiry)**  return (FatirA[Fatirx - 1][Fatiry - 1] == 1)  **Kamus/Deklarasi Variabel Function** **FatirGraph::FatiraddEdge(int Fatirx, int Fatiry)**  **-**  **Algoritma/Deskripsi Function FatirGraph::FatiraddEdge(int Fatirx, int Fatiry)**  FatirA[Fatirx - 1][Fatiry - 1] = FatirA[Fatiry - 1][Fatirx - 1] = 1  **Kamus/Deklarasi Variabel Function FatirGraph::FatirDFS(int Fatirx, int required)**  Fatiri = int  **Algoritma/Deskripsi Function FatirGraph::FatirDFS(int Fatirx, int required)**  Fatirstack Fatirs  bool \*Fatirvisited = new bool[Fatirn + 1]  for (Fatiri = 0; Fatiri <= Fatirn; Fatiri++)  Fatirvisited[Fatiri] = false  endfor  Fatirs.Fatirpush(Fatirx)  Fatirvisited[Fatirx] = true  if (Fatirx == required)  return  endif  print(Fatirx)  while (!Fatirs.FatirisEmpty())  int Fatirk = Fatirs.Fatirpop()  if (Fatirk == required)  break  endif  print(Fatirk)  for (Fatiri = Fatirn; Fatiri >= 1; --Fatiri)  if (FatirisConnected(Fatirk, Fatiri) && !Fatirvisited[Fatiri])  Fatirs.Fatirpush(Fatiri)  Fatirvisited[Fatiri] = true  endif  endfor  endwhile  delete[] Fatirvisited  **Kamus/Deklarasi Variabel**  **-**  **Algoritma/Deskripsi**  struct Fatirnode  int Fatirinfo  struct Fatirnode \*Fatirnext  class Fatirstack  struct Fatirnode \*Fatirtop  public:  Fatirstack()  void Fatirpush(int)  int Fatirpop()  bool FatirisEmpty()  void Fatirdisplay()  class FatirGraph  private:  int Fatirn  int \*\*FatirA  public:  FatirGraph(int Fatirsize = 2)  ~FatirGraph()  bool FatirisConnected(int, int)  void FatiraddEdge(int Fatirx, int Fatiry)  void FatirDFS(int Fatirx, int required)  FatirGraph Fatirg(8)  Fatirg.FatiraddEdge(1, 2)  Fatirg.FatiraddEdge(1, 3)  Fatirg.FatiraddEdge(1, 4)  Fatirg.FatiraddEdge(2, 5)  Fatirg.FatiraddEdge(3, 6)  Fatirg.FatiraddEdge(4, 7)  Fatirg.FatiraddEdge(4, 8)  Fatirg.FatirDFS(1, 8) |
| **Algoritma/Bahasa Natural (Nomor 02)**   1. Membuat function Fatirstack::Fatirstack() 2. Fatirtop = NULL 3. Membuat function Fatirstack::Fatirpush(int Fatirdata) 4. Mendefinisikan struct (Fatirnode \*Fatirp) 5. Jika ((Fatirp = (Fatirnode\*)malloc(sizeof(Fatirnode))) == NULL), maka kerjakan baris 6 s.d 7, kalau tidak kerjakan baris 8 6. Menampilkan (“Memori Penuh”) 7. exit(0) 8. Fatirp = new Fatirnode 9. Fatirp->Fatirinfo = Fatirdata 10. Fatirp->Fatirnext = NULL 11. Jika (Fatirtop != NULL), maka kerjakan baris 12, kalau tidak kerjakan baris 13 12. Fatirp->Fatirnext = Fatirtop 13. Fatirtop = Fatirp 14. Membuat function Fatirstack::Fatirpop() 15. Mendefinisikan struct (struct Fatirnode \*Fatirtemp) 16. Jika (Fatirtop == NULL), maka kerjakan baris 17, kalau tidak kerjakan baris 18 s.d 21 17. Menampilkan ("Stack Kosong") 18. Fatirtemp = Fatirtop 19. Fatirtop = Fatirtop->Fatirnext 20. Fatirvalue = Fatirtemp->Fatirinfo 21. delete Fatirtemp 22. return Fatirvalue 23. Membuat function Fatirstack::FatirisEmpty() 24. return (Fatirtop == NULL) 25. Membuat function Fatirstack::Fatirdisplay() 26. Mendefinisikan struct (struct Fatirnode \*Fatirp = Fatirtop) 27. Jika (Fatirtop == NULL), maka kerjakan baris 28, kalau tidak kerjakan baris 29 s.d 31 28. Menampilkan ("Tidak ada tampilan") 29. Selama (Fatirp != NULL), maka kerjakan baris 30 s.d 31, kalau tidak kerjakan baris 32 30. Menampilkan isi/nilai variabel (Fatirp->Fatirinfo) 31. Fatirp = Fatirp->Fatirnext 32. Membuat function FatirGraph::FatirGraph(int Fatirsize) 33. Jika (Fatirsize < 2), maka kerjakan baris 34, kalau tidak kerjakan baris 35 34. Fatirn = 2 35. Fatirn = Fatirsize 36. FatirA = new int \*[Fatirn] 37. Fatiri = 0 38. Selama (Fatiri < Fatirn), maka kerjakan baris 39 s.d 40, kalau tidak kerjakan baris 41 39. FatirA[Fatiri] = new int[Fatirn] 40. ++Fatiri 41. Fatiri = 0 42. Selama (Fatiri < Fatirn), maka kerjakan baris 43 s.d 47, kalau tidak kerjakan baris 48 43. Fatirj = 0 44. Selama (Fatirj < Fatirn), maka kerjakan baris 45 s.d 46, kalau tidak kerjakan baris 47 45. FatirA[Fatiri][Fatirj] = 0 46. ++Fatirj 47. ++Fatiri 48. Membuat function FatirGraph::~FatirGraph() 49. int Fatiri = 0 50. Selama (Fatiri < Fatirn), maka kerjakan baris 51 s.d 52, kalau tidak kerjakan baris 53 51. delete[] FatirA[Fatiri] 52. ++Fatiri 53. delete[] FatirA 54. Membuat function FatirGraph::FatirisConnected(int Fatirx, int Fatiry) 55. return (FatirA[Fatirx - 1][Fatiry - 1] == 1) 56. Membuat function FatirGraph::FatiraddEdge(int Fatirx, int Fatiry) 57. FatirA[Fatirx - 1][Fatiry - 1] = FatirA[Fatiry - 1][Fatirx - 1] = 1 58. Membuat function FatirGraph::FatirDFS(int Fatirx, int required) 59. Mendefinisikan struct (Fatirstack Fatirs) 60. bool \*Fatirvisited = new bool[Fatirn + 1] 61. Fatiri = 0 62. Selama (Fatiri <= Fatirn), maka kerjakan baris 63 s.d 64, kalau tidak kerjakan baris 65 63. Fatirvisited[Fatiri] = false 64. Fatiri++ 65. Menampilkan pengaksesan anggota class (Fatirs.Fatirpush(Fatirx)) 66. Fatirvisited[Fatirx] = true 67. Jika (Fatirx == required), maka kerjakan baris 68, kalau tidak kerjakan baris 69 68. return 69. Menampilkan isi/nilai variabel Fatirx 70. Selama (!Fatirs.FatirisEmpty()), maka kerjakan baris 71 s.d 80, kalau tidak kerjakan baris 81 71. int Fatirk = Fatirs.Fatirpop() 72. Jika (Fatirk == required), maka kerjakan baris 73, kalau tidak kerjakan baris 74 73. break 74. Menampilkan isi/nilai variabel Fatirk 75. Fatiri = Fatirn 76. Selama (Fatiri >= 1), maka kerjakan baris 77 s.d 80, kalau tidak kerjakan baris 81 77. Jika (FatirisConnected(Fatirk, Fatiri) && !Fatirvisited[Fatiri]), maka kerjakan baris 78 s.d 79, kalau tidak kerjakan baris 80 78. Fatirs.Fatirpush(Fatiri) 79. Fatirvisited[Fatiri] = true 80. --Fatiri 81. delete[] Fatirvisited 82. Mendeklarasikan struct (struct Fatirnode(int Fatirinfo, struct Fatirnode \*Fatirnext)) 83. Mendeklarasikan class (class Fatirstack(struct Fatirnode \*Fatirtop)) 84. Mendeklarasikan class dengan tipe akses public (class Fatirstack(Fatirstack(), void Fatirpush(int), int Fatirpop(), bool FatirisEmpty(), void Fatirdisplay())) 85. Mendeklarasikan class dengan tipe akses private (class FatirGraph(int Fatirn, int \*\*FatirA)) 86. Mendeklarasikan class dengan tipe akses public (class FatirGraph(FatirGraph(int Fatirsize = 2), ~FatirGraph(), bool FatirisConnected(int, int), void FatiraddEdge(int Fatirx, int Fatiry), void FatirDFS(int Fatirx, int required))) 87. Mendefinisikan class (FatirGraph Fatirg(8)) 88. Menampilkan pengaksesan anggota class (Fatirg.FatiraddEdge(1, 2)) 89. Menampilkan pengaksesan anggota class (Fatirg.FatiraddEdge(1, 3)) 90. Menampilkan pengaksesan anggota class (Fatirg.FatiraddEdge(1, 4)) 91. Menampilkan pengaksesan anggota class (Fatirg.FatiraddEdge(2, 5)) 92. Menampilkan pengaksesan anggota class (Fatirg.FatiraddEdge(3, 6)) 93. Menampilkan pengaksesan anggota class (Fatirg.FatiraddEdge(4, 7)) 94. Menampilkan pengaksesan anggota class (Fatirg.FatiraddEdge(4, 8)) 95. Menampilkan pengaksesan anggota class (Fatirg.FatirDFS(1, 8)) 96. Selesai |



|  |
| --- |
| **Praktikum 12** |
| **Pseudocode (Nomor 01)**  **Kamus/Deklarasi Variabel Function Fatirinithashtab()**  -  **Algoritma/Deskripsi Function Fatirinithashtab()**  for (int Fatiri = 0; Fatiri < FatirHASHSIZE; Fatiri++)  Fatirhashtab[Fatiri] = NULL  endfor  **Kamus/Deklarasi Variabel Function FatirmyHash(char\* Fatirs)**  **-**  **Algoritma/Deskripsi Function FatirmyHash(char\* Fatirs)**  unsigned int Fatirh = 0  for (; \*Fatirs; Fatirs++)  Fatirh = \*Fatirs + Fatirh \* 31  endfor  return Fatirh % FatirHASHSIZE  **Kamus/Deklarasi Variabel Function Fatirnode\* Fatirlookup(char\* Fatirn)**  **-**  **Algoritma/Deskripsi Function Fatirnode\* Fatirlookup(char\* Fatirn)**  unsigned int Fatirhi = FatirmyHash(Fatirn)  Fatirnode\* Fatirnp = Fatirhashtab[Fatirhi]  for (; Fatirnp != NULL; Fatirnp = Fatirnp->Fatirnext)  if (!strcmp(Fatirnp->Fatirnama, Fatirn))  return Fatirnp  endif  endfor  return NULL  **Kamus/Deklarasi Variabel Function** **Fatirm\_strdup(const char\* Fatiro)**  **-**  **Algoritma/Deskripsi Function Fatirm\_strdup(const char\* Fatiro)**  int Fatirl = strlen(Fatiro) + 1  char\* Fatirns = (char\*)malloc(Fatirl \* sizeof(char))  strcpy(Fatirns, Fatiro)  if (Fatirns == NULL)  return NULL  else  return Fatirns  endif  **Kamus/Deklarasi Variabel Function get(const char\* Fatirnama)**  **-**  **Algoritma/Deskripsi Function get(const char\* Fatirnama)**  Fatirnode\* Fatirn = Fatirlookup(const\_cast<char\*>(Fatirnama))  if (Fatirn == NULL)  return NULL  else  return Fatirn->Fatirdesc  endif  **Kamus/Deklarasi Variabel Function Fatirinstall(char\* Fatirnama, char\* Fatirdesc)**  -  **Algoritma/Deskripsi Function Fatirinstall(char\* Fatirnama, char\* Fatirdesc)**  unsigned int Fatirhi = FatirmyHash(Fatirnama)  Fatirnode\* Fatirnp  if ((Fatirnp = Fatirlookup(Fatirnama)) == NULL)  Fatirnp = (Fatirnode\*)malloc(sizeof(Fatirnode))  if (Fatirnp == NULL)  return 0  endif  Fatirnp->Fatirnama = Fatirm\_strdup(Fatirnama)  if (Fatirnp->Fatirnama == NULL)  return 0  endif  Fatirnp->Fatirnext = Fatirhashtab[Fatirhi]  Fatirhashtab[Fatirhi] = Fatirnp  else  free(Fatirnp->Fatirdesc)  endif  Fatirnp->Fatirdesc = Fatirm\_strdup(Fatirdesc)  if (Fatirnp->Fatirdesc == NULL)  return 0  endif  return 1  **Kamus/Deklarasi Variabel Function** **Fatirdisplaytable()**  **-**  **Algoritma/Deskripsi Function Fatirdisplaytable()**  for (int Fatiri = 0; Fatiri < FatirHASHSIZE; Fatiri++)  Fatirnode\* Fatirt = Fatirhashtab[Fatiri]  while (Fatirt != NULL)  print(Fatirt->Fatirnama)  print(Fatirt->Fatirdesc)  Fatirt = Fatirt->Fatirnext  endwhile  endfor  **Kamus/Deklarasi Variabel Function** **Fatircleanup()**  **-**  **Algoritma/Deskripsi Function Fatircleanup()**  Fatirnode\* Fatirnp, \* Fatirt  for (int Fatiri = 0; Fatiri < FatirHASHSIZE; Fatiri++)  if (Fatirhashtab[Fatiri] != NULL)  Fatirnp = Fatirhashtab[Fatiri]  while (Fatirnp != NULL)  Fatirt = Fatirnp->Fatirnext  free(Fatirnp->Fatirnama)  free(Fatirnp->Fatirdesc)  free(Fatirnp)  Fatirnp = Fatirt  endwhile  endif  endfor  **Kamus/Deklarasi Variabel Function** **Fatirdata()**  Fatiri = int  **Algoritma/Deskripsi Function** **Fatirdata()**  const char\* Fatirnamas[] = {"Fatirnama ", "alamat ", "telpon ", "cita-cita ", "sekolah "}  const char\* Fatirdescs[] = {"Ichan", "Depok", "089999991", "Dosen", "Mahasiswa"}  Fatirinithashtab()  for (Fatiri = 0; Fatiri < 5; Fatiri++)  Fatirinstall(const\_cast<char\*>(Fatirnamas[Fatiri]), const\_cast<char\*>(Fatirdescs[Fatiri]))  endfor  print(("nama "), ("alamat "), ("telpon "), ("cita-cita "), ("sekolah "))  **Kamus/Deklarasi Variabel**  Fatirpilihan = int  **Algoritma/Deskripsi**  typedef struct Fatir\_node  char\* Fatirnama  char\* Fatirdesc  struct Fatir\_node\* Fatirnext  Fatirnode  static Fatirnode\* Fatirhashtab[FatirHASHSIZE]  do  input(Fatirpilihan)  switch (Fatirpilihan)  case 1:  Fatirdata()  case 2:  Fatirdisplaytable();  case 3:  Fatircleanup();  case 4:  print("Terima Kasih! ")  while (Fatirpilihan != 4)  endwhile |
| **Algoritma/Bahasa Natural (Nomor 01)**   1. Membuat function Fatirinithashtab() 2. int Fatiri = 0 3. Selama (Fatiri < FatirHASHSIZE), maka kerjakan baris 4 s.d 5, kalau tidak kerjakan baris 6 4. Fatirhashtab[Fatiri] = NULL 5. Fatiri++ 6. Membuat function FatirmyHash(char\* Fatirs) 7. unsigned int Fatirh = 0 8. Selama (\*Fatirs), maka kerjakan baris 9 s.d 10, kalau tidak kerjakan baris 11 9. Fatirh = \*Fatirs + Fatirh \* 31 10. Fatirs++ 11. return Fatirh % FatirHASHSIZE 12. Membuat function Fatirnode\* Fatirlookup(char\* Fatirn) 13. unsigned int Fatirhi = FatirmyHash(Fatirn) 14. Fatirnode\* Fatirnp = Fatirhashtab[Fatirhi] 15. Selama (Fatirnp != NULL), maka kerjakan baris 16 s.d 18, kalau tidak kerjakan baris 19 16. Jika (!strcmp(Fatirnp->Fatirnama, Fatirn)), maka kerjakan baris 17, kalau tidak kerjakan baris 18 17. return Fatirnp 18. Fatirnp = Fatirnp->Fatirnext 19. return NULL 20. Membuat function Fatirm\_strdup(const char\* Fatiro) 21. int Fatirl = strlen(Fatiro) + 1 22. char\* Fatirns = (char\*)malloc(Fatirl \* sizeof(char)) 23. strcpy(Fatirns, Fatiro) 24. Jika (Fatirns == NULL), maka kerjakan baris 25, kalau tidak kerjakan baris 26 25. return NULL 26. return Fatirns 27. Membuat function get(const char\* Fatirnama) 28. Fatirnode\* Fatirn = Fatirlookup(const\_cast<char\*>(Fatirnama)) 29. Jika (Fatirn == NULL), maka kerjakan baris 30, kalau tidak kerjakan baris 31 30. return NULL 31. return Fatirn->Fatirdesc 32. Membuat function Fatirinstall(char\* Fatirnama, char\* Fatirdesc) 33. unsigned int Fatirhi = FatirmyHash(Fatirnama) 34. Mendefinisikan struct (Fatirnode\* Fatirnp) 35. Jika ((Fatirnp = Fatirlookup(Fatirnama)) == NULL), maka kerjakan baris 36 s.d 44, kalau tidak kerjakan baris 45 36. Fatirnp = (Fatirnode\*)malloc(sizeof(Fatirnode)) 37. Jika (Fatirnp == NULL), maka kerjakan baris 38, kalau tidak kerjakan baris 39 38. return 0 39. Fatirnp->Fatirnama = Fatirm\_strdup(Fatirnama) 40. Jika (Fatirnp->Fatirnama == NULL), maka kerjakan baris 40, kalau tidak kerjakan baris 41 41. return 0 42. Fatirnp->Fatirnext = Fatirhashtab[Fatirhi] 43. Fatirhashtab[Fatirhi] = Fatirnp 44. free(Fatirnp->Fatirdesc) 45. Fatirnp->Fatirdesc = Fatirm\_strdup(Fatirdesc) 46. Jika (Fatirnp->Fatirdesc == NULL), maka kerjakan baris 47, kalau tidak kerjakan baris 48 47. return 0 48. return 1 49. Membuat function Fatirdisplaytable() 50. int Fatiri = 0 51. Selama (Fatiri < FatirHASHSIZE), maka kerjakan baris 52 s.d 57, kalau tida kerjakan baris 58 52. Fatirnode\* Fatirt = Fatirhashtab[Fatiri] 53. Selama (Fatirt != NULL), maka kerjakan baris 54 s.d 56, kalau tidak kerjakan baris 57 54. Menampilkan isi/nilai variabel (Fatirt->Fatirnama) 55. Menampilkan isi/nilai variabel (Fatirt->Fatirdesc) 56. Fatirt = Fatirt->Fatirnext 57. Fatiri++ 58. Membuat function Fatircleanup() 59. Mendefinisikan static (Fatirnode\* Fatirnp, \* Fatirt) 60. int Fatiri = 0 61. Selama (Fatiri < FatirHASHSIZE), maka kerjakan baris 62 s.d 70, kalau tidak kerjakan baris 71 62. Jika (Fatirhashtab[Fatiri] != NULL), maka kerjakan baris 63 s.d 69, kalau tidak kerjakan baris 70 63. Fatirnp = Fatirhashtab[Fatiri] 64. Selama (Fatirnp != NULL), maka kerjakan baris 65 s.d 69, kalau tidak kerjakan baris 70 65. Fatirt = Fatirnp->Fatirnext 66. free(Fatirnp->Fatirnama) 67. free(Fatirnp->Fatirdesc 68. free(Fatirnp) 69. Fatirnp = Fatirt 70. Fatiri++ 71. Membuat function Fatirdata() 72. const char\* Fatirnamas[] = {"nama ", "alamat ", "telpon ", "cita-cita ", "sekolah "} 73. const char\* Fatirdescs[] = {"Ichan", "Depok", "089999991", "Dosen", "Mahasiswa"} 74. Memanggil function Fatirinithashtab() 75. Fatiri = 0 76. Selama (Fatiri < 5), maka kerjakan baris 77 s.d 78, kalau tidak kerjakan baris 79 77. Fatirinstall(const\_cast<char\*>(Fatirnamas[Fatiri]), const\_cast<char\*>(Fatirdescs[Fatiri])) 78. Fatiri++ 79. Menampilkan isi/nilai (("nama "),("alamat "),("telpon "),("cita-cita "), ("sekolah ")) 80. Mendeklarasikan struct (typedef struct Fatir\_node(char\* Fatirnama, char\* Fatirdesc, struct Fatir\_node\* Fatirnext)) 81. Mendefinisikan struct (Fatirnode) 82. static Fatirnode\* Fatirhashtab[FatirHASHSIZE] 83. Memasukkan isi/nilai variabel Fatirpilihan 84. Jika (Fatirpilihan=1), maka kerjakan baris 85, kalau tidak kerjakan baris 86 85. Memanggil function Fatirdata() 86. Jika (Fatirpilihan=2), maka kerjakan baris 87, kalau tidak kerjakan baris 88 87. Memanggil function Fatirdisplaytable() 88. Jika (Fatirpilihan=3), maka kerjakan baris 89, kalau tidak kerjakan baris 90 89. Memanggil function Fatircleanup() 90. Jika (Fatirpilihan=4), maka kerjakan baris 91, kalau tidak kerjakan baris 92 91. Menampilkan ("Terima Kasih! ") 92. Selama (Fatirpilihan != 4), maka kerjakan baris 83 s.d 91, kalau tidak kerjakan baris 93 93. Selesai |



**Bab 2**

**Tugas Besar**

**2.1 Pembahasan Tugas Besar**

**2.1.1 Pseudocode**

|  |
| --- |
| **Kamus/Deklarasi Variabel Function insertNode(data newData)**  **-**  **Algoritma/Deskripsi Function insertNode(data newData)**  Node \*newNode = new Node  newNode->info = new data  newNode->info->token = newData.token  newNode->info->pulsa = newData.pulsa  newNode->info->hasil = newData.hasil  newNode->next = NULL  if (head == NULL)  head = newNode  else  Node \*temp = head  while (temp->next != NULL)  temp = temp->next  endwhile  temp->next = newNode  endif  **Kamus/Deklarasi Variabel Function displayList()**  **-**  **Algoritma/Deskripsi Function displayList()**  Node \*temp = head  while (temp != NULL)  print((temp->info->token),(temp->info->pulsa),( temp->info->hasil))  temp = temp->next  endwhile  **Kamus/Deklarasi Variabel Function bubbleSortAscending(int arr[], int n)**  **-**  **Algoritma/Deskripsi Function bubbleSortAscending(int arr[], int n)**  for (int i = 0; i < n - 1; i++)  for (int j = 0; j < n - i - 1; j++)  if (arr[j] > arr[j + 1])  tukar(&arr[j], &arr[j + 1])  endif  endfor  endfor  **Kamus/Deklarasi Variabel Function tukar(int \*a, int \*b)**  **-**  **Algoritma/Deskripsi Function tukar(int \*a, int \*b)**  int temp = \*a  \*a = \*b  \*b = temp  **Kamus/Deklarasi Variabel Function sequentialSearch(int arr[], int n, int x)**  **-**  **Algoritma/Deskripsi Function sequentialSearch(int arr[], int n, int x)**  for (int i = 0; i < n; i++)  if (arr[i] == x)  return i  endif  endfor  return -1  **Kamus/Deklarasi Variabel Function push(int value)**  **-**  **Algoritma/Deskripsi Function push(int value)**  if (isFull())  print(“Stack penuh. Tidak dapat menambahkan elemen lagi.”)  else  top++  stack[top] = value  endif  **Kamus/Deklarasi Variabel Function pop()**  **-**  **Algoritma/Deskripsi Function pop()**  if (isEmpty())  print(“Stack kosong. Tidak ada elemen yang dapat dihapus.”)  return -1  else  int value = stack[top]  top--  return value  endif  **Kamus/Deklarasi Variabel Function isEmpty()**  **-**  **Algoritma/Deskripsi Function isEmpty()**  return top == -1  **Kamus/Deklarasi Variabel Function isFull()**  **-**  **Algoritma/Deskripsi Function isFull()**  return top == MAX\_SIZE - 1  **Kamus/Deklarasi Variabel Function peek()**  **-**  **Algoritma/Deskripsi Function peek()**  if (isEmpty())  return -1  else  return stack[top]  endif  **Kamus/Deklarasi Variabel Function isi()**  pulsa = int  **Algoritma/Deskripsi Function isi()**  input(pulsa)  push(pulsa)  **Kamus/Deklarasi Variabel Function pulsa()**  **-**  **Algoritma/Deskripsi Function pulsa()**  print(peek())  **Kamus/Deklarasi Variabel Function belitoken()**  token = int  **Algoritma/Deskripsi Function belitoken()**  input(token)  push(token)  **Kamus/Deklarasi Variabel Function hargatoken()**  n = int  **Algoritma/Deskripsi Function hargatoken()**  input(n)  int \*tokenPrices = new int[n]  for (int i = 0; i < n; i++)  print(i+1)  input(tokenPrices[i])  endfor  bubbleSortAscending(tokenPrices, n)  tampil(tokenPrices, n)  delete[] tokenPrices  **Kamus/Deklarasi Variabel Function tampil(int arr[], int n)**  **-**  **Algoritma/Deskripsi Function tampil(int arr[], int n)**  for (int i = 0; i < n; i++)  print((i+1),( arr[i]))  endfor  **Kamus/Deklarasi Variabel Function bayartagihan()**  n, tagihan, posisi = int  **Algoritma/Deskripsi Function bayartagihan()**  input(tagihan)  input(n)  int \*tokens = new int[n]  for (int i = 0; i < n; i++)  print(i+1)  input(tokens[i])  endfor  posisi = sequentialSearch(tokens, n, tagihan)  if (posisi != -1)  print(tokens[posisi])  data newData  newData.token = tokens[posisi]  newData.pulsa = pop()  newData.hasil = newData.pulsa - newData.token  insertNode(newData)  else  print(tagihan)  endif  delete[] tokens  **Kamus/Deklarasi Variabel Function exitProgram()**  **-**  **Algoritma/Deskripsi Function exitProgram()**  print(“Terima kasih telah menggunakan program ini.”)  print(“Program berakhir.”)  **Kamus/Deklarasi Variabel**  insertNode(),displayList(),bubbleSortAscending(),tukar(),push(), isi(), pulsa(), belitoken(),hargatoken(), tampil(),bayartagihan(), exitProgram() = void  sequentialSearch(), stack[MAX\_SIZE], pop(), peek(), pilih = int  isEmpty(),isFull() = bool  **Algoritma/Deskripsi**  struct data  int token = 0  int pulsa = 0  int hasil = 0  typedef struct data \*DataPtr  struct Node  DataPtr info  Node \*next  Node \*head = NULL  const int MAX\_SIZE = 100  int top = -1  while (true)  input(pilih)  switch (pilih)  case 1:  isi()  case 2:  pulsa()  case 3:  belitoken()  case 4:  hargatoken()  case 5:  bayartagihan()  case 6:  exitProgram()  default:  print("Pilihan menu tidak ditemukan")  endwhile |

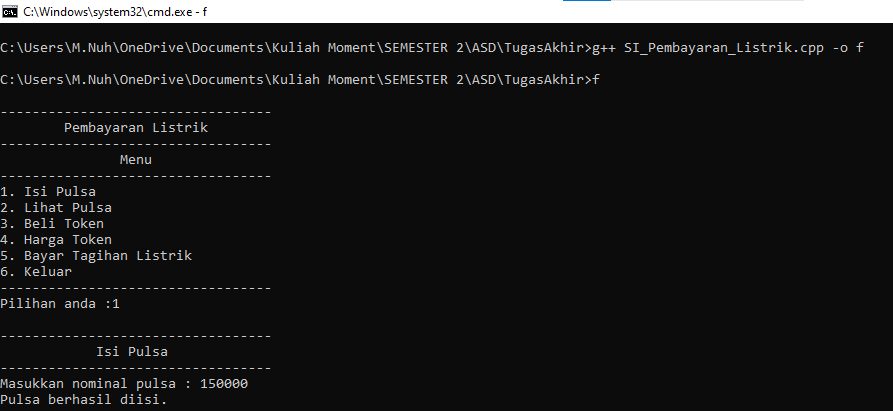
**2.1.2 Algoritma/Bahasa Natural**

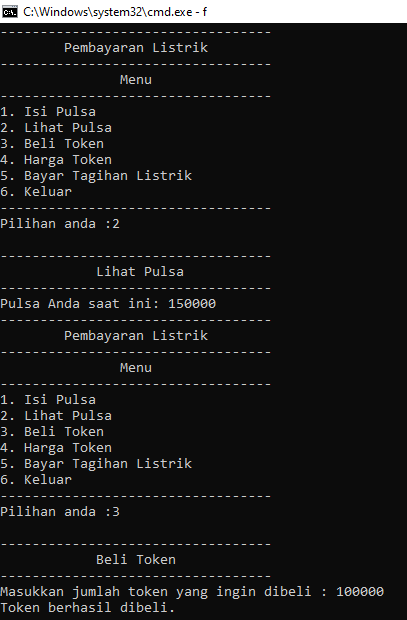
|  |
| --- |
| 1. Membuat function insertNode(data newData) 2. Node \*newNode = new Node 3. newNode->info = new data 4. newNode->info->token = newData.token 5. newNode->info->pulsa = newData.pulsa 6. newNode->info->hasil = newData.hasil 7. newNode->next = NULL 8. Jika (head == NULL), maka kerjakan baris 9, kalau tidak kerjakan baris 10 s.d 13 9. head = newNode 10. Node \*temp = head 11. Selama (temp->next != NULL), maka kerjakan baris 12, kalau tidak kerjakan baris 13 12. temp = temp->next 13. temp->next = newNode 14. Membuat function displayList() 15. Node \*temp = head 16. Selama (temp != NULL), maka kerjakan baris 17 s.d 18, kalau tidak kerjakan baris 19 17. Menampilkan isi/nilai variabel ((temp->info->token),( temp->info->pulsa),( temp->info->hasil)) 18. temp = temp->next 19. Membuat function bubbleSortAscending(int arr[], int n) 20. int i = 0 21. Selama (i < n), maka kerjakan baris 22 s.d 27, kalau tidak kerjakan baris 28 22. int j = 0 23. Selama (j < n - i - 1), maka kerjakan baris 24 s.d 26, kalau tidak kerjakan baris 27 24. Jika (arr[j] > arr[j + 1]), maka kerjakan baris 25, kalau tidak kerjakan baris 26 25. tukar(&arr[j], &arr[j + 1]) 26. j++ 27. i++ 28. Membuat function tukar(int \*a, int \*b) 29. int temp = \*a 30. \*a = \*b 31. \*b = temp 32. Membuat function sequentialSearch(int arr[], int n, int x) 33. int i = 0 34. Selama (i < n), maka kerjakan baris 35 s.d 37, kalau tidak kerjakan baris 38 35. Jika (arr[i] == x), maka kerjakan baris 36, kalau tidak kerjakan baris 37 36. return i 37. i++ 38. return -1 39. Membuat function push(int value) 40. Jika (isFull()), maka kerjakan baris 41, kalau tidak kerjakan baris 42 s.d 43 41. Menampilkan ("Stack penuh. Tidak dapat menambahkan elemen lagi.") 42. top++ 43. stack[top] = value 44. Membuat function pop() 45. Jika (isEmpty()), maka kerjakan baris 46, kalau tidak kerjakan baris 47 s.d 49 46. return -1 47. int value = stack[top] 48. top-- 49. return value 50. Membuat function isEmpty() 51. return top == -1 52. Membuat function isFull() 53. return top == MAX\_SIZE - 1 54. Membuat function peek() 55. Jika (isEmpty()), maka kerjakan baris 56, kalau tidak kerjakan baris 57 56. return -1 57. return stack[top] 58. Membuat function isi() 59. Memasukkan isi/nilai variabel pulsa 60. Memanggil function push(pulsa) 61. Membuat function pulsa() 62. Menampilkan isi/nilai function peek() 63. Membuat function belitoken() 64. Memasukkan isi/nilai variabel token 65. Memanggil function push(token) 66. Membuat function hargatoken() 67. Memasukkan is/nilai variabel n 68. int \*tokenPrices = new int[n] 69. int i = 0 70. Selama (i < n), maka kerjakan baris 71 s.d 73, kalau tidak kerjakan baris 74 71. Menampilkan isi/nilai variabel (i + 1) 72. Memasukkan isi/nilai variabel (tokenPrices[i]) 73. i++ 74. Memanggil function bubbleSortAscending(tokenPrices, n) 75. Memanggil function tampil(tokenPrices, n) 76. delete[] tokenPrices 77. Membuat function tampil(int arr[], int n) 78. int i = 0 79. Selama (i < n), maka kerjakan baris 80 s.d 81, kalau tidak kerjakan baris 82 80. Menampilkan isi/nilai variabel ((i + 1),(arr[i])) 81. i++ 82. Membuat function bayartagihan() 83. Memasukkan isi/nilai variabel tagihan 84. Memasukkan isi/nilai variabel n 85. int \*tokens = new int[n] 86. int i = 0 87. Selama (i < n), maka kerjakan baris 88 s.d 90, kalau tidak kerjakan baris 91 88. Menampilkan isi/nilai variabel (i + 1) 89. Memasukkan isi/nilai variabel (tokens[i]) 90. i++ 91. posisi = sequentialSearch(tokens, n, tagihan) 92. Jika (posisi != -1), maka kerjakan baris 93 s.d 98, kalau tidak kerjakan baris 99 93. Menampilkan isi/nilai variabel (tokens[posisi]) 94. Mendefinisikan struct (data newData) 95. newData.token = tokens[posisi] 96. newData.pulsa = pop() 97. newData.hasil = newData.pulsa - newData.token 98. insertNode(newData) 99. Menampilkan isi/nilai variabel tagihan 100. delete[] tokens 101. Membuat function exitProgram() 102. Menampilkan ("Terima kasih telah menggunakan program ini.") 103. Menampilkan ("Program berakhir.") 104. Mendeklarasikan struct data (int token = 0, int pulsa = 0, int hasil = 0) 105. Mendefinisikan struct data (typedef struct data \*DataPtr) 106. Mendeklarasikan struct Node (DataPtr info, Node \*next) 107. Node \*head = NULL 108. const int MAX\_SIZE = 100 109. int top = -1 110. Selama (true), maka kerjakan baris 111 s.d 125, kalau tidak kerjakan baris 126 111. Memasukkan isi/nilai variabel pilih 112. Jika (pilih==1), maka kerjakan baris 113, kalau tidak kerjakan baris 114 113. Memanggil function isi() 114. Jika (pilih==2), maka kerjakan baris 115, kalau tidak kerjakan baris 116 115. Memanggil function pulsa() 116. Jika (pilih==3), maka kerjakan baris 117, kalau tidak kerjakan baris 118 117. Memanggil function belitoken() 118. Jika (pilih==4), maka kerjakan baris 119, kalau tidak kerjakan baris 120 119. Memanggil function hargatoken() 120. Jika (pilih==5), maka kerjakan baris 121, kalau tidak kerjakan baris 122 121. Memanggil function bayartagihan() 122. Jika (pilih==6), maka kerjakan baris 123, kalau tidak kerjakan baris 124 123. Memanggil function exitProgram 124. Jika (pilih==default), maka kerjakan baris 125, kalau tidak kerjakan baris 126 125. Menampilkan ("Pilihan menu tidak ditemukan") 126. Selesai |

**2.1.3 Program (Source Code)**

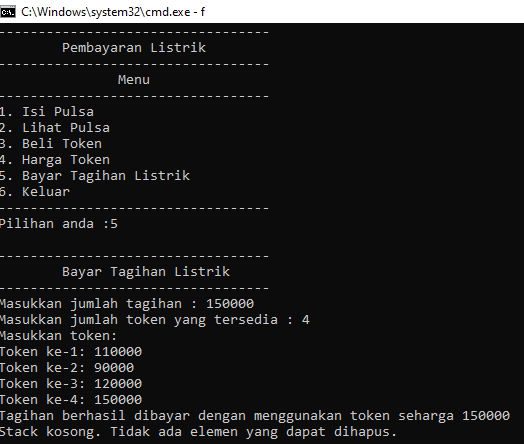
|  |
| --- |
| /\* Nama = Faathir Akbar Nugroho  NPM = 4522210033  Tugas Besar  No.2 SI Pembayaran Listrik dengan menggunakan Inputan  Materi - Struktur  - Pointer  - ADT SDL Linear Single Linked List  - Bubble Sort Ascending  - Sequential Search \*/  #include <iostream>  #include <iomanip>  #include <cstdlib>  using namespace std;  // Struktur  struct data  {  int token = 0;  int pulsa = 0;  int hasil = 0;  };  // Pointer Struktur  typedef struct data \*DataPtr;  // ADT SDL Linear Single Linked List  struct Node  {  DataPtr info;  Node \*next;  };  Node \*head = NULL;  void insertNode(data);  void displayList();  // Bubble Sort  void bubbleSortAscending(int[], int);  void tukar(int \*, int \*);  // Sequential Search  int sequentialSearch(int[], int, int);  // SDL Stack  const int MAX\_SIZE = 100;  int stack[MAX\_SIZE];  int top = -1;  void push(int);  int pop();  bool isEmpty();  bool isFull();  int peek();  void isi();  void pulsa();  void belitoken();  void hargatoken();  void tampil(int[], int);  void bayartagihan();  void exitProgram();  int main()  {  cout << endl;  int pilih;  while (true)  {  cout << "----------------------------------" << endl;  cout << " Pembayaran Listrik " << endl;  cout << "----------------------------------" << endl;  cout << " Menu " << endl;  cout << "----------------------------------" << endl;  cout << "1. Isi Pulsa " << endl;  cout << "2. Lihat Pulsa " << endl;  cout << "3. Beli Token " << endl;  cout << "4. Harga Token " << endl;  cout << "5. Bayar Tagihan Listrik " << endl;  cout << "6. Keluar " << endl;  cout << "----------------------------------" << endl;  cout << "Pilihan anda :";  cin >> pilih;  switch (pilih)  {  case 1:  isi();  break;  case 2:  pulsa();  break;  case 3:  belitoken();  break;  case 4:  hargatoken();  break;  case 5:  bayartagihan();  break;  case 6:  exitProgram();  return 0;  default:  cout << endl;  cout << "Pilihan menu tidak ditemukan" << endl;  }  }  }  // Implementasi ADT SDL Linear Single Linked List  void insertNode(data newData)  {  Node \*newNode = new Node;  newNode->info = new data;  newNode->info->token = newData.token;  newNode->info->pulsa = newData.pulsa;  newNode->info->hasil = newData.hasil;  newNode->next = NULL;  if (head == NULL)  {  head = newNode;  }  else  {  Node \*temp = head;  while (temp->next != NULL)  {  temp = temp->next;  }  temp->next = newNode;  }  }  void displayList()  {  Node \*temp = head;  cout << endl;  cout << "----------------------------------" << endl;  cout << " Data SDL Linear Linked List " << endl;  cout << "----------------------------------" << endl;  cout << "Token\tPulsa\tHasil" << endl;  cout << "----------------------------------" << endl;  while (temp != NULL)  {  cout << temp->info->token << "\t" << temp->info->pulsa << "\t" << temp->info->hasil << endl;  temp = temp->next;  }  cout << "----------------------------------" << endl;  }  // Implementasi Bubble Sort  void bubbleSortAscending(int arr[], int n)  {  for (int i = 0; i < n - 1; i++)  {  for (int j = 0; j < n - i - 1; j++)  {  if (arr[j] > arr[j + 1])  {  tukar(&arr[j], &arr[j + 1]);  }  }  }  }  void tukar(int \*a, int \*b)  {  int temp = \*a;  \*a = \*b;  \*b = temp;  }  // Implementasi Sequential Search  int sequentialSearch(int arr[], int n, int x)  {  for (int i = 0; i < n; i++)  {  if (arr[i] == x)  {  return i;  }  }  return -1;  }  // Implementasi SDL Stack  void push(int value)  {  if (isFull())  {  cout << "Stack penuh. Tidak dapat menambahkan elemen lagi." << endl;  }  else  {  top++;  stack[top] = value;  }  }  int pop()  {  if (isEmpty())  {  cout << "Stack kosong. Tidak ada elemen yang dapat dihapus." << endl;  return -1;  }  else  {  int value = stack[top];  top--;  return value;  }  }  bool isEmpty()  {  return top == -1;  }  bool isFull()  {  return top == MAX\_SIZE - 1;  }  int peek()  {  if (isEmpty())  {  cout << "Stack kosong. Tidak ada elemen yang dapat dilihat." << endl;  return -1;  }  else  {  return stack[top];  }  }  void isi()  {  int pulsa;  cout << endl;  cout << "----------------------------------" << endl;  cout << " Isi Pulsa " << endl;  cout << "----------------------------------" << endl;  cout << "Masukkan nominal pulsa : ";  cin >> pulsa;  push(pulsa);  cout << "Pulsa berhasil diisi." << endl;  }  void pulsa()  {  cout << endl;  cout << "----------------------------------" << endl;  cout << " Lihat Pulsa " << endl;  cout << "----------------------------------" << endl;  cout << "Pulsa Anda saat ini: " << peek() << endl;  }  void belitoken()  {  int token;  cout << endl;  cout << "----------------------------------" << endl;  cout << " Beli Token " << endl;  cout << "----------------------------------" << endl;  cout << "Masukkan jumlah token yang ingin dibeli : ";  cin >> token;  push(token);  cout << "Token berhasil dibeli." << endl;  }  void hargatoken()  {  int n;  cout << endl;  cout << "----------------------------------" << endl;  cout << " Harga Token " << endl;  cout << "----------------------------------" << endl;  cout << "Masukkan jumlah harga token yang ingin dilihat : ";  cin >> n;  int \*tokenPrices = new int[n];  cout << "Masukkan harga token:" << endl;  for (int i = 0; i < n; i++)  {  cout << "Token ke-" << i + 1 << ": ";  cin >> tokenPrices[i];  }  bubbleSortAscending(tokenPrices, n);  tampil(tokenPrices, n);  delete[] tokenPrices;  }  void tampil(int arr[], int n)  {  cout << endl;  cout << "----------------------------------" << endl;  cout << " Harga Token Terurut " << endl;  cout << "----------------------------------" << endl;  cout << "Token\tHarga" << endl;  cout << "----------------------------------" << endl;  for (int i = 0; i < n; i++)  {  cout << i + 1 << "\t" << arr[i] << endl;  }  cout << "----------------------------------" << endl;  }  void bayartagihan()  {  int n, tagihan, posisi;  cout << endl;  cout << "----------------------------------" << endl;  cout << " Bayar Tagihan Listrik " << endl;  cout << "----------------------------------" << endl;  cout << "Masukkan jumlah tagihan : ";  cin >> tagihan;  cout << "Masukkan jumlah token yang tersedia : ";  cin >> n;  int \*tokens = new int[n];  cout << "Masukkan token:" << endl;  for (int i = 0; i < n; i++)  {  cout << "Token ke-" << i + 1 << ": ";  cin >> tokens[i];  }  posisi = sequentialSearch(tokens, n, tagihan);  if (posisi != -1)  {  cout << "Tagihan berhasil dibayar dengan menggunakan token seharga " << tokens[posisi] << endl;  data newData;  newData.token = tokens[posisi];  newData.pulsa = pop();  newData.hasil = newData.pulsa - newData.token;  insertNode(newData);  }  else  {  cout << "Tidak ada token yang cukup untuk membayar tagihan sebesar " << tagihan << endl;  }  delete[] tokens;  }  void exitProgram()  {  cout << endl;  cout << "Terima kasih telah menggunakan program ini." << endl;  cout << "Program berakhir." << endl;  } |

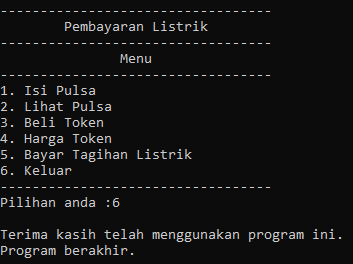
**2.1.4 Command Prompt Capture**

****

****

****

****

****

**Bab 3**

**Penutup**

**3.1 Kesimpulan**

Berdasarkan program dan pembahasan laporan tugas besar ini kita mempelajari tentang pemrograman Bahasa C++ yaitu Struktur, Struktur&Pointer, ADT SDL Linear Single Linked List, SDL Stack, Bubble Sort secara Ascending, dan Sequential Search serta membuat program untuk SI Pembayaran Listrik. Dalam pembahasan laporan tugas besar ini kita juga dapat lebih memahami perintah-perintah dan fungsi-fungsi yang ada pada C++.

**3.2 Kritik dan Saran**

Berdasarkan hasil dari program dan laporan tugas besar ini masih terdapat kekurangan seperti penggunaan materi yang tidak terlalu berguna untuk program serta masih kurang pemahaman tentang pembuatan program C++ yang harus dipelajari lagi lebih dalam.

**DAFTAR PUSTAKA**

Nursari, Sri Rezeki Candra. 2022. “Algoritma dan Struktur Data Gasal 2022/2023”.

http://elearning.teknik.univpancasila.ac.id/ (diakses pada bulan Juni)