Nama : Anak Agung Indi Kusuma Putra

NIM : 2205551079

Kelompok : 23

|  |
| --- |
| MODUL II PENYELEKSIAN KONDISI DAN PERULANGAN |

**Tujuan**

1. Mengetahui dan memahami *selection if, if-else,* dan *switch* sekaligus dapat memahami sintaks penulisannya dalam Bahasa C/C++ serta bisa mengaplikasikannya ke dalam *flowchart*.
2. Mengetahui dan memahami *repetition for, while,* dan *do-while* sekaligus dapat memahami sintaks penulisannya dalam Bahasa C/C++ serta bisa mengaplikasikannya ke dalam *flowchart*.
3. Mengetahui dan memahami metode perulangan rekursif dan iteratif serta memahami perbedaan dari kedua metode tersebut.
4. Mengetahui dan memahami tentang konsep *pseudocode* (notasi algoritma), *flowchart* dan *trace* serta mengimplementasikannya ke program yang kita buat.

**Tugas Pendahuluan**

1. Jelaskan dan berikan contoh penggunaan *selection if, if-else,* dan *switch,* serta buatlah contoh *flowchart* serta sintaks penulisannya dalam C/C++!
2. Jelaskan dan berikan contoh penggunaan *repetition for, while,* dan *do-while* serta buatlah contoh *flowchart* serta sintaks penulisannya dalam C/C++!
3. Jelaskan perbedaan metode perulangan rekursif dan iteratif!
4. Buatlah diagram alir (*flowchart*), *pseudecode* (notasi algoritma), dan masing-masing fungsi/prosedur pendukung yang digunakan!

**Jawaban**

1. **Selection**

Selection dalam pemrograman adalah instruksi dalam pemrograman untuk menentukan menjalankan suatu perintah berdasarkan kondisi tertentu. Bisa dibilang ini merupakan percabangan dalam sebuah program dimana hanya satu perintah yang akan dijalankan dari beberapa kondisi yang mungkin.

* 1. **Selection If**

Selection *if* adalah pengkondisian yang memiliki satu syarat kondisi agar perintah di dalamnya dapat dijalankan. Di dalam selection *if* terdapat *expression* yang menjadi parameter dan *statement* yang berisi perintah yang akan dijalankan apabila kondisi dalam *expression* bernilai *true*. Penulisan sintaks dalam selection *if* dituliskan sebagai if(expression){statement}. Untuk implementasinya pada sebuah program bisa dilihat dalam program bilangan genap dibawah.

|  |
| --- |
| #include <stdio.h>  int main()  {  int angka;  printf("masukan angka : ");  scanf("%d", &angka);  if(angka % 2 == 0){  printf("%d adalah bilangan genap",angka);  }  } |

Kode Program 1.1 Contoh Penulisan Sintaks Pada *Selection If*

Kode program 1.1 merupakan contoh pengimplementasian *selection if* dimana pada program diatas *statement* akan dijalankan jika angka yang diinputkan jika di modulus 2 akan menghasilkan nilai 0 yang membuat nilai pada *expression* bernilai *true*.

*flowchart* pada *selection if* sendiri dilambangkan dengan bentuk belah ketupat. Implementasi dari *selection-if* pada *flowchart* bisa dilihat pada *flowchart* program bilangan ganjil dibawah ini.

|  |
| --- |
|  |

Gambar 1.1 Contoh Pengaplikasian *Selection If* Pada *Flowchart*

Gambar 1.1 menampilkan contoh pengaplikasian *selection if* pada program bilangan genap. *Flowchart* diawali dengan *start* lalu diikuti dengan pengkondisian *if* dan jika kondisi bernilai *true* maka akan keluar *output* angka adalah bilangan genap. Lalu jika kondisi bernilai false maka program akan langsung berhenti.

* 1. **Selection If-Else**

*Selection If-Else* adalah pengkondisian yang konsepnya sama seperti *selection if*. Hanya saja pada *selection if-else* kita bisa memberikan perintah khususjika kondisi di dalam *if*  bernilai *false*. Jadi program tidak langsung berhenti. Penulisan sintaks pada *selection if-else* hampir sama seperti *selection if* hanya saja setelah statement *if* kita menambahan kondisi *else* dan statementnya. Singkatnya sintaks pada *selection if-else* ditulis dalam format if (expressions){statement}else{statement}. Contoh penulisan sintaks pada *selecton if*  bisa dilihat pada program untuk mendeteksi bilangan positif dibawah.

|  |
| --- |
| #include <stdio.h>  int main()  {  int angka;  printf("masukan angka : ");  scanf("%d", &angka);  if(angka > 0){  printf("%d adalah bilangan positif",angka);  }else{  printf("%d adalah bukan bilangan positif",angka);  }  } |

Kode Program 1.2 Contoh Penulisan Sintaks Pada *Selection If-Else*

Kode program 1.2 merupakan contoh pengimplementasian *selection if-else* dalam bahasa C/C++. Dimana jika angka lebih dari 0, maka *expression* dalam *if* bernilai *true* yang membuat *statement* didalamnya dijalankan dan mengeluarkan *output* angka adalah bilangan positif. Sedangkan jika *expression* dalam *if*  bernilai *false* maka *statement* dalam *else* yang akan dijalankan dan akan mengeluarkan *output* angka bukan bilangan positif.

*Flowchart* dalam *selection if-else* juga hampir sama dengan *selection if.* hanya saja yang membedakan adalah dalam *flowchart selection if-else* jika *expression* dalam *if*  bernilai *true* maka program tidak langsung berhenti, melainkan mengeksekusi *statement* dalam *else.* Contoh implementasi *selection if-else* pada *flowchart* dapat dilihat pada *flowchart* dibawah.

|  |
| --- |
|  |

Gambar 1.2 Contoh Pengaplikasian *Selection If-Else* Pada *Flowchart*

Gambar 1.2 diatas merupakan contoh pengaplikasian *flowhart* pada *selection if-else* pada program untuk mendeteksi bilangan positif. *Flowchart* diawali dengan menginput nilai angka lalu masuk ke dalam pengkondisian, jika *expression* dalam *if*  bernilai *true* maka *statement* didalamnya akan dijalankan dan akan mengeluarkan *output* angka adalah bilangan positif. Namun jika *expression* dalam *if* bernilai *false* maka *statement* dalam else yang akan dijalankan dan akan mengeluarkan *output* angka bukan bilangan positif.

* 1. Switch

*Switch* adalah pengkondisian yang digunakan untuk membandingkan nilai dari variabel dengan beberapa nilai yang biasa disebut dengan *case*. Dan nanti variabel tersebut akan dibandingkan dengan *case* yang ada secara bergantian sampai perbandingan yang dilakukan bernilai *true* atau saat nilai variabel dan *case* sama. Sintaks yang digunakan pada *switch* memiliki format berikut. Switch(expression){case nilai case: statement}. Contoh penulisan sintaks *selection switch* pada bahasa C/C++ bisa dilihat pada program untuk menentukan hari berdasarkan urutannya dibawah ini.

|  |
| --- |
| #include <stdio.h>  int main()  {  int angka1;  int angka2;  int hasil;  char operator;  printf("PROGRAM PENJUMLAHAN DAN PENGURANGAN\n");  printf("masukan angka pertama: ");  scanf("%d", &angka1);  printf("masukan operator (-/+) : ");  scanf("%s", &operator);  printf("masukan angka kedua: ");  scanf("%d", &angka2);    switch(operator){  case '-':  hasil = angka1 - angka2;  printf("%d - %d = %d",angka1, angka2,hasil);  break;  case '+':  hasil = angka1 + angka2;  printf("%d + %d = %d",angka1, angka2,hasil);  break;  default:  printf("input yang anda masukan salah!");  }  } |

Kode Program 1.3 Contoh Penulisan Sintaks Pada *Selection Switch*

Kode program 1.3 merupakan contoh penulisan sintaks pada *selection switch*. Dimana program diatas akan meminta input dari user dan nilai dari variabel yang diinputkan akan dibandingkan dengan nilai dalam *case*. Jika nilai dalam *case* bernilai *true* maka *statement* dibawahnya akan dijalankan. Begitu pula dengan *case*-*case* selanjutnya, namun jika nilai input tidak sesuai dengan seluruh *case* yang ada, maka perintah *default* yang akan dijalankan.

*Flowchart* pada *selection switch* dilambangkan dengan bentuk belah ketupat, lambingnya sama seperti *selection if* maupun *selection if-else* karena ketiganya termasuk dalam pengkondisian dan pengkondisian dalam *flowchart* dilambangkan dengan bentuk belah ketupat. Untuk lebih jelasnya implementasi *flowchart* dalam *selection switch* bisa dilihat pada *flowchart* dibawah.

|  |
| --- |
|  |

Gambar 1.3 Contoh Pengaplikasian *Selection Switch* pada *Flowchart*

Gambar 1.3 merupakan contoh pengaplikasian program *selection switch* kedalam bentuk *flowchart.* Program diatas akan meminta user menginput 2 angka dan 1 operator. Nantinya operator akan menjadi nilai variabel yang akan dibandingkan dengan nilai dalam *case*, jika nilai dalam *case* pertama bernilai *true* maka proses didalamnya akan dijalankan yang mengoperasikan penjumlahan pada angka1 dan angka2 lalu hasilnya dimasukan ke dalam variabel hasil dan setelah itu mengeluarkan *output* berupa pertambahan tadi. Proses yang sama juga terjadi saat *case* kedua bernilai *true*. Hanya saja pada *case* kedua yang dilakukan adalah proses pengurangan dan akan mengeluarkan *output* berupa pengurangan. Jika kedua *case* bernilai *false* maka akan langsung mengeluarkan *output* jika *input* anda salah.

1. **Repetition**

*Repetition* atau perulangan merupakan sebuah mekanisme dalam pemrograman dimana suatu perintah akan dijalankan secara terus menerus dalam kondisi tertentu yang bernilai *true* sampai kondisi tersebut bernilai *false*. Perulangan akan sangat berguna jika kita ingin mengeksekusi suatu perintah lebih dari satu atau bahkan hingga ratusan kali, daripada kita menulis perintah secara terus menerus yang tentunya tidak efesien memori dan waktu, kita bisa melakukannya dengan menuliskan program perulangan. Dalam Bahasa C/C++ terdapat 3 jenis perulangan yaitu *for, while,* dan *do-while.*

* 1. **Repetition For**

*Repetiton for* merupakan perulangan yang biasanya digunakan saat kita sudah mengetahui berapa kali program akan dijalankan, atau kita sudah mengetahui kapan program tersebut akan berakhir. Dalam *repetition for* menggunakan variabel kontrol yang akan dijadikan parameter yang nantinya dievaluasi dengan kondisi yang akan kita tentukan. Pertama variabel kontrol diinisialisasi, lalu akan dievaluasi dengan kondisi yang telah kita tentukan, jika bernilai *true* maka *statement* di dalamnya akan dieksekusi dan variabel kontrol akan di *increment*/*decrement*. Proses tersebut akan terus dilakukan berulang-ulang sampai hasil evaluasi variabel control oleh kondisi bernilai *false. Repetition for* memiliki format sintaks for(inisialisasi;kondisi;*increment/decrement*){ *statement}.* Contoh implementasi *repetition for* bisa dilihat pada program dibawah.

|  |
| --- |
| #include <stdio.h>  #include <unistd.h>  int main(){  for (int i = 10;i>0;i--){  printf("%d\n",i);  sleep(1);  }  printf("HAPPY NEW YEAR");  } |

Kode Program 2.1 Contoh Implementasi *Repetition For*

Kode Program 2.1 merupakan contoh implementasi dari *selection for* dalam Bahasa C/C++. Program diatas akan terus berjalan sampai kondisi bernilai *false* atau pada saat i <= 0. Dan program akan diulang sebanyak 10 kali karena I diinisialisasi awal dengan nilai 10 dan akan mengalami *decrement* setiap program nya berulang,dan batas kondisinya I > 0 yang berarti program akan terus diulang sampai I bernilai <= 0.

*Flowchart* pada *selection for* diawali dengan pengkondisian yang mengecek kondisi di dalam *repetition* apakah bernilai *true*  atau *false.* Lalu jika bernilai *true* maka *statement* selanjutnya akan dijalankan. Hal tersebut akan terus berulang sampai kondisi bernilai *false.* Contoh *flowchart selection* dapat dilihat pada *flowchart* dibawah.

|  |
| --- |
|  |

Gambar 2.1 Contoh Pengaplikasian *Repetition For* pada *Flowchart*

Gambar 2.1 diatas merupakan contoh pengaplikasian *repetition for* pada *flowchart*. Pada *flowchart* diatas *loop* akan terus berulang sampai nilai i <= 0, dan pada akhirnya akan mengeluarkan *output Happy New Year*.

* 1. **Repetition While**

*Repetition while* merupakan perulangan yang akan terus dijalankan pada kondisi tertentu sampai kondisi tersebut bernilai *false*. Pada *repetition while* lebih rentan untuk terjadi *infinite loop* karena tidak seperti *repetition for* yang terdapat pembatas dan kita tahu berapa kali perulangan tersebut akan dijalankan*,* karena biasanya kita menggunakan *repetition while* saat kita tidak mengetahui berapa kali perulangan tersebut akan dijalankan, dimana perulangan tersebut bergantung pada kondisi tertentu. *Repetition while* akan mengecek dulu suatu kondisi, apabila bernilai *true* maka baru *statement* didalamnya akan dijalankan. Contoh implementasi *repetition while* bisa kita lihat pada program dibawah.

|  |
| --- |
| #include <stdio.h>  #include <unistd.h>  int main()  {  int a;  int b;  printf("masukan angka awal : ");  scanf("%d",&a);  printf("masukan range akhir : ");  scanf("%d",&b);  printf("angka genap dari %d - %d :\n",a,b);  while (a<=b){  while(a % 2 == 0){  printf("%d ",a);  sleep(1);  a++;  }  a++;  }  printf("\nSELESAI!");  } |

Kode Program 2.2 Contoh Implementasi *Repetiton While*

Kode Program 2.2 merupakan contoh implementasi *repetition while* dalam bahasa C/C++. Program diatas adalah program untuk mencari bilangan genap dari suatu range angka. Dimana program akan meminta 2 *input* dari *user*. *Input* pertama merupakan range awal dan *input* kedua merupakan range akhir. Lalu akan masuk ke dalam *while* dimana ketika selama nilai a <= b maka *while* akan terus berjalan dan nilai akan terus ditambah 1 sampai nilainya > b. lalu *while* kedua akan mengecek apakah nilai a saat itu adalah bilangan genap atau tidak.

*Flowchart* dalam *repetition while* sebenarnya sama dengan *repetition for*. Contoh implementasi *flowchart* pada *repetition while* dapat dilihat pada contoh dibawah.

|  |
| --- |
|  |

Gambar 2.2 Contoh Implementasi *Repetition While* dalam *Flowchart*

Gambar 2.2 merupakan contoh implementasi *repetition while* ke dalam sebuah *flowchart.* Pada *flowchart* diatas *user* akan memasukan 2 *input*. Lalu masuk ke dalam *while.* Jika kondisi dalam *while* pertama bernilai *true* maka akan masuk ke kondisi kedua dan jika bernilai *true* maka nilai a akan di print. Setelah itu nilai a akan di *increment.* Proses ini akan terus berulang hingga nilai a >b.

* 1. **Repetition Do-While**

*Repetition do-while* sebenarnya sama dengan *repetition while.* Yang membedakannya adalah pada *repetition do-while* *statement* akan dijalankan dulu sekali baru setelah itu di kondisi dalam while akan di evaluasi. Jadi yang membedakan keduanya jika dalam *repetition do-while* *statement* pasti dijalankan setidaknya sekali sedankan dalam *repetition while* *statement* bisa saja tidak dijalankan jika kondisi didalamnya bernilai *false.* Contoh implementasi *repetition do-while* bisa dilihat pada contoh program dibawah.

|  |
| --- |
| #include <stdio.h>  #include <stdlib.h>  #include <string.h>  int main()  {  char ulang[1];  do{  printf("\n\n\n\n\t\t\t\tHELLO WORLD!\n\n\n\n");  system("pause");  printf("apakah anda ingin mengulang program? (y/n): ");  scanf("%s",&ulang);  system("cls");  }while (strcmp(ulang ,"y") == 0);  } |

Program 2.3 Contoh Implementasi *Repetition Do-While*

Program 2.3 merupakan contoh implementasi *repetition do-while* dalam Bahasa C/C++. Pada program diatas *statement* akan dijalankan terlebih dahulu baru kondisi dalam *while* akan di evaluasi. Pada *statement* akan diminta memasukan nilai ‘y’ atau ‘n’, jika nilai ‘y’ yang dimasukan maka kondisi dalam *while* akan bernilai *true* dan program akan kembali berulang.

*Flowchart* dalam *repetition do-while* sedikit berbeda dengan *repetition* yang lainnya. Dimana pada *repetition do-while* *flowchart* akan masuk ke *statement* terlebih dahulu baru mengecek kondisi dalam *while.* Contoh implementasi *repetition do-while* dapat dilihat pada *flowchart* dibawah.

|  |
| --- |
|  |

Gambar 2.3 Contoh Implementasi *Repetition Do-While* pada *Flowchart*

Gambar 2.3 merupakan contoh implementasi *repetition do-while* pada *flowchart.* Pada *flowchart* diatas *statement* akan terlebih dulu dijalankan sebelum masuk ke dalam kondisi. Pertama program akan mengeluarkan *output* Hello World. Lalu akan diminta untuk memasukan input ‘y/n’ dimana jika input ‘y’ yang dimasukan maka program akan kembali mengulang.

1. **Perulangan Rekursif dan Iteratif**

Perulangan merupakan suatu program yang akan menjalankan suatu instruksi secara berulang, terdapat 2 metode perulangan yang sering digunakan. Yaitu metode rekursif dan iteratif.

Metode perulangan rekursif adalah metode perulangan dimana sebuah fungsi akan terus memanggil dirinya sendiri, sedangkan metode perulangan iteratif adalah metode perulangan yang menggunakan mengulang suatu instruksi berdasarkan kondisi tertentu. Perbedaan mendasar diantara kedua metode ini adalah cara kerjanya, pada rekursif perulangan terjadi karena ada suatu fungsi yang memanggil dirinya sendiri, sedangkan pada iteratif perulangan terjadi melalui perintah yang akan mengulangan suatu instruksi bergantung pada kondisi tertentu, biasanya menggunakan *repetition for,while,* atau *do-while.*

## Flowchart, Pseudocode, dan Trace Flowchart Modul II

*Flowchart* merupakan suatu diagram yang menggambarkan proses, sistem, atau algoritma komputer. *Flowchart* sering digunakan dalam berbagai bidang, khususnya dalam pembuatan notasi algoritma, untuk mempermudah dalam mempelajari, merencanakan, atau pun berkomunikasi mengenai suatu proses algoritma yang kompleks. *Flowchart* menggunakan berbagai simbol dalam penggunaannya.

*Pseudocode* merupakan suatu metode ‘informal’ yang dilakukan dalam pemrograman untuk mendeskripsikan suatu notasi algoritma dalam bentuk kode program informal tanpa mengikuti peraturan *syntax* dari bahasa program mana pun. *Pseudocode* sering digunakan dalam pembuatan *outline* atau sketsa kasar dari sebuah program.

*Trace* adalah suatu metode yang digunakan untuk memeriksa alur dari sebuah algoritma. *Trace* yang berarti melacak akan memeriksa *input* yang diberikan, proses yang dijalankan oleh alur algoritma berdasarkan *input* tersebut, lalu *output* yang diberikan oleh algoritma setelah proses selesai. *Trace* berguna dalam memastikan apakah alur algoritma, khususnya *flowchart*, sudah sesuai atau belum. Berikut ini adalah *flowchart, pseudocode,* serta *trace* dari soal praktikum modul II.

### Flowchart, Pseudocode, dan Trace Flowchart Program Zodiak

Program zodiak merupakan suatu program yang mampu menentukan zodiak seseorang berdasarkan *input* tanggal dan bulan lahir yang diberikan oleh pengguna yang disesuaikan dengan ketentuan dari jangkauan tanggal zodiak tersebut. Berikut adalah *flowchart, pseudocode,* serta *trace flowchart* pada program zodiak.

#### Flowchart Program Zodiak

*Flowchart* digunakan untuk merepresentasikan algoritma atau langkah-langkah instruksi yang berurutan dalam sistem. *Flowchart* yang digunakan untuk menyelesaikan program zodiak yang dibuat dapat dilihat pada gambar dibawah ini.

|  |
| --- |
| C:\Users\User\AppData\Local\Microsoft\Windows\INetCache\Content.Word\Flowchart Zodiac-Halaman-1.jpg |

**Gambar 4.1** Flowchart Int Main Program Zodiak

Gambar 4.1 adalah *flowchart* dari subprogram utama dari program zodiak. Setelah program dijalankan, *user* akan dibawa ke subprogram *cover().* *Flowchart* subprogram *cover()* dapat dilihat pada gambar di bawah ini.

|  |
| --- |
|  |

**Gambar 4.2** *Flowchart* Subproses *Cover*

Gambar 4.2 adalah *flowchart* dari subprogram *cover()* pada program zodiak. Setelah subprogram dijalankan, program akan menampilkan identitas dari masing-masing anggota kelompok. Setelah itu, *user* dibawa ke subprogram *menu1()*. *Flowchart* dari subprogram *menu1()* dapat dilihat pada gambar di bawah.

|  |
| --- |
| C:\Users\User\AppData\Local\Microsoft\Windows\INetCache\Content.Word\Flowchart Zodiac-Halaman-3.drawio.png |

**Gambar 4.3** *Flowchart* Subproses Menu 1

Gambar 4.3 adalah *flowchart* dari subprogram *menu1()* pada program zodiak. Setelah subprogram dijalankan, akan muncul pilihan-pilihan menu zodiak yang bisa dipilih berdasarkan tanggal yang diinput *user*. Setelah memilih menu, *user* akan dibawa ke subprogram *tanggal()*. *Flowchart* dari subprogram *tanggal()* dapat dilihat pada kode program di bawah.

|  |
| --- |
|  |

**Gambar 4.4** *Flowchart* Subproses Tanggal

Gambar 4.4 adalah *flowchart* dari subprogram *tanggal()* pada program zodiak. Setelah subprogram dijalankan, *user* dapat menginputkan tanggal yang ingin ditentukan zodiaknya. Setelah meng*input* tanggal, *user* akan dibawa ke subprogram *bulan().* *Flowchart* dari subprogram *validasi()* dapat dilihat pada gambar dibawah.

|  |
| --- |
|  |

**Gambar 4.5** *Flowchart* Subproses Validasi *Input*

Gambar 4.5 adalah *flowchart* dari subprogram *validasi().* Pada subprogram ini, *input* tanggal dari *user* pada subprogram sebelumnya akan diproses. Apabila ada kesalahan dalam meng*input* tanggal, maka *user* akan diminta untuk mengulangi meng*inputkan* tanggalnya. *Flowchart* dari subprogram *bulan(int b)* dapat dilihat pada gambar di bawah.

|  |
| --- |
|  |

**Gambar 4.6** *Flowchart* Subproses Bulan

Gambar 4.6 adalah *flowchart* dari subprogram bulan. Pada subprogram ini, *user* dapat menginputkan bulan yang akan diproses dalam program zodiak. Setelah itu, user akan dibawa ke subprogram *validasi\_bulan (intb).* *Flowchart* subprogram *validasi\_bulan (intb)* dapat dilihat pada gambar di bawah.

|  |
| --- |
|  |

**Gambar 4.7** *Flowchart* Subproses Validasi Bulan

Gambar 4.7 adalah *flowchart* dari subprogram *validasi\_bulan().* Pada subprogram ini, *input* bulan dari user pada subprogram sebelumnya akan diproses disini. Apabila ada kesalahan dalam meng*inputkan* bulan, maka *user* akan diminta untuk mengulangi *input*. Apabila bulan yang di*inputkan* tidak ada masalah, maka *user* akan dibawa ke subprogram *zodiak().* *Flowchart* dari subprogram *zodiak()* dapat dilihat pada gambar di bawah.

|  |
| --- |
|  |

**Gambar 4.8** *Flowchart* Subproses Zodiak

Gambar 4.8 adalah *flowchart* dari subprogram *zodiak(intb, int hasil, int cadangan).* Pada subprogram ini, tanggal dan bulan yang di*inputkan* pada subprogram akan diproses. Program nantinya akan mengeluarkan *output* zodiak berdasarkan tanggal dan bulan yang di*inputkan user*. Setelah itu, *user* akan dibawa ke subprogram *ulang().* *Flowchart s*ubprogram *ulang()* dapat dilihat pada gambar di bawah.

|  |
| --- |
|  |

**Gambar 4.9** *Flowchart* Subproses Ulang

Gambar 4.9 adalah *flowchart* subprogram *ulang().* Pada subprogram ini *user* dapat memilih untuk mengulang program apabila ingin mencoba program kembali. Apabila tidak ingin mengulang, maka *user* akan keluar dari program zodiak dan program akan dihentikan.

#### Pseudocode Program Zodiak

*Pseudocode* merupakan sebuah cara penulisan dari algoritma yang menyerupai bahasa pemrograman, namun *pseudocode* ditulis lebih sederhana dan ringkas sehingga dapat dengan mudah dipahami oleh manusia. Selain itu, *pseudocode* juga dapat mempermudah seorang *programmer* dalam menyusun sebuah program secara terstruktur dan sekuensial. Berikut ini merupakan *pseudocode* dari program zodiak yang terdapat di Modul II.

|  |
| --- |
| “Program Menentukan Zodiak  DESKRIPSI  (Program menentukan zodiak berdasarkan tanggal dan bulan lahir )  FUNCTION  tanggal()  DEKLARASI  a, hasil : float  b : int  BEGIN  read(input)  while(a!=b || a<0 || a>31) do  a <- validasi()  b <- (int)a  if(a!=b || a<0 || a>31) then  write("Ulangi Input Anda")  end if  end while  call system("cls")  call bulan(b)  STOP  PROCEDURE  bulan(int b)  DEKLARASI  a : char  BEGIN  read(input)  call validasi\_bulan(b)  STOP  FUNCTION  validasi()  DEKLARASI  input[100] : char  a, salah, depan : int  hasil : float    BEGIN  a <- 0  salah <- 0  depan <- 0  read(input)  if(input[a]=='\0') then  salah <- 1  end if  while(input[a]!='\0') do  if(input[a]>='0' && input[a]<='9')  depan=(depan\*10)+(input[a]-48)  a++  else  salah <- 1  input[a] <- '\0'  end if  hasil <- depan  end while  if(salah==1) the  n  write("Ulangi Input Anda")  return call validasi()  else  return hasil  end if  STOP  PROCEDURE  validasi\_bulan(int b)  DEKLARASI  a, salah, depan, hasil, i : int  input[100], cadangan[100] : char  BEGIN  a <- 0  salah <- 0  depan <- 0  hasil <- 0  read(input)  if(input[a]=='\0') then  salah <- 1  end if  for (i=0;i<1;)  if(input[a]!='\0') then  cadangan[a] <- input[a]  a++  else  cadangan[a] <- '\0'  i <- 1  end if  end for  a <- 0  while(input[a]!='\0') do  if(input[0]>='A' && input[0]<='Z') then  a++  else if(input[a]>='0' && input[a]<='9') then  if(input[a+1]>='a' && input[a+1]<='z') then  salah <- 1  input[a] <- '\0'  else  depan <- (depan\*10)+(input[a]-48)  a++  end if  if(depan<1 || depan>12) then  salah <- 1  input[a] <- '\0'  end if  else if(input[a]>='a' && input[a]<='z') then  if(input[0]>='a' && input[0]<='z') then  salah <- 1  input[a] <- '\0'  else if(input[a+1]>='0' && input[a]<='9') then  salah <- 1  input[a] <- '\0'  else  a++  end if  else  salah <- 1  input[a] <- '\0'  end if  hasil <- depan  end while  if(salah==1) then  write("Ulangi Input Anda")  call return validasi\_bulan(b)  else  call system("cls")  if(hasil!=0) then  b, hasil  write("Tanggal/Bulan Lahir")  else  b, cadangan  write("Tanggal/Bulan Lahir")  end if  call zodiak(b, hasil, cadangan)  end if  STOP  PROCEDURE  zodiak(int b, int hasil, char cadangan[])  DEKLARASI  bulan : int  BEGIN  if(hasil!=0) then  bulan <- hasil  else  if((cadangan, "Maret")==0) then  bulan <- 3  else if((cadangan, "April")==0) then  bulan <- 4  else if((cadangan, "Mei")==0) then  bulan <- 5  else if((cadangan, "Juni")==0) then  bulan <- 6  else if((cadangan, "Juli")==0) then  bulan <- 7  else if((cadangan, "Agustus")==0) then  bulan <- 8  else if((cadangan, "September")==0) then  bulan <- 9  else if((cadangan, "Oktober")==0) then  bulan <- 10;  else if((cadangan, "November")==0) then  bulan <- 11  else if((cadangan, "Desember")==0) then  bulan <- 12  else if((cadangan, "Januari")==0) then  bulan <- 1  else if((cadangan, "Februari")==0) then  bulan <- 2  else  bulan <- 0  end if  if(b>=21 && b<=31 && bulan==3 || b>=1 && b<=19 && bulan==4) then  write(" ZODIAK ANDA ADALAH ARIES")  else if(b>=20 && b<=30 && bulan==4 || b>=1 && b<=20 &&  bulan==5) then  write(" ZODIAK ANDA ADALAH TAURUS")  else if(b>=21 && b<=31 && bulan==5 || b>=1 && b<=20 &&  bulan==6) then  write(" ZODIAK ANDA ADALAH GEMINI")  else if(b>=21 && b<=30 && bulan==6 || b>=1 && b<=22 &&  bulan==7) then  write(" ZODIAK ANDA ADALAH CANCER")  else if(b>=23 && b<=31 && bulan==7 || b>=1 && b<=22 &&  bulan==8) then  write(" ZODIAK ANDA ADALAH LEO")  else if(b>=23 && b<=31 && bulan==8 || b>=1 && b<=22 &&  bulan==9) then  write(" ZODIAK ANDA ADALAH VIRGO")  else if(b>=23 && b<=30 && bulan==9 || b>=1 && b<=22 &&  bulan==10) then  write(" ZODIAK ANDA ADALAH LIBRA")  else if(b>=23 && b<=31 && bulan==10 || b>=1 && b<=21 &&  bulan==11) then  write(" ZODIAK ANDA ADALAH SCORPIO")  else if(b>=22 && b<=30 && bulan==11 || b>=1 && b<=21 &&  bulan==12) then  write(" ZODIAK ANDA ADALAH SAGITARIUS")  else if(b>=22 && b<=31 && bulan==12 || b>=1 && b<=19 &&  bulan==1) then  write(" ZODIAK ANDA ADALAH CAPRICORN")  else if(b>=20 && b<=31 && bulan==1 || b>=1 && b<=18 &&  bulan==2) then  write(" ZODIAK ANDA ADALAH AQUARIUS")  else if(b>=19 && b<=29 && bulan==2 || b>=1 && b<=20 &&  bulan==3) then  write(" ZODIAK ANDA ADALAH PISCES")  else  write("Ulangi Input Anda")  call system("pause")  call system("cls")  call menu1()  enf id  call ulang()  STOP  PROCEDURE  ulang()  DEKLARASI  a : float  b : int  BEGIN  write("Ingin Mencoba Programnya Lagi")  write("1.Ya")  write  ("2.Tidak")  while(a!=b || a<1 || a>2) do  a <- validasi()  b <- (int)a  if(a!=b || a<1 || a>2) then  write("Ulangi Input Anda")  end if  end while  call system("cls")  if(a==1) then  call menu1()  else if(a==2){  write("Terima Kasih")  end if  call system("exit")  STOP  PROCEDURE  cover()  BEGIN  write("Nama kelompok")  call system("pause")  call system("cls")  call menu1()  STOP  FUNCTION  main()  BEGIN  call cover()  return 0  STOP  “ |

**Kode Program 4.1** *Pseudocode* Program Zodiak

Kode Program 4.1 merupakan bentuk *pseudocode* atau kode sederhana yang dibuat dengan menggunakan bahasa manusia, sehingga dapat lebih mudah untuk dipahami. *Pseudocode* di atas juga telah di buat sesuai dengan program zodiak.

#### Trace Flowchart Program Zodiak

*Trace* program merupakan sebuah proses untuk melacak *input* yang dimasukkan oleh *user* ke dalam sebuah program serta menganalisa hasil *input* tersebut serta memahami proses pengolahan yang ada pada sebuah program untuk mendapatkan *output* yang sesuai. Berikut ini adalah *trace* dari program zodiak yang dijabarkan melalui tabel di bawah ini.

**Tabel 4.1** *Trace* Program Zodiak

|  |  |
| --- | --- |
| *Input* | b = 11 // tanggal  cadangan = Maret //bulan |
| Proses | if(strcmp(cadangan, "Maret")==0){  bulan=3;  }  dan  (b>=19 && b<=29 && bulan==2 || b>=1 && b<=20 && bulan==3) |
| *Output* | Tanggal/Bulan Lahir Anda : 11/Maret  Zodiak Anda Adalah Pisces |

Tabel 4.1 merupakan hasil *trace* dari Program Menentukan Zodiak. Berdasarkan hasil pengujian tersebut, *input* tanggal yang bernilai 11 dan bulan yang bernilai 3 sesuai dengan tanggal <= 29 && bulan == 2 OR tanggal <= 20 && bulan == 3 sehingga menghasilkan *output* “Zodiak Anda Adalah Pisces”.

### Flowchart, Pseudocode, dan Trace Flowchart Program Nilai Akhir

Program nilai akhir adalah suatu program yang mampu menentukan Nilai Akhir (NA) mata kuliah berupa huruf berdasarkan ketentuan yang sudah diberikan. Program ini membutuhkan *input* berupa banyaknya kehadiran, nilai tugas sebanyak tiga kali, nilai *quiz*, nilai UTS, dan nilai UAS. *Output* yang akan diberikan oleh program adalah total nilai angka serta nilai huruf yang didapat dengan ketentuan yang diberikan. Berikut adalah *flowchart, pseudocode,* serta *trace* dari program nilai akhir.

#### Flowchart Program Nilai Akhir

*Flowchart* merupakan salah satu metode dalam menyampaikan suatu alur algoritma yang menggunakan simbol-simbol dengan arti tertentu. *Flowchart* dapat membantu dalam memahami alur dari algoritma yang akan digunakan. Berikut ini adalah *flowchart* dari program nilai akhir.

1. **Flowchart Program Nilai Akhir Int Main()**

Program akan mengeksekusi perintah-perintah yang berada pada fungsi *int main()* sebagai fungsi utama. Fungsi ini adalah fungsi pertama yang akan dipanggil. Berikut adalah *flowchart* dari bagian *int main()*.

|  |
| --- |
|  |

**Gambar 4.10** *Flowchart* Fungsi Int Main() Program Nilai Akhir

Gambar 4.10 merupakan *flowchart* dari bagian fungsi *int main()*. Pada fungsi ini, prosedur *nama\_kelompok()* akan dipanggil terlebih dahulu. Setelahnya, pengguna akan diberikan pilihan menu yaitu nilai akhir atau keluar dari program. Program pun akan membaca *input* dari pengguna dan mengarahkan pengguna ke sub-proses atau prosedur yang sesuai dengan *input* tersebut. Sub-proses yang dipanggil setelahnya akan dijelaskan pada *flowchart* selanjutnya.

1. **Flowchart Program Nilai Akhir Prosedur Nama\_Kelompok()**

Alur algoritma pada prosedur *nama\_kelompok()* dapat dilihat pada *flowchart* di bawah ini. Prosedur ini akan dipanggil di bagian awal pada saat program dimulai. Berikut adalah *flowchart* dari *nama\_kelompok().*

|  |
| --- |
|  |

**Gambar 4.11** *Flowchart* Prosedur Nama Kelompok

Gambar 4.11 merupakan *flowchart* dari prosedur *nama\_kelompok()*. Prosedur ini cukup sederhana, yang mana hanya memberikan *output* berupa nama kelompok dan lalu kembali ke fungsi sebelumnya.

1. **Flowchart Program Nilai Akhir Prosedur Nilai\_Akhir()**

Bila pengguna memberikan *input* 1 pada bagian sebelumnya, maka program akan mengarahkan pengguna pada prosedur *nilai\_akhir()*. Berikut adalah *flowchart* dari prosedur *nilai\_akhir()*.

|  |
| --- |
|  |

**Gambar 4.12** *Flowchart* Prosedur Nilai Akhir

Gambar 4.12 telah memberikan ilustrasi prosedur *nilai\_akhir()*. Prosedur ini bekerja dengan cara mendapatkan *input* yang diperlukan untuk menghitung nilai, seperti absensi, nilai tugas 1, nilai tugas 2, nilai tugas 3, nilai *quiz,* nilai UTS, serta nilai UAS. Setiap *input* tersebut akan digunakan untuk menghitung nilai akhir sesuai dengan ketentuan yang telah diberikan. Fungsi *totalNilaiTugas()* dan *totalNilaiAkhir()* selanjutnya akan dipanggil untuk menghitung nilai. Setelah selesai, program akan memberikan *output* berupa nilai akhir dan menanyakan apakah pengguna ingin mengulang proses tersebut atau tidak. Bila iya, maka pengguna akan diarahkan pada menu awal. Bila tidak, maka program akan berakhir.

1. **Flowchart Program Nilai Akhir Fungsi TotalNilaiTugas()**

Fungsi ini akan dipanggil pada prosedur *nilai\_akhir()* untuk melakukan perhitungan dari nilai tugas sesuai dengan ketentuan yang diberikan. Berikut adalah *flowchart* dari fungsi *totalNilaiTugas().*

|  |
| --- |
|  |

**Gambar 4.13** *Flowchart* Fungsi Total Nilai Tugas

Gambar 4.13 merupakan *flowchart* dari fungsi *totalNilaiTugas()*. Fungsi ini akan mengembalikan nilai bertipe *float* sebagai total nilai tugas yang nantinya akan digunakan kembali pada fungsi *totalNilaiAkhir()* untuk menghitung perolehan nilai akhir.

1. **Flowchart Program Nilai Akhir Fungsi TotalNilaiAkhir()**

Fungsi *totalNilaiAkhir()* akan dipanggil setelah fungsi *totalNilaiTugas()* untuk menyelesaikan perhitungan nilai akhir. Perhitungan dilakukan sesuai dengan ketentuan yang telah diberikan. Berikut adalah *flowchart* dari fungsi *totalNilaiAkhir()*.

|  |
| --- |
|  |

**Gambar 4.14** *Flowchart* Fungsi Total Nilai Akhir

Gambar 4.14 merupakan *flowchart* dari fungsi *totalNilaiAkhir()*. Fungsi ini memiliki parameter yang akan menerima argumen berupa nilai *quiz,* UTS, UAS, dan nilai tugas yang nantinya akan digunakan dalam perhitungan. Setelah selesai, fungsi *totalNilaiAkhir()* akan mengembalikan nilai bertipe *float* sebagai hasil dari perhitungan nilai akhir.

#### Pseudocode Program Nilai Akhir

Setelah selesai menyusun *flowchart*, ada baiknya untuk melanjutkan ke penyusunan *pseudocode* untuk mempermudah dalam menyusun kode program yang asli. Berikut adalah *pseudocode* dari program nilai akhir.

|  |
| --- |
| Algoritma Menghitung Nilai Akhir  DESKRIPSI:  nilaiAkhir(): void //prototype function nilaiAkhir  FUNCTION totalNilaiTugas(tugas1: float, tugas2: float, tugas3: float): float  DEKLARASI:  nilaiTugas: float  DESKRIPSI:  BEGIN  nilaiTugas = ((tugas1+tugas2+tugas3)/3)\*20/100  RETURN nilaiTugas  END  FUNCTION totalNilaiAkhir(absensi: int, nilaiTugas: float, quiz: float, uts: float, uas: float): float  DEKLARASI:  nilaiAkhir: float  DESKRIPSI:  BEGIN  quiz = quiz\*15/100  uts = uts\*30/100  uas = uas\*30/100  nilaiAkhir = (absensi\*5/15) + nilaiTugas + quiz + uts + uas  RETURN nilaiAkhir  END  FUNCTION namaKelompok(): void  DESKRIPSI:  BEGIN  write “nama kelompok”  system(pause)  system(cls)  END  FUNCTION validasiMenu(): void  DEKLARASI:  input: int  DESKRIPSI:  BEGIN  IF input == 1 && input == 2 THEN  TRUE  ELSE  ERROR  write “masukkan angka dengan benar”  END  FUNCTION validasiAbsensi(): void  DEKLARASI:  input: int  DESKRIPSI:  BEGIN  IF input >= 0 && input <= 15 THEN  TRUE  ELSE  ERROR  write “masukkan angka dengan benar”  END  FUNCTION validasiNilai(): void  DEKLARASI:  input: float  DESKRIPSI:  BEGIN  IF input >= 0 && input <= 100 THEN  TRUE  ELSE  ERROR  write “masukkan angka dengan benar”  END  FUNCTION main(): int  DEKLARASI:  input: int  DESKRIPSI:  BEGIN  namaKelompok()    write “menu”  write “pilihan menu”  write “1. Hitung nilai akhir & 2. tutup program  read(input)  validasiMenu()  system(cls)    IF input == 1 THEN  nilaiAKhir()  ELSE IF input == 2 THEN  exit(0)  RETURN 0;  END  FUNCTION nilaiAkhir(): void  DEKLARASI:  absensi: int  tugas1: float  tugas2: float  tugas3: float  quiz: float  uts: float  uas: float  nilaiTugas: float  nilaiAkhir: float  read: int  DESKRIPSI:  BEGIN  DO  write “masukkan jumlah kehadiran: “  read(absensi)  validasiAbsensi()    IF absensi == 0 THEN  tugas1 = 0;  tugas2 = 0;  tugas3 = 0;  quiz = 0;  uts = 0;  uas = 0;  ELSE IF absensi >=1 && absensi < 12 THEN  uas = 0  write “masukkan nilai tugas 1: “  read(tugas1)  validasiFloat()  write “masukkan nilai tugas 2: “  read(tugas2)  validasiFloat()  write “masukkan nilai tugas 3: “  read(tugas3)  validasiFloat()  write “masukkan nilai quiz: “  read(quiz)  validasiFloat()  write “masukkan nilai uts: “  read(uts)  validasiFloat()  ELSE THEN  write “masukkan nilai tugas 1: “  read(tugas1)  validasiFloat()  write “masukkan nilai tugas 2: “  read(tugas2)  validasiFloat()  write “masukkan nilai tugas 3: “  read(tugas3)  validasiFloat()  write “masukkan nilai quiz: “  read(quiz)  validasiFloat()  write “masukkan nilai uts: “  read(uts)  validasiFloat()  write “masukkan nilai uas: “  read(uas)  validasiFloat()    system(cls)  nilaiTugas = totalNilaiTugas(tugas1, tugas2, tugas3);  nilaiAkhir = totalNilaiAkhir(absensi, nilaiTugas, quiz, uts, uas);    write “jumlah kehadiran: %d/15”, absensi  write “nilai tugas: %d.2f”, nilaiTugas  write “nilai quiz: %d.2f”, quiz  write “nilai uts: %d.2f”, uts  write “nilai uas: %d.2f”, uas  write “nilai akhir: %d.2f”, nilaiAKhir  IF nilaiAkhir >= 0 && nilaiAkhir < 45 THEN  write “(E)”  ELSE IF nilaiAkhir >= 45 && nilaiAkhir < 50 THEN  write “(D)”  ELSE IF nilaiAkhir >= 50 && nilaiAkhir < 55 THEN  write “(D+)”  ELSE IF nilaiAkhir >= 55 && nilaiAkhir < 60 THEN  write “(C)”  ELSE IF nilaiAkhir >= 60 && nilaiAkhir < 65 THEN  write “(C+)”  ELSE IF nilaiAkhir >= 65 && nilaiAkhir < 75 THEN  write “(B)”  ELSE IF nilaiAkhir >= 75 && nilaiAkhir < 80 THEN  write “(B+)”  ELSE IF nilaiAkhir >= 80 && nilaiAkhir <= 100 THEN  write “(A)”  write “apakah ingin mengulang?”  write “1. iya & 2. tidak”  read(read)  validasiMenu()  WHILE read == 1  END |

**Kode Program 4.2** *Pseudocode* Program Nilai Akhir

Kode Program 4.2 merupakan *pseudocode* dari program nilai akhir. *Pseudocode* ini akan membantu dalam mempelajari alur algoritma yang akan digunakan pada kode program yang asli dan dapat dimodifikasi dengan menyesuaikan *syntax* sesuai dengan bahasa pemrograman yang akan digunakan.

#### Trace Flowchart Program Nilai Akhir

Langkah terakhir adalah melakukan *trace* atau pengecekan pada alur algoritma yang telah disusun sebelumnya demi memeriksa apakah algoritma yang diterapkan sudah sesuai dengan kebutuhan atau belum. Berikut adalah *trace* dari program nilai akhir.

**Tabel 4.2** *Trace* Program Nilai Akhir

|  |  |
| --- | --- |
| *Input* | input = 1  absensi = 15  tugas1 = 89  tugas2 = 78  tugas3 = 90  quiz = 100  uts = 86  uas = 89  read = 2 |
| Proses | CALL namaKelompok()  nilaiTugas = CALL totalNilaiTugas(89, 78, 90)  nilaiAkhir = CALL totalNilaiAkhir(15, nilaiTugas, quiz, 86, 89) |
| *Output* | Nama Kelompok  Instruksi menu  Instruksi masukkan jumlah kehadiran, nilai 1-3, nilai UTS, *quiz*, UAS  Nilai akhir = 89.63 (A)  Ingin mengulang?  Terima kasih |

Tabel 4.2 merupakan contoh dari *trace* pada program nilai akhir. Pengguna akan memilih menu 1 dan lalu memasukkan total absensi beserta nilai-nilai yang diperlukan. Selanjutnya, program akan mengeksekusi fungsi, prosedur, serta *statement-statement* lainnya yang bersangkutan dan akhirnya memberikan *output* berupa nilai akhir dari pengguna tersebut. *Trace* lebih lengkap dapat disimak pada tabel yang telah disediakan.

### Flowchart, Pseudocode, dan Trace Flowchart Program Konversi

Program konversi merupakan suatu program yang dapat melakukan suatu konversi dari bilangan desimal ke bilangan biner dan dari bilangan biner ke dalam bilangan desimal. Berikut adalah *flowchart*, *pseudocode*, serta *trace* dari program konversi.

#### Flowchart Program Konversi

*Flowchart* program konversi biner memiliki fungsi untuk memvisualisasikan alur dari program konversi biner. Hal ini berguna untuk memudahkan *programmer* untuk membuat program karena algoritma dari program sudah divisualisasikan sehingga akan mudah dipahami oleh *programmer*. Alur *flowchart* konversi biner bisa dilihat pada gambar.

|  |
| --- |
|  |

**Gambar 4.15** *Flowchart Int Main* Program Konversi

Gambar 4.15 di atas merupakan *flowchart* dari fungsi main pada program konversi biner. Alur dari *flowchart* fungsi main program konversi biner dimulai dari *“mulai”.* Lalu akan menampilkan nama kelompok dan lanjut ke fungsi main menu.

|  |
| --- |
|  |

**Gambar 4.16** *Flowchart* Subproses *Main Menu*

Gambar 4.16 di atas merupakan *flowchart* dari fungsi *mainMenu()* program konversi biner. pertama akan masuk ke pilihan konversi. Setelah pengguna memilih konversi, maka akan masuk ke fungsi dari masing-masing konversi untuk diproses selanjutnya. Setelah selesai akan masuk ke pilihan apakah pengguna ingin mengulang program atau tidak.

|  |
| --- |
|  |

**Gambar 4.17** *Flowchart* Subproses *Decimal to Binary*

Gambar 4.17 di atas merupakan *flowchart* dari prosedur *decimalToBinary()* yang digunakan untuk menjalankan proses untuk melakukan konversi bilangan desimal ke bilangan biner. Fungsi ini akan mengambil *input* berupa nilai desimal yang akan diproses oleh program dan akan mengeluarkan hasil konversi bilangan tersebut ke dalam bilangan biner.

|  |
| --- |
|  |

**Gambar 4.18** *Flowchart* Subproses *Binary to Decimal*

Gambar 4.18 di atas merupakan *flowchart* dari fungsi *binaryToDec()* yang digunakan untuk menjalankan proses untuk mengkonversi bilangan biner ke bilangan desimal. Fungsi ini akan mengambil *input* berupa nilai biner yang akan diproses oleh program dan akan mengeluarkan hasil konversi bilangan yang diberikan ke dalam bilangan desimal.

|  |
| --- |
|  |

**Gambar 4.19** *Flowchart* Subproses Validasi *Input*

Gambar 4.19 di atas merupakan *flowchart* dari validasi yang digunakan pada program konversi biner. Validasi ini memiliki dua alur fungsi utama yang bisa dipakai sesuai dengan kondisi yang ada. Di mana fungsi *validasiRange()* digunakan untuk memvalidasi angka dengan *range* tertentu, dan fungsi *validasi()* digunakan untuk mengkonversi angka secara general.

#### Pseudocode Program Konversi

*Pseudocode* program konversi biner berisi perintah-perintah yang akan digunakan untuk membuat program konversi biner. *Pseudocode* tidak terpaku pada peraturan dari bahasa pemrograman mana pun, jadi pada dasarnya *pseudocode* merupakan orek-orek awal bagi *programmer* untuk membuat program yang diinginkan.

|  |
| --- |
| //Program Konversi Biner  FUNCTION validasi() : int  DEKLARASI:  input, notValid : char  valid : int  READ input, notValid  DESKRIPSI:  BEGIN  IF (sscanf(input, "%d%c", &valid, &notValid) == 1) THEN  return valid  ELSE  CALL system("cls")  WRITE " Maaf Input anda salah !"  WRITE " Masukan Input Dengan Benar "  CALL validasi()  END  FUNTION validasiRange() : int  DEKLARASI:  input, notValid : char  valid : int  READ input, notValid  DESKRIPSI:  BEGIN  IF (sscanf(input, "%d%c", &valid, &notValid) == 1 && valid == 1 || valid == 2) THEN  return valid  ELSE  CALL system("cls")  WRITE " Maaf Input anda salah !"  WRITE " Masukan Input Dengan Benar"  CALL validasiRange()  END  PROCEDURE decimalToBinary() : void  DEKLARASI:  a, n, hasil, desimal, binary: int  DESKRIPSI:  BEGIN  WRITE " Masukan Nilai Desimal : "  READ desimal  a = validasi()  FOR(int i = 0 a > 0 i++)  n = a%2  a/= 2  hasil[i] = n  ENDFOR  FOR(int i = 0 i < sizeof(hasil)/sizeof(hasil[0]) i++)  hasil[i] = (hasil[i] \* pow(10,i))  binary += hasil[i]  ENDFOR  CALL system("cls")  WRITE "Bilangan Desimal : $desimal"  WRITE "Bilangan Biner : $binary"  END  PROCEDURE binarytoDec() : void  DEKLARASI:  a, hasil : int  biner : char  WRITE "masukan nilai biner : "  READ biner  DESKRIPSI:  BEGIN  FOR(int i = 0 i < strlen(biner) i++)  IF(biner[i] != '1' && biner[i] != '0')  a++  WRITE("i = $i, a = $a, biner = $biner[i]  ELSE IF(biner[i] == '1' )  hasil += (pow(2,strlen(biner)-1-i))  ENDFOR  IF(a > 0)  CALL system("cls")  WRITE " Maaf Input anda salah !"  WRITE " Masukan Input Dengan Benar"  CALL binarytoDec()  ELSE  WRITE "\n$hasil\n"  CALL system("cls")  WRITE " Bilangan Biner : $biner"  WRITE " Bilangan Desimal : $hasil"  END  PROCEDURE namaKelompok() : void  CALL system("cls")  WRITE "namakelompok"  CALL system("pause")  CALL system("cls")  }  PROCEDURE mainMenu() : void  DEKLARASI  pilihan, read : int  DEKRIPSI:  BEGIN  DO  CALL system("cls")  WRITE " mainMenu"  WRITE " Konversi : "  pilihan = validasiRange()  SWITCH (pilihan)  case 1:  CALL decimalToBinary()  break  case 2:  CALL binarytoDec()  break  default:  WRITE " Maaf Input anda salah !"  WRITE " Masukan Input Dengan Benar"  CALL mainMenu()  ENDSWITCH  CALL system("pause")  CALL system("cls")  WRITE " Apakah anda ingin menghitung lagi?"  WRITE "(1 = ya / 2 = tidak)")  read = validasiRange()  WHILE(read == 1)  CALL system("cls")  WRITE "\t\t \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \n"  WRITE "\t\t \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \n"  WRITE "\t\t| | | |\n"  WRITE " \t\t| | | |\n"  WRITE "\t\t| | Terima Kasih Telah Menggunakan Program Ini yaaaaaa!!! | |\n"  WRITE "\t\t| | | |\n"  WRITE "\t\t| | | |\n"  WRITE "\t\t| | \\|||||||||||||||||||| |||||||||||||||||\\ | |\n"  WRITE "\t\t| | \\||||||||||||||||||| ||||||||||||||||||\\ | |\n"  WRITE "\t\t| | \\|||||||||||||||||| |||||||||||||||||||\\ | |\n"  WRITE "\t\t| | |||||||||| | |\n"  WRITE "\t\t| | |||||||||| |||||||||||||||||||/ | |\n"  WRITE "\t\t| | |||||||||| ||||||||||||||||||/ | |\n"  WRITE "\t\t| | |||||||||| |||||||||||||||||/ | |\n"  WRITE "\t\t| | |||||||||| ||||||||| | |\n"  WRITE "\t\t| | |||||||||| ||||||||| | |\n"  WRITE "\t\t| | |||||||||| ||||||||| | |\n"  WRITE "\t\t| | |||||||||| ||||||||| | |\n"  WRITE "\t\t| | \\|||||||| |||||||/ | |\n"  WRITE "\t\t| | \\||||||| ||||||/ | |\n"  WRITE "\t\t| | \\|||||| |||||/ | |\n"  WRITE "\t\t| | \\||||| ||||/ | |\n"  WRITE "\t\t| | \\|||| |||/ | |\n"  WRITE "\t\t| |\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_| |\n"  WRITE "\t\t\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \n"  END  FUNCTION main() : int  DESKRIPSI  BEGIN  CALL namaKelompok()  CALL mainMenu()  END |

**Kode Program 4.3** *Pseudocode* Program Konversi

Kode Program 4.3 di atas merupakan *pseudocode* dari program konversi biner. Program meminta pengguna untuk meng-*input* pilihan apakah ingin melakukan konversi dari bilangan desimal ke biner atau sebaliknya. Setelah itu akan diproses di fungsi berdasarkan pilihan pengguna. Dan hasil dari konversi akan ditampilkan sebagai *output.*

#### Trace Flowchart Program Konversi

*Trace* pada program merupakan salah satu hal yang penting. *Trace* berguna untuk mengetahui alur program bekerja dan dapat digunakan untuk mendeteksi kesalahan yang ada pada program sehingga kesalahan bisa ditemukan dan diperbaiki. *Trace* pada program konversi bisa dilihat pada tabel.

**Tabel 4.3** *Trace* Program Konversi

|  |  |
| --- | --- |
| *Input* | Masukan Pilihan Anda (1/2) = 1  Bilangan desimal = 101  Ingin Mencoba Menginput Kembali (1/2) = 2 |
| Proses | CALL decimalToBinary()  for(int i = 0; a > 0; i++){  n = a%2;  a/= 2;  hasil[i] = n;  }  for (int i = 0; i < sizeof(hasil)/sizeof(hasil[0]); i++) {  hasil[i] = (hasil[i] \* pow(10,i));  binary += hasil[i];  }  Return binary; |
| *Output* | Tampilkan decimalToBinary()  Biner = 1100101  Terima kasih |

Tabel 4.3 adalah hasil *trace* dari program konversi. *Trace* di atas menjelaskan bagaimana program ini berjalan, pengguna memilih konversi no 1 yaitu konversi dari desimal ke biner, lalu pengguna akan menginput bilangan desimal yang ingin dikonversi menjadi bilangan biner yang dalam *trace* ini bilangan 101, lalu program akan menjalankan fungsi untuk mengubah bilangan desimal ke biner, setelah fungsi dijalankan program akan menampilkan hasil dari konversi yang bernilai 1100101. Lalu program akan menampilkan menu ulang untuk menanyakan pengguna apakah ingin mengulang program (1 = ya, 2 = tidak), jika pengguna memilih 1 maka program akan kembali berulang, namun jika pengguna memilih 2 maka menu terima kasih akan muncul dan program akan berakhir.

### Flowchart, Pseudocode, dan Trace Flowchart Program Bilangan Fibonacci

Bilangan *fibonacci* merupakan suatu deret bilangan yang terbentuk dari susunan angkanya merupakan penjumlahan dari dua angka sebelumnya. Program bilangan *fibonacci* ini dapat menampilkan *n* deret bilangan *fibonacci* dengan menggunakan metode rekursif dan iteratif. Program ini akan menerima *input* berupa sembarang bilangan *n* dan memberikan *output* berupa *n* deret bilangan *Fibonacci* dengan kedua metode tersebut. Berikut adalah *flowchart, pseudocode,* serta *trace* dari program *fibonacci*.

#### Flowchart Program Bilangan Fibonacci

*Flowchart* merupakan suatu diagram yang kerap digunakan untuk ilustrasi alur atau notasi algoritma dengan menggunakan simbol-simbol tertentu. Penggunaan *flowchart* sering ditemukan untuk mempermudah dalam pemahaman notasi algoritma. Berikut adalah *flowchart* dari program bilangan *fibonacci*.

|  |
| --- |
|  |

**Gambar 4.20** *Flowchart* Program Bilangan *Fibonnaci*

Gambar 4.20 merupakan *flowchart* dari program bilangan *fibonacci*. Pengguna akan diminta untuk memilih metode yang diinginkan. Setelah menerima *input* dari pengguna, maka program pun akan menjalankan prosedur dan fungsi yang sesuai dengan permintaan pengguna, baik secara rekursif maupun iteratif. Pengguna lantas memberikan *input* yang akan menjadi batas elemen deret bilangan *fibonacci.* Program pun akan memberikan deret bilangan *fibonacci* yang diberikan setelah menjalankan metode yang dipilih. Setelah selesai, pengguna dapat mengulang kembali dengan kembali ke menu utama atau keluar dari program.

#### Pseudocode Program Bilangan Fibonacci

*Pseudocode* merupakan salah satu bentuk notasi algoritma yang digunakan untuk menggambarkan suatu proses atau alur algoritma ke dalam ‘kode program informal’ yang mudah dimengerti manusia. *Pseudocode* sendiri aslinya bukanlah kode program asli, tapi dapat dimodifikasi atau dijadikan acuan dalam menyusun kode program dengan bahasa pemrograman tertentu. Berikut adalah *pseudocode* dari program bilangan *fibonacci.*

|  |
| --- |
| DESKRIPSI :  Program Fibonacci  FUNCTION fibonacciRekursif(n : int) : int  DEKLARASI :  BEGIN  if (n==0)THEN  return 0  else if (n==1) THEN  return 1  else THEN  return (fibonacciRekursif (n-1)+ fibonacciRekursif (n-2))  ENDIF  END  FUNCTION fiboRekursif() : void  DEKLARASI :  BEGIN  n : int  i : int  ulang = 1 : int  WRITE("\t\t============================================\n")  input\_int(&n, "\t\tMasukan batas bilangan fibonacci: ")  WRITE ("\t\t============================================\n")  if (n<0)THEN  fflush(stdin)  WRITE ("\t\t!Tidak ada fibonacci dibawah nol! \n")  else THEN  WRITE("\t\tUrutan bilangan fibonacci:")  WRITE("\n\t\t=> ")  for (i=0 i<n i++){  WRITE(" %d", fibonacciRekursif(i))  END  ENDIF WRITE("\n\t\t============================================\n")  END  FUNCTION fibonacciIteratif(n : int) : int  DEKLARASI :  BEGIN  i : int  fi : int  f1 = 0 : int  f2 = 1 : int  if (n == 0)THEN  return 0  else if (n == 1)THEN  return 1  else THEN  for (i=1 i<n i++)  fi = f1  f1 = f2  f2 = fi + f1  ENDFOR  return f2  ENDIF  END  FUNCTION fiboIteratif() : void  DEKLARASI :  BEGIN  n : int  i : int  ulang = 1 : int  WRITE("\t\t============================================\n")  input\_int(&n, "\t\tMasukan batas bilangan fibonacci: ")  WRITE("\t\t============================================\n")  if (n<0)THEN  fflush(stdin)  WRITE("\t\t!Tidak ada fibonacci dibawah nol! \n")  else THEN  WRITE("\t\tUrutan bilangan fibonacci""\n\t\t=>")  for (i=0 i<n i++){  WRITE(" %d", fibonacciIteratif(i))  END  ENDIF WRITE("\n\t\t============================================\n")  END  FUNCTION valid\_int(\*var : int) : int  DEKLARASI :  BEGIN  VAR buff[20] : char  VAR cek : char  fflush(stdin)  if(fgets(buff, sizeof(buff), stdin)!=NULL)THEN  if(sscanf(buff, "%d %c", var, &cek)==1)THEN  return 1  END  ENDIF  return 0  END  FUNCTION input\_int(\*var : int, \*prompt : char) : void  DEKLARASI :  BEGIN  while (1) THEN  WRITE(prompt)  if(valid\_int(var))  break  WRITE("\t\t--------------------------------------------")  WRITE("\n\t\t!Masukan angka saja, jangan yang lain!\n")  WRITE("\t\t--------------------------------------------\n")  WRITE("\n")  END  END  FUNCTION input\_pilihan(\*var : int, \*prompt : char) : void  DEKLARASI:  BEGIN  while (1) THEN  WRITE(prompt)  if(valid\_int(var))  break  WRITE("\n\t\tPilih salah satu metode!\n")  WRITE("\n")  END  END  FUNCTION input\_ulang(\*var : int, \*prompt : char) : void  DEKLARASI :  BEGIN  while (1) THEN  WRITE(prompt)  if(valid\_int(var))  break  WRITE("\t\tMau mengulang atau tidak?!")  END  END  FUNCTION main() : int  DEKLARASI :  BEGIN  pilihan : int  WRITE("\n")  WRITE("\t\t============================================\n")  WRITE("\t\t|| Program Fibonacci ||\n")  WRITE("\t\t|| Kelompok 23 ||\n")  WRITE("\t\t============================================\n")  WRITE("\t\t|| Pilih Salah Satu Opsi ||\n")  WRITE("\t\t============================================\n")  WRITE("\t\t|| 1. Fibonacci Rekursif ||\n")  WRITE("\t\t|| 2. Fibonacci Iteratif ||\n")  WRITE("\t\t============================================\n")  input\_pilihan(&pilihan, "\t\tMasukan pilihanmu => ")  if (pilihan <1)THEN  WRITE("\t\tMasukan angka yang tersedia!")    else if (pilihan == 1) THEN  fiboRekursif (fibonacciRekursif)  }else if (pilihan == 2) THEN  fiboIteratif (fibonacciIteratif)  }else if (pilihan >2) THEN  WRITE("\n\t\tPilihan tidak tersedia\n")  ENDIF  ulang()  return 0  END  FUNCTION ulang() : void  DEKLARASI :  BEGIN  mengulang : int  input\_ulang(&mengulang, "\n\t\tIngin mengulang program?\n\t\t1. YA \n\t\t2. Tidak\n""\t\t=> ")  if (mengulang == 1 ) THEN  system("cls")  main()  else if(mengulang == 2)THEN  WRITE("\t\t\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Terima Kasih \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n")  ENDIF  END |

**Kode Program 4.4** *Pseudocode* Program Bilangan *Fibonacci*

Kode Program 4.4 merupakan *pseudocode* dari program bilangan *fibonacci.* Selanjutnya, *pseudocode* dapat dijadikan sebagai acuan dalam pembuatan kode program yang asli, khususnya dengan bahasa pemrograman C.

#### Trace Flowchart Program Bilangan Fibonacci

*Trace* merupakan tahapan terakhir dalam penyusunan alur algoritma, yang mana bertujuan untuk memeriksa apakah algoritma yang digunakan sudah sesuai atau belum. Berikut adalah *trace* dari algoritma yang digunakan pada program bilangan *fibonacci*.

**Tabel 4.4** *Trace* Program *Fibonacci*

|  |  |
| --- | --- |
| *Input* | pilihan = 1  n = 5  pilihan = 2  n = 5  mengulang = 2 |
| Proses | CALL fiboRekursif()  (fibonacciRekursif (n-1) + fibonacciRekursif (n-2))  CALL fiboIteratif()  fi = f1; f1 = f2; f2 = fi + f1;  CALL ulang() |
| *Output* | Intruksi memilih pilihan :  Output urutan fibonacci rekursif:  => 0 1 1 2 3  Output urutan fibonacci iteratif:  => 0 1 1 2 3  Ingin mengulang mengulang?  Terima Kasih |

Tabel 4.4 di atas adalah *trace* dari program *fibonacci*. *User* memasukan input berupa pilihan 1 atau 2 dan memasukan batas bilangan *fibonacci* (n). *User* memasukan input pilihan 1 atau 2 dan batas bilangan *fibonacci*-nya yaitu 5, kemudian akan memanggil prosedur untuk melakukan operasi deret *fibonacci* secara rekursif atau iteratif, setelah operasi selesai akan menampilkan hasil dari urutan dan akan menanyakan apakah ingin mengulang program atau tidak. *Trace* di atas *user* memilih tidak, dimana program akan dihentikan.

### Flowchart, Pseudocode, dan Trace Flowchart Program Menara Hanoi

Menara Hanoi adalah sebuah teka-teki matematika yang mana pemain diharuskan untuk memindahkan seluruh cakram ke tiang yang berbeda. Teka-teki ini memiliki tiga tiang dan *n* cakram. Seluruh cakram ditumpuk pada satu tiang dengan cakram terkecil ditempatkan di bagian paling atas. Tujuan dari teka-teki ini adalah untuk memindahkan seluruh cakram ke tiang lainnya dengan mematuhi peraturan sebagai berikut:

1. Hanya dapat memindahkan satu cakram dalam satu waktu.
2. Cakram yang boleh dipindahkan hanya cakram teratas dari salah satu tumpukan dan ditempatkan pada tumpukan lain.
3. Tidak diperbolehkan untuk menumpuk cakram yang lebih besar di bawah cakram yang lebih kecil.

Program Menara Hanoi ini dapat memecahkan teka-teki Menara Hanoi secara rekursif dan iteratif. Program ini pun dapat menghitung banyaknya langkah minimum yang diperlukan untuk memindahkan *n* buah cakram.

*Input* yang diperlukan oleh program ini adalah *n* jumlah cakram serta tiang awal dan tiang tujuan, sementara *output* yang diberikan adalah banyaknya langkah minimal yang dibutuhkan serta langkah-langkah penyelesaiannya. Secara rekursif, program ini akan memanggil fungsi untuk memindahkan cakram sesuai peraturan secara terus menerus hingga kondisinya terpenuhi. Secara iteratif, program akan menjalankan *statement* tertentu secara terus menerus hingga kondisinya terpenuhi. Berikut adalah *flowchart, pseudocode,* dan *trace* dari program Menara Hanoi.

#### Flowchart Program Menara Hanoi

*Flowchart* merupakan suatu notasi algoritma yang kerap digunakan dalam proses mempelajari, menyusun, atau komunikasi mengenai suatu algoritma yang nantinya akan digunakan pada suatu program. Berikut adalah *flowchart* dari program Menara Hanoi.

|  |
| --- |
|  |

**Gambar 4.21** *Flowchart* Program Menara Hanoi

Gambar 4.21 merupakan *flowchart* dari program Menara Hanoi. *Flowchart* tersebut menjelaskan alur dari algoritma yang akan digunakan dalam membuat program Menara Hanoi dan ilustrasi mengenai bagaimana program tersebut akan berjalan.

Pada saat memasuki program, program akan memanggil prosedur *menu()* untuk menunjukkan instruksi berupa pilihan menu. Pengguna dapat memilih untuk melakukan pemecahan teka-teki Menara Hanoi secara rekursif, iteratif, atau pun menampilkan kredit berisi nama anggota kelompok dan keluar untuk menutup program. Setelah memilih, pengguna akan diminta untuk memberikan *input* berupa banyak cakram serta tiang awal dan tiang tujuan pada prosedur *hitungHanoiR()* untuk rekursif dan *hitungHanoiI()* untuk iteratif. Bila memilih rekursif, program akan menjalankan prosedur *hanoiRekursif (cakram, tiang1, tiang2)* dan memecahkan teka-teki dengan algoritma rekursif. Bila memilih iteratif, program akan menjalankan prosedur *hanoiIteratif (n, start, end)* dan memecahkan teka-teki dengan algoritma iteratif.

Setelah menjalankan prosedur tersebut, program pun akan memberikan sebuah *output* berupa langkah-langkah yang diambil dalam memecahkan teka-teki Menara Hanoi beserta langkah minimal untuk menyelesaikannya. Prosedur *repeat()* pun dipanggil untuk menanyakan apakah pengguna ingin mengulang program atau tidak. Bila ya, maka pengguna akan diarahkan kembali pada menu utama. Bila tidak, maka prosedur *keluar()* akan dipanggil, menampilkan *output* berupa ucapan terima kasih, dan program pun ditutup.

#### Pseudocode Program Menara Hanoi

Dalam menyusun suatu alur algoritma, *pseudocode* sering digunakan demi mempermudah implementasi algoritma tersebut ke dalam suatu kode program yang matang dan dapat dijalankan. Sifat *pseudocode* yang mirip dengan kode program tapi mudah dimengerti oleh manusia memudahkan dalam penggunaannya ke dalam seluruh bahasa pemrograman. Berikut adalah *pseudocode* dari program Menara Hanoi.

|  |
| --- |
| PROGRAM menaraHanoi  BEGIN  FUNCTION int validasiInt(int \*var)  fflush(stdin)  DECLARE  char buffer[1024]  char cek  IF (fgets(buffer, sizeof(buffer), stdin) != NULL) THEN  IF (sscanf(buffer, "%d %c", var, &cek) == 1)  return \*var  END IF  return 0  END IF  END FUNCTION  PROCEDURE validasiInput(int \*var)  WHILE (1)  fflush(stdin)  IF (CALL validasiInt(var)) THEN  BREAK  END IF  WRITE ("Masukan salah. Mohon masukkan input sesuai arahan.")  WRITE ("Masukkan input: ")  END PROCEDURE  PROCEDURE validasiHanoi(int \*var)  WHILE (1)  CALL validasiInput(var)  fflush(stdin)  IF (\*var > 0) THEN  BREAK  END IF  WRITE ("Masukan salah. Mohon masukkan input sesuai arahan.")  WRITE ("\t\t Masukkan input: ")  END PROCEDURE  PROCEDURE validasiTiangHanoi(int \*var)  WHILE (1)  CALL validasiHanoi(var)  fflush(stdin)  IF (1<=\*var && \*var<= 3) THEN  BREAK  END IF  WRITE ("Masukan salah. Mohon masukkan input sesuai arahan.\")  WRITE ("Masukkan input: ")  END PROCEDURE  PROCEDURE void validasiMenu(int \*var, int batas1, int batas2)  WHILE (1)  CALL validasiInput(var)  fflush(stdin)  IF (batas1 <= \*var && \*var <= batas2) THEN  BREAK  WRITE ("Masukan salah. Mohon masukkan input sesuai arahan.")  printf("Masukkan input: ")  END PROCEDURE  PROCEDURE hanoiRekursif(int n, int start, int dest)  IF (n==1) THEN  WRITE ("Cakram %d dipindahkan dari tiang ke-%d ke tiang ke-%d.", n, start, dest)  ELSE  int other  other = 6 - (start + dest)  IF (other<1 || other>3) THEN  WRITE ("Masukan tidak sesuai. Mohon masukkan nomor tiang yang berbeda.")  sleep(2)  system("cls")  CALL hitungHanoiR()  END IF  CALL hanoiRekursif(n-1, start, other)  WRITE ("Cakram %d dipindahkan dari tiang ke-%d ke tiang ke-%d.", n, start, dest)  CALL hanoiRekursif(n-1, other, dest)  END IF  END PROCEDURE  PROCEDURE hanoiIteratif(int n, int start, int dest)  DECLARE  int tiang[3]  tiang[0] = start  tiang[2] = dest  int langkahMin = pow(2, n)-1  WRITE ("Langkah minimal untuk memindahkan %d cakram adalah %d langkah.\n\n", n, langkahMin)  DECLARE  int other  other = 6 - (start + dest)  IF (other<1 || other>3) THEN  WRITE ("Masukan tidak sesuai. Mohon masukkan nomor tiang yang berbeda.")  sleep(2)  system("cls")  CALL hitungHanoiI()  END IF  DECLARE  tiang[1] = other  IF (n%2==0) THEN  tiang[1] = dest;  tiang[2] = other;  END IF  FOR (int i=1; i<=langkahMin; i++)  WRITE ("Cakram teratas dipindahkan dari tiang ke-%d ke tiang ke-%d.\n", tiang[(i&i-1)%3], tiang[((i|i-1)+1)%3])  END FOR  END PROCEDURE  PROCEDURE hitungHanoiR()  DECLARE  int cakram  int tiang1  int tiang2  WRITE (" \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_")  WRITE ("|+ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ +|")  WRITE ("|| ||")  WRITE ("|| MENARA HANOI REKURSIF ||")  WRITE ("|| Penyelesaian Menara Hanoi dengan Metode Rekursif ||")  WRITE ("|| ||")  WRITE ("|+ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ +|")  WRITE (" \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_")  WRITE ("Masukkan jumlah cakram : ")  CALL validasiHanoi(&cakram)  WRITE ("Nomor tiang permulaan : ")  CALL validasiTiangHanoi(&tiang1)  WRITE ("Nomor tiang tujuan : ")  CALL validasiTiangHanoi(&tiang2)  WRITE (" ")  int langkahMin = pow(2, cakram)-1  WRITE ("Langkah minimal untuk memindahkan %d cakram adalah %d langkah.", cakram, langkahMin)  CALL hanoiRekursif(cakram, tiang1, tiang2)  WRITE ("\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\n")  END PROCEDURE  PROCEDURE hitungHanoiI()  DECLARE  int cakram  int tiang1  int tiang2  WRITE (" \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_")  WRITE ("|+ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ +|")  WRITE ("|| ||")  WRITE ("|| MENARA HANOI ITERATIF ||")  WRITE ("|| Penyelesaian Menara Hanoi dengan Metode Rekursif ||")  WRITE ("|| ||")  WRITE ("|+ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ +|")  WRITE ("\t\t \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_")  WRITE ("Masukkan jumlah cakram : ")  CALL validasiHanoi(&cakram)  WRITE ("Nomor tiang permulaan : ")  CALL validasiTiangHanoi(&tiang1)  WRITE ("Nomor tiang tujuan : ")  CALL validasiTiangHanoi(&tiang2)  WRITE (" ")  CALL hanoiIteratif(cakram, tiang1, tiang2)  WRITE (" \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_")  END PROCEDURE  PROCEDURE menu()  DECLARE  int pilihan  WRITE ("\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_");  WRITE ("|+ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ +|")  WRITE ("|| ||")  WRITE ("|| M E N A R A H A N O I ||")  WRITE ("|| Mencari Langkah Penyelesaian Menara Hanoi ||")  WRITE ("|| ||")  WRITE ("|+ ---------------------------------------------------- +|")  WRITE ("|| Pilih Metode Penyelesaian Menara Hanoi ||")  WRITE ("|| ||")  WRITE ("|| [1] Penyelesaian secara Rekursif ||")  WRITE ("|| [2] Penyelesaian secara Iteratif ||")  WRITE ("|| ||")  WRITE ("|+ ---------------------------------------------------- +|")  WRITE ("|| Menu Lainnya ||")  WRITE ("|| [3] Kredit ||")  WRITE ("|| [4] Keluar dari program ||")  WRITE ("|+ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ +|")  WRITE (" \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_")  WRITE (" \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_")  WRITE ("Pilih menu yang diinginkan: ")  CALL validasiMenu(&pilihan, 1, 5)  IF (pilihan == 1) THEN  system("cls")  CALL hitungHanoiR()  ELSE IF (pilihan == 2) THEN  system("cls")  CALL hitungHanoiI()  ELSE IF (pilihan == 3) THEN  system("cls")  CALL kredit()  ELSE IF (pilihan == 4) THEN  system("cls")  CALL keluar()  END IF  CALL repeat()  END PROCEDURE  PROCEDURE repeat()  DECLARE  int pilihanUlang  WRITE ("Apakah Anda ingin kembali ke menu utama?")  WRITE ("Ketik [1] untuk 'Ya' dan [2] untuk 'Tidak' :")  CALL validasiMenu(&pilihanUlang, 1, 2)  IF (pilihanUlang==1) THEN  system("cls")  CALL menu()  ELSE IF (pilihanUlang==2) THEN  system("cls")  CALL keluar()  END IF  END PROCEDURE  PROCEDURE kredit()  WRITE (" \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_")  WRITE ("|+ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ +|")  WRITE ("|| ||")  WRITE ("|| K E L O M P O K 2 3 ||")  WRITE ("|| PROGRAM MENARA HANOI REKURSIF DAN ITERATIF ||")  WRITE ("|| ||")  WRITE ("|| -------------------------------------------------------------------- ||")  WRITE ("|| Nama Anggota Kelompok ||")  WRITE ("|| - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - ||")  WRITE ("|| > 2205551005 | Gede Made Rapriananta Pande ||")  WRITE ("|| > 2205551003 | Ida Bagus Paalakaa RNB ||")  WRITE ("|| > 2205551069 | Ni Kadek Ari Diah Lestari ||")  WRITE ("|| > 2205551072 | I Gusti Ayu Krisna Kusuma Dewi ||")  WRITE ("|| > 2205551076 | Kadek Yogi Dwi Putra Utama ||")  WRITE ("|| > 2205551079 | Anak Agung Indi Kusuma Putra ||")  WRITE ("|+ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ +|")  WRITE (" \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ ")  system("pause")  system("cls")  CALL menu()  END PROCEDURE  PROCEDURE void keluar()  WRITE (" \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_")  WRITE ("|+ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_+|")  WRITE ("|| ||")  WRITE ("|| T E R I M A K A S I H ||")  WRITE ("|| Sampai jumpa di lain waktu! ||")  WRITE ("|| ||")  WRITE ("|+ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_+|")  WRITE (" \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_");  exit(0)  END PROGRAM  END PROCEDURE  FUNCTION int main()  CALL menu()  return 0  END FUNCTION  END |

**Kode Program 4.5** *Pseudocode* Program Menara Hanoi

Kode Program 4.5 merupakan *pseudocode* dari program Menara Hanoi. *Pseudocode* ini merupakan implementasi dari *flowchart* yang sebelumnya telah disusun. Dengan *pseudocode*, kode program pun dapat disusun secara lebih matang dan lebih mudah dalam mengomunikasikan, mempelajari, maupun menyusun struktur kode program tersebut.

#### Trace Flowchart Program Menara Hanoi

*Trace* merupakan suatu prosedur yang tidak dapat dilewatkan dalam menyusun suatu alur algoritma khususnya *flowchart*. *Trace* dilakukan untuk memeriksa apakah alur algoritma tersebut telah berjalan dengan baik dan membantu dalam memperbaiki *error* bila ada. Berikut adalah hasil *trace* dari program Menara Hanoi.

**Tabel 4.5** *Trace* Program Menara Hanoi

|  |  |
| --- | --- |
| *Input* | pilihan = 1  cakram = 3  tiang1 = 1  tiang2 = 3  pilihanUlang = 1  pilihan = 2  cakram = 3  tiang1 = 1  tiang2 = 3  pilihanUlang = 2 |
| Proses | CALL menu()  CALL hitungHanoiR() dengan prosedur rekursif hanoiRekursif  (cakram, tiang1, tiang2) di dalamnya.  CALL repeat()  CALL menu()  CALL hitungHanoiI()  CALL repeat()  CALL keluar() |
| *Output* | Instruksi pilih menu  Instruksi masukkan cakram, tiang1, tiang2  Langkah minimal untuk memindahkan 3 cakram adalah 7 langkah.  Cakram 1 dipindahkan dari tiang ke-1 ke tiang ke-3.  Cakram 2 dipindahkan dari tiang ke-1 ke tiang ke-2.  Cakram 1 dipindahkan dari tiang ke-3 ke tiang ke-2.  Cakram 3 dipindahkan dari tiang ke-1 ke tiang ke-3.  Cakram 1 dipindahkan dari tiang ke-2 ke tiang ke-1.  Cakram 2 dipindahkan dari tiang ke-2 ke tiang ke-3.  Cakram 1 dipindahkan dari tiang ke-1 ke tiang ke-3.  Kembali ke menu utama?  Instruksi pilih menu  Instruksi masukkan cakram, tiang1, tiang2  Langkah minimal untuk memindahkan 3 cakram adalah 7 langkah.  Cakram teratas dipindahkan dari tiang ke-1 ke tiang ke-3.  Cakram teratas dipindahkan dari tiang ke-1 ke tiang ke-2.  Cakram teratas dipindahkan dari tiang ke-3 ke tiang ke-2.  Cakram teratas dipindahkan dari tiang ke-1 ke tiang ke-3.  Cakram teratas dipindahkan dari tiang ke-2 ke tiang ke-1.  Cakram teratas dipindahkan dari tiang ke-2 ke tiang ke-3.  Cakram teratas dipindahkan dari tiang ke-1 ke tiang ke-3.  Kembali ke menu utama?  Terima kasih |

Tabel 4.5 merupakan contoh dari pelaksanaan *trace* pada program Menara Hanoi yang mana membuktikan bahwa algoritma yang disusun sudah benar. Pengguna akan memberikan *input* untuk menentukan pilihan menu yang diinginkan, banyaknya cakram serta tiang awal dan tujuan, serta pilihan untuk mengulang atau keluar dari program.

Pada *trace* tersebut, pengguna memilih untuk menyelesaikan teka-teki Menara Hanoi dengan menggunakan metode rekursif, sehingga program pun memanggil prosedur *hitungHanoiR()* dengan prosedur rekursif *hanoiRekursif (cakram, tiang1, tiang2)* di dalamnya. Pengguna memasukkan banyak cakram sebesar 3, tiang 1 sebagai awal dan tiang 3 sebagai akhir. Setelah diproses di dalam prosedur, program pun memberikan *output* berupa langkah minimal yang diperlukan beserta langkah-langkah yang dapat ditempuh. Pada prosedur *repeat()*, pengguna lantas memilih untuk mengakhiri program, sehingga program pun ditutup melalui prosedur *keluar()* dan memberikan *output* berupa ucapan terima kasih.

### Flowchart, Pseudocode, dan Trace Flowchart Program Angsuran

Program ini dapat menghitung angsuran yang perlu dibayarkan dengan berdasarkan pada pokok pinjaman, besar bunga (menurun) serta lama pinjaman dengan rumus yang telah ditentukan. *Input* yang diterima adalah pokok pinjaman, besar bunga (%), dan lama pinjaman. Program ini pun dapat memberikan *output* berupa rincian angsuran setiap bulan, bunga per bulan, angsuran pokok, dan total angsuran. Berikut adalah *flowchart, pseudocode,* serta *trace* dari program angsuran.

#### Flowchart Program Angsuran

*Flowchart* adalah sekumpulan simbol-simbol yang menjelaskan mengenai suatu alur proses atau langkah-langkah yang berurutan dari program tersebut yang harus diikuti oleh pemrogram. Dapat disimpulkan, bahwa *flowchart* digunakan untuk menunjukkan alur proses dari sebuah program secara sistematis hingga selesai.

|  |
| --- |
|  |

**Gambar 4.22** *Flowchart* Subproses Validasi *Input*

Gambar 4.22 di atas merupakan *flowchart* subproses *validasi()* dari program angsuran. Pada flowchart tersebut terdapat validasi, dimana inputan yang diterima hanya berupa angka saja, selain itu akan diminta untuk menginput ulang.

|  |
| --- |
|  |

**Gambar 4.23** *Flowchart* Subproses Validasi *Range*

Gambar 4.23 di atas merupakan *flowchart* subproses *validasiRange()* dari program angsuran. Pada flowchart tersebut terdapat validasi, di mana *input* yang diterima hanya berupa angka dan angka yang diterima hanya angka 1 atau 2 saja, selain itu akan diminta untuk meng-*input* ulang.

|  |
| --- |
|  |

**Gambar 4.24** *Flowchart* Subproses Angsuran

Gambar 4.24 di atas merupakan *flowchart* subproses *angsuran()* dari program angsuran. Pada *flowchart* tersebut dimulai dengan mendeklarasikan beberapa variabel. Kemudian lanjut ke proses memasukkan pokok pinjaman dan terdapat validasi, selanjutnya masuk ke proses memasukkan besaran bunga dan terdapat validasi, dan proses memasukkan lama pinjaman dalam bulan. Setelah itu data inputan akan dihitung berdasarkan suatu rumus. Kemudian untuk menampilkan hasilnya, maka digunakan perulangan, dan *flowchart* selesai.

|  |
| --- |
|  |

**Gambar 4.25** *Flowchart Int Main* Program Angsuran

Gambar 4.25 di atas merupakan *flowchart main()* dari program angsuran. Pada *flowchart* tersebut menampilkan nama kelompok, kemudian masuk ke subproses *angsuran()* dan pertanyaan apakah ingin mengulang perhitungan dan berisi validasi. Pada *flowchart* tersebut digunakannya perulangan *do-while*, dimana jika variabel *read* bernilai 1, maka akan terjadi perulangan mulai dari subproses *angsuran()*.

#### Pseudocode Program Angsuran

*Pseudocode* adalah suatu cara menulis algoritma yang mirip dengan bahasa pemrograman, tetapi *pseudocode* lebih sederhana dan mudah dipahami oleh manusia pada umumnya. *Pseudocode* memudahkan user untuk menyelesaikan suatu masalah tanpa memikirkan sintaks dari suatu bahasa pemrograman. Berikut ini *pseudocode* dari program angsuran.

|  |
| --- |
| Algoritma Menghitung Angsuran  FUNCTION validasiMenu(): void  DEKLARASI:  input: int  DESKRIPSI:  BEGIN  IF input == 1 && input == 2 THEN  TRUE  ELSE  ERROR  write “masukkan angka dengan benar”  END  FUNCTION validasiBunga(): void  DEKLARASI:  input: float  DESKRIPSI:  BEGIN  IF input >= 0 THEN  TRUE  ELSE  ERROR  write “masukkan angka dengan benar”  END  FUNCTION validasiAngka(): void  DEKLARASI:  input: float  DESKRIPSI:  BEGIN  IF input >= 1 THEN  TRUE  ELSE  ERROR  write “masukkan angka dengan benar”  END  FUNCTION namaKelompok(): void  DESKRIPSI:  BEGIN  write “nama kelompok”  system(pause)  system(cls)  END  FUNCTION angsuran(): int  DEKLARASI:  pokokPinjaman: int  bungaSetahun: float  lamaPinjamanBulan: int  angsuranPokok: int  bungaBulanY = 0: int  angsuranPerbulan = 0: int  DESKRIPSI:  BEGIN  write “masukkuan pokok pinjaman”  read(pokokPinjaman)  validasiAngka()  write “masukkan besar bunga”  read(bungaSetahun)  validasiBunga()  write “masukkan lama pinjaman dalam bulan”  read(lamaPinjamanBulan)  validasiAngka()  angsuranPokok = pokokPinjaman/lamaPinjamanBulan  bungaSetahun = bungaSetahun/100  write “bulan ke | bunga | angsuran pokok | angsruan perbulan”    FOR (i = 1: int; i <= lamaPinjamanBulan; i++)  bungaBulanY = (pokokPinjaman - ((i-1) \* angsuranPokok)) \* bungaSetahun/12    angsuranPerbulan = bungaBulanY + angsuranPokok    IF i < 10 THEN  write “%d | rp. %d | rp. %d | rp. %d”, i, bungaBulanY, angsuranPokok, angsuranPerbulan  ELSE IF bungaBulanY < 1000 THEN  write “%d | rp. %d | rp. %d | rp. %d”, i, bungaBulanY, angsuranPokok, angsuranPerbulan  ELSE THEN  write “%d | rp. %d | rp. %d | rp. %d”, i, bungaBulanY, angsuranPokok, angsuranPerbulan  END FOR  system(pause)  END    FUNCTION main(): int  DEKLARASI:  read: int  DESKRIPSI:  BEGIN  namaKelompok()    DO    system(cls)  write “menghitung angsuran”  angsuran()  system (cls)    write “apakah ingin mengulang? 1 = iya | 2 = tidak”  read(read)  validasiMenu()  WHILE read == 1  system(cls)  write “Terima Kasih”  END |

**Kode Program 4.6** *Pseudocode* Program Angsuran

*Pseudocode* pada Kode Program 4.6 akan membantu *programmer* dalam membuat program yang sebenarnya. *Pseudocode* ini sudah dapat diimplementasikan ke dalam sebuah program yang sudah siap digunakan dengan menambah sintaks-sintaks program yang kurang.

#### Trace Program Angsuran

*Trace* pada program merupakan salah satu hal yang krusial. *Trace* berguna untuk mengetahui alur program bekerja dan dapat digunakan untuk mendeteksi kesalahan yang ada pada program sehingga kesalahan bisa ditemukan dan diperbaiki. *Trace* pada program konversi bisa dilihat pada tabel di bawah ini.

**Tabel 4.6** *Trace* Program Angsuran

|  |  |
| --- | --- |
| *Input* | pinjaman pokok = 1000000  besar bunga = 10  lama pinjaman = 12  Ingin Mencoba Menginput Kembali (1/2) = 2 |
| Proses | CALL angsuran()  angsuranPokok = pokokPinjaman/lamaPinjamanBulan;  bungaSetahun = bungaSetahun/100;  for ( int i = 1; i <= lamaPinjamanBulan;i++){  bungaBulanY = (pokokPinjaman - ((i-1) angsuranPokok)) \* bungaSetahun/12;  angsuranPerbulan = bungaBulanY + angsuranPokok;  } |
| *Output* | Tampilkan angsuran()  Terima kasih |

Tabel 4.6 merupakan hasil *trace* dari program konversi. Trace di atas menjelaskan tentang bagaimana alur program ini berjalan, pertama pengguna akan meng-*input* pinjaman pokok, besar bunga dan lama pinjaman, pada trace ini pinjaman pokok = 1000000, besar bunga = 10, lama pinjaman = 12. Setelah itu program akan berjalan dan akan memanggil fungsi angsuran dan menghitung angsuran tiap bulan berdasarkan jumlah yang tadi di-*input*-kan user. Program kemudian akan menampilkan hasil dari perhitungan. Lalu program akan menampilkan menu ulang untuk menanyakan pengguna apakah ingin mengulang program (1 = ya, 2 = tidak), jika pengguna memilih 1 maka program akan kembali berulang, tapi jika pengguna memilih 2 maka menu terima kasih akan muncul dan program akan berakhir.

# Daftar Pustaka

Agarwal, R., 2022. *Recursion Vs Iteration |10 Differences (& When to use?).* [Online]   
Available at: https://favtutor.com/blogs/recursion-vs-iteration  
[Diakses 26 Oktober 2022].

GeeksforGeeks, 2022. *Decision Making in C / C++ (if , if..else, Nested if, if-else-if ).* [Online]   
Available at: https://www.geeksforgeeks.org/decision-making-c-cpp/  
[Diakses 25 Oktober 2022].

Hanly, J. R. & Koffman, E. B., 2013. *Problem Solving and Program Design.* 7th penyunt. New Jersey: Pearson Education, Inc..

IBM, 2021. *The if Statement.* [Online]   
Available at: https://www.ibm.com/docs/en/zos/2.3.0?topic=ss-if-statement  
[Diakses 25 Oktober 2022].

Microsoft, 2021. *for Statement (C).* [Online]   
Available at: https://learn.microsoft.com/en-us/cpp/c-language/for-statement-c?view=msvc-170  
[Diakses 26 Oktober 2022].

TechDifference, 2018. *Difference Between Recursion and Iteration.* [Online]   
Available at: https://techdifferences.com/difference-between-recursion-and-iteration-2.html  
[Diakses 26 Oktober 2022].

Thompson, B., 2022. *C Conditional Statement: IF, IF Else and Nested IF Else with Example.* [Online]   
Available at: https://www.guru99.com/c-if-else-statement.html  
[Diakses 25 Oktober 2022].

Thompson, B., 2022. *Loops in C: For, While, Do While looping Statements [Examples].* [Online]   
Available at: https://www.guru99.com/c-loop-statement.html  
[Diakses 26 Oktober 2022].

Thompson, B., 2022. *switch…case in C (Switch Statement in C) with Examples.* [Online]   
Available at: https://www.guru99.com/c-switch-case-statement.html  
[Diakses 25 Oktober 2022].

Tiwari, B., 2022. *Conditional Statements : if, else, switch.* [Online]   
Available at: https://www.dotnettricks.com/learn/c/conditional-statements-if-else-switch-ladder  
[Diakses 25 Oktober 2022].