***Modular :***

***Non-Recursif dan Recursif***

***Metode Sorting:***

***Internal Sorting***

**Sanyata Purwidayanta, MT**



Counting

Insertion

Selectiom

Exchange

….

**1**

**2**

**3**

**4**

**5**

**6**

**7**

**8**

**9**

**10**

**L**

**K**

**J**

**I**

**H**

**G**

**F**

**E**

**D**

**C**

**B**

A



DAFTAR ISI

[DAFTAR ISI i](#_Toc86621998)

[BAB 1 KONSEP TENTANG DATA DAN TIPE 1](#_Toc86621999)

[1.1 Konsep Data 1](#_Toc86622000)

[1.2 Tipe Data 3](#_Toc86622001)

[1.2.1 Tipe Integer 4](#_Toc86622002)

[1.2.2 Tipe Real 5](#_Toc86622003)

[1.2.3 Tipe Character 6](#_Toc86622004)

[1.2.4 Tipe Boolean 7](#_Toc86622005)

[1.2.5 Tipe String 8](#_Toc86622006)

[1.2.6 Tipe Enumerasi 9](#_Toc86622007)

[1.2.7 Tipe Subrange 10](#_Toc86622008)

[1.2.8 Tipe Set 11](#_Toc86622009)

[1.2.9 Tipe Pointer 12](#_Toc86622010)

[1.3 Identitas dan Ekspresi Data 13](#_Toc86622011)

[1.3.1 Identitas Data 13](#_Toc86622012)

[1.3.2 Ekspresi Data 15](#_Toc86622013)

[BAB 2 SKEMA ALGORITMA MODULAR TIPE NON-REKURSIF 17](#_Toc86622014)

[2.1 Skema Algoritma Modular 17](#_Toc86622015)

[2.2 Skema Prosedur 20](#_Toc86622016)

[2.3 Skema Fungsi 23](#_Toc86622017)

[2.4 Pelewatan Parameter (Passing Parameter) 26](#_Toc86622018)

[2.5 Skop dan Penamaan Variabel 31](#_Toc86622019)

[2.6 Studi Kasus Implementasi Skema Turbo Pascal 33](#_Toc86622020)

[2.7 Studi Kasus Aplikasi Spreadsheet Excel 42](#_Toc86622021)

[2.7.1 Modul “Statistical” 43](#_Toc86622022)

[2.7.2 Modul “Lookup & Reference” 46](#_Toc86622023)

[2.7.3 Modul “Text & Data” 47](#_Toc86622024)

[2.7.4 Modul “Logical” 48](#_Toc86622025)

[2.7.5 Modul “Math & Trigonometry” 48](#_Toc86622026)

[2.7.6 Modul “Database” 50](#_Toc86622027)

[2.7.7 Modul “Financial” 51](#_Toc86622028)

[2.7.8 Modul “Date & Time” 53](#_Toc86622029)

[2.7.9 Modul “Information” 53](#_Toc86622030)

[2.7.10 Modul “Engineering” 54](#_Toc86622031)

[2.7.11 Modul “Cube” 56](#_Toc86622032)

[2.8 SOAL-SOAL LATIHAN. 57](#_Toc86622033)

[BAB 3 SKEMA ALGORITMA PROSEDUR DAN FUNGSI MODEL REKURSIF 59](#_Toc86622034)

[3.1 Pengertian 59](#_Toc86622035)

[3.2 Contoh Implementasi Turbo Pascal 60](#_Toc86622036)

[BAB 4 ALGORITMA PEMROSESAN FILE SEKUENSIAL DAN FILE ACAK 64](#_Toc86622037)

[4.1 File Sekuensial 64](#_Toc86622038)

[4.1.1 Definisi File Sekuensial 64](#_Toc86622039)

[4.1.2 Prosedur dan Fungsi Primitif 65](#_Toc86622040)

[4.1.3 Algoritma Pengolelolaan File Sekuensial 66](#_Toc86622041)

[4.2 File Acak 69](#_Toc86622042)

[4.2.1 Definisi File Acak 69](#_Toc86622043)

[4.2.2 Algoritma Pengolahan Data 69](#_Toc86622044)

[BAB 5 METODE ALGORITMA PADA PROSES SORTING INTERNAL 76](#_Toc86622045)

[5.1 Tipe Algoritma Sorting 76](#_Toc86622046)

[5.2 Internal Sorting 77](#_Toc86622047)

[5.2.1 Metode Counting 77](#_Toc86622048)

[5.2.2 Metode Insertion 79](#_Toc86622049)

[5.2.2.1 Natural\_InsertionSort 79](#_Toc86622050)

[5.2.2.2 Binary\_InsertionSort 80](#_Toc86622051)

[5.2.2.3 Shell\_InsertionSort 81](#_Toc86622052)

[5.3 Metode Selection 83](#_Toc86622053)

[5.3.1 Natural\_SelectionSort 83](#_Toc86622054)

[5.3.2 Heap\_SelectionSort 83](#_Toc86622055)

[5.4 Metode Exchange 84](#_Toc86622056)

[5.4.1 Natural\_BubbleSort 84](#_Toc86622057)

[5.4.2 Optimal\_BubbleSort 85](#_Toc86622058)

[5.4.3 Flag\_BubbleSort 85](#_Toc86622059)

[5.4.4 Shaker\_ ExchangeSort 86](#_Toc86622060)

[5.4.5 Quick\_ExchangeSort 87](#_Toc86622061)

[5.4.6 Radix\_ExchangeSort 88](#_Toc86622062)

[5.4.7 Merge\_ExchangeSort 89](#_Toc86622063)

[DAFTAR PUSTAKA 90](#_Toc86622064)

[RIWAYAT PENULIS 91](#_Toc86622065)

# KONSEP TENTANG DATA DAN TIPE

## Konsep Data

Konsep data dalam kehidupan sehari-hari berkaitan dengan apa yang disebut fakta. Fakta adalah sebuah objek nyata yang menjadi instan (bukti materiil) dalam kehidupan sehari-hari yang dapat berupa perwujudan benda/materi, peristiwa/kejadian, mapun perilaku. Objek nyata yang bukan materiil (immateriil) atau yang gaib sementara dikesampingkan dulu. Objek nyata sebagai sebuah fakta dikenal atau diidentifikasi dengan menggunakan sekumpulan atribut yang melekat pada fakta tersebut, dan minimal memiliki atribut nama. atribut nama atau identitas dari objek tersebut harus sudah dipahami atau sudah dikenal dalam ilmu pengetahuan. Atribut selain nama merupakan informasi tambahan yang menjelaskan ciri atau karakter objek tersebut agar lebih dapat dipahami, seperti ukuran, bentuk, alamat, warna, dan lain sebagainya. Data adalah representasi (perwujudan) simbolik yang menggambarkan tentang fakta. Representasi simbolik bisa berupa angka, kata/ tulisan, gambar, video, suara, dan kombinasinya. Representasi simbolik ini dalam proses komputer disebut data yang dikelompokan dalam bentuk tipe data. Tipe data inilah yang menjadi bagian penting dalam proses transformasi masalah dan solusi kehidupan sehari-hari ke dalam data yang siap diproses menggunakan mesin komputer

Data adalah sebuah ekspresi umum yang mendeskripsikan/menyatakan segala hal yang dioperasikan oleh komputer pada tingkat kode mesin, dan direpresentasikan dalam bentuk rentetan digit biner (*bits*) [Wirth 74]. Data sebagai sebuah ekspresi umum merepresentasikan sekumpulan fakta atau kejadian terbatas dari dunia nyata yang dinyatakan dalam bentuk nama atau simbolik. Beberapa contoh data yang mewakili kehidupan sehari-hari dapat dideskripsikan dengan contoh berikut ini.

1. Data warna pelangi yaitu merah, jingga, kuning, hijau, biru, nila, dan ungu. Sedangkan dalam skop yang lebih luas dapat dinyatakan data warna meliputi merah, jingga, kuning, hijau, biru, nila, ungu, hitam, putih, coklat, dan seterusnya.
2. Data jumlah kecelakaan motor dari tahun 2000 – 2010 yaitu 1000, 1200, 1225, 1500, 1300, 1400, 1350, 1500, 1600, 1700.
3. Data abjad yaitu ’A’, ’B’, ’C’, dan seterusnya.
4. Data bilangan bulat atau *integer* yaitu (-...), (-3), (-2), (-1), 0, 1, 2, 3, ..., (+...)
5. Data logika yaitu TRUE atau FALSE
6. Data nama yaitu ’Sanyata’, ’Purwidayanta’, ’Robert’, dan seterusnya
7. Data hari dalam seminggu yaitu ’Senin’, ’Selasa’, ’Rabu’, ’Kamis’, ’Jumat’, ’Sabtu’, dan ’Minggu’.
8. Data mahasiswa dengan merupkan data agak komposisi yana agak kompleks struktur komposisinya misal struktur <NIM, NAMA, NILAI MATAKULIAH ALGORITMA> maka contoh datanya yaitu <’001’, ’Ali’, 100>, <’002’, ’Budi’, 60>, <’003’, ’Cyntia’, 90>, <’004’, ’Diana’, 75>, dan seterusnya.

Dalam kehidupan sehari-hari, data merupakan representasi fakta dunia nyata yang berupa nama atau simbol. Dalam komputer digital, simbol ini dimanipulasi secara numerik dan selanjutnya diterjemahkan dalam bentuk data digital. Misalnya manipulasi data warna dalam beraneka simbol diperlihatkan pada tabel di bawah.

Contoh manipulasi data warna dalam aneka simbol

|  |  |  |  |
| --- | --- | --- | --- |
| **Kode**  **Desimal** | **Kode Biner**  **8-bit** | **Nama Data Warna** | **Representasi Sinyal Digital** |
| 0 | 00000000 | Black |  |
| 1 | 00000001 | Blue |  |
| 2 | 00000010 | Green |  |
| 3 | 00000011 | Cyan |  |
| 4 | 00000100 | Red |  |
| 5 | 00000101 | Magenta |  |
| 6 | 00000110 | Brown |  |
| 7 | 00000111 | White |  |
| 8 | 00001000 | gray |  |
| 9 | 00001001 | Light Blue |  |
| 10 | 00001011 | Light Green |  |
| 11 | 00001100 | Light Cyan |  |
| 12 | 00001101 | Light Red |  |
| 13 | 00001111 | Light Magenta |  |
| 14 | 00010000 | Yellow |  |

Sebuah konsep yang terkait tentang data disebut tipe data (*data type*). Tipe data adalah konsep yang mendefinisikan sekumpulan nilai (*set of values*) yang merepresentasikan skop (*domain*) data tertentu. Berkaitan dengan pendefinisian suatu variabel, maka tipe data dapat diartikan sebagai sekumpulan nilai yang mungkin diberikan pada sebuah variabel yang diasumsikan/didefinisikan [Wirth 74].

Berorientasi pada mekanisme kerja mesin komputer model *Von Newmann* dan berdasarkan kumuman fasilitas dalam bahasa pemrograman*,* tipe data sebagai suatu konsep memiliki empat karakteristik dalam definisi yaitu :

1. Tipe data memiliki unsur elementer sebagai nama pengenal, contoh : *integer, real, character, boolean, string,* dan lain-lain
2. Tipe data memiliki skop (*domain*) nilai yang akan diproses, contoh tipe *integer* memiliki skop nilai antara -216 s/d +216
3. Tipe data memiliki konvensi/perjanjian tentang penulisan konstanta, contoh tipe *integer* memiliki konvensi penulisan konstanta C = 1000
4. Tipe data memiliki operator yang digunakan untuk mengoperasikan data tersebut, contoh tipe *integer* memiliki operator Plus (+), Minus (-), Kali (\*), dan lain-lain.

Operator-operator ini biasanya dibedakan menjadi beberapa jenis meliputi :

1. Operator Aritmetika
2. Operator Logika
3. Operator Ralasional

## Tipe Data

Tipe data dibedakan menjadi dua macam yaitu tipe sederhana /data dasar dan tipe kompleks (bentukan atau struktur atau komposisi). Tipe data bentukan dikomposisi berdasarkan tipe data sederhana dan menjadi kompleks tergantung pada konsep abtraksi pembentukan struktur datanya. Tipe data bentukan ini seringkali disebut dengan istilah struktur data. Struktur ada adalah konsep atau abstraksi yang menyatakan koleksi dari objek-objek data yang tersusun secara spesifik beserta sekumpulan operasi yang dapat dilakukan terhadap koleksi tersebut. Jadi, seringkali istilah struktur data digunakan untuk penyebutan tipe data yang kompleks.

Pondasi dari tipe data sederhana adalah tipe skalar (*scalar*) yang bersifat statis dan *pointer* (*address* atau alamat memori utama/RAM) yang bersifat dinamis. Tipe skalar adalah definisi yang menyatakan sekumpulan nilai-nilai yang berbeda yang memiliki keterurutan sehingga dapat dipahami nilai-nilai sebelumnya atau nilai-nilai sesudahnya. Tipe skalar dibedakan menjadi dua yaitu skalar dasar dan skalar turunan. Skalar dasar adalah tipe yang menjadi dasar untuk membuat tipe data baru baik yang berupa tipe skalar turunan maupun tipe data kompleks. Tipe skalar dasar terdiri empat unsur yaitu *integer, real, character,* dan *boolean* dengan catatan bahwa meskipun *real* termasuk dalam kategori skalar tetapi tidak dapat selalu digunakan dalam konteks yang sama dengan skalar lainnya, misalnya dalam *real* tidak bisa dipakai fungsi *predecessor* (sebelum nilai tertentu) dan *successor* (setelah nilai tertentu). **Catatan** bahwa meskipun *Real* termasuk dalam tipe skalar tetapi tidak dapat digunakan dengan konteks yang sama dengan tipe skalar lainnya.

Tipe skalar adalah tipe yang memiliki karakteristik sebagai sekumpulan nilai berbeda dan mempunyai keterurutan secara linear. Tipe data ini terdiri dari tipe yang terstruktur (structured) yaitu tipe data dasar dan tidak terstruktur (*unstructured*) yang terdiri dari tipe skalar turunan. Tipe skalar turunan merupakan formulasi yang lebih spesifik atau *subset* dari tipe skalar dasar yang membentuk tipe data skalar baru (turunan).

**Tipe Data**

**(Data Type)**

**Sederhana**

**(Dasar)**

Skalar Dasar

1. *Integer*
2. *Real* (kontek skalarnya bersifat khusus)
3. *Character*
4. *Boolean*

Skalar Turunan

1. *String*
2. *Enumeration*
3. *Subrange*
4. *Set*

**Kompleks**

**(Bentukan)**

9. *Pointer*

**Statis**

**(Skalar)**

Bentukan Dasar

1. *Record*
2. *Array*
3. *File*
4. *object*

Bentukan Turunan (Abstrak)

1. *LIST*
2. *STACK*
3. *TREE*
4. *GRAPH*
5. *TOPOLOGICAL*
6. dan seterusnya

**Dinamis**

* 1. Skema pengelompokan tipe data

Untuk memudahkan pembahasan, tipe data dibedakan menjadi tipe data sederhana dan kompleks. Selanjutnya berdasarkan karakteristiksnya tipe data tersebut dikelompokkan sampai menjadi unsur-unsur tipe elementer tertentu, yaitu *pointer, integer, real, character, boolean, string, enumeration, subrange, set, record, array, file, object, LIST, STACK, TREE, GRAPH, TOPOLOGICAL*, dan seterusnya, seperti diilustrasikan dalam gambar di atas.

Tipe data didefinisikan dengan cara menguraikan karakteristik yang dimiliki setiap unsur elementernya. Sebagai informasi tambahan bahwa tipe data umumnya memiliki fungsi standar yang telah disediakan (*bulit-in*) bahasa pemrograman, contoh tipe *integer* dalam bahasa *pascal* memiliki fungsi misalnya *absoulut* (harga mutlak), *truncate* (pemotongan digit desimal dari tipe *real*), *rounded* (pembulatan dari tipe *real*), dan lain-lain. Karakteristik ini bersifat umum dan spesifik untuk setiap bahasa pemrograman.

### Tipe Integer

Integer merupakan tipe yang memiliki skop terbatas bilangan bulat (bergantung panjang bit data pada komputer). Secara umum terdiri dari - ~, …, -3, -2, -1, 0, 1, 2, 3, …, + ~. Misalnya untuk integer 16 bit maka skopnya –216 s/d +(216-1).

Konvensi konstanta tipe ini adalah Konstanta C = 275 atau C = 5

Operator yang digunakan sehingga menghasilkan skop hasil *integer*:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Jenis**  **Operator** | **Simbol**  **Operator** | **Arti**  **Operator** | **Contoh**  **pernyataan/ ekspresi** | **Hasil** |
| Aritmetika | \* | kali | 3 \* 5 | 15 |
|  | div | bagi & potong pecahan | 5 div 2  5 div 3 | 2  1 |
|  | mod | sisa pembagian | 5 mod 2  5 mod 3 | 1  2 |
|  | ^ | pangkat | 5 ^ 3 | 125 |
|  | + | tambah | 5 + 3 | 8 |
|  | - | kurang | 5 – 3 | 2 |
| Contoh  Fungsi standar | abs (x) | absoulut | abs(-5) | 5 |
| trunc(x) | bagi & potong pecahan | trunc(5,75)  trunc(5,45) | 5  5 |
| round(x) | sisa pembagian | round(5,75)  round(5,45) | 6  5 |
| sqr(x) | kuadrat | sqr(5) | 25 |
| sqrt(x) | akar kuadrat | sqrt(25) | 5 |

Operator yang digunakan sehingga menghasilkan skop hasil *boolean*:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Jenis**  **Operator** | **Simbol**  **Operator** | **Arti**  **Operator** | **Contoh**  **pernyataan/ ekspresi** | **Hasil** |
| Relasional | = | sama dengan | 5 = 3 | False |
|  | > | lebih besar | 5 > 3 | True |
|  | ≥ | lebih besar atau sama dengan | 5 ≥ 5 | True |
|  | < | lebih kecil | 5 < 3 | False |
|  | ≤ | lebih kecil atau sama dengan | 5 ≤ 5 | False |
|  | ≠ | tidak sama dengan | 5 ≠ 3 | True |

Contoh variasi cakupan tipe data Integer dalam bahasa Turbo Pascal 7.0

|  |  |  |  |
| --- | --- | --- | --- |
| **No** | **Nama Variasi Tipe** | **Format Data** | **Skop** |
| 1. | ShortInt | 8 bit bertanda | -128 s/d 127 |
| 2. | Integer | 16 bit bertanda | -32768 s/d 32767 |
| 3. | LongInt | 32 bit bertanda | -2147483648 s/d 2147483647 |
| 4. | Byte | 8 bit tak bertanda | 0 s/d 255 |
| 5. | Word | 16 bit tak bertanda | 0 s/d 65535 |

Contoh variasi cakupan tipe data integer dalam bahasa Turbo C

|  |  |  |  |
| --- | --- | --- | --- |
| **No** | **Nama Variasi Tipe** | **Format Data** | **Ruang Lingkup** |
| 1. | ShortInt | 8 bit bertanda | -128 s/d 127 |
| 2. | Integer | 16 bit bertanda | -32768 s/d 32767 |
| 3. | LongInt | 32 bit bertanda | -2147483648 s/d 2147483647 |
| 4. | Byte | 8 bit tak bertanda | 0 s/d 255 |
| 5. | Word | 16 bit tak bertanda | s/d 65535 |

Pendefinisian teks algoritma tipe dan variabel dengan contoh :

Type NamaTipe = **Integer**

Panjang = **Integer**

Variable NamaVar : NamaTipe

P : Panjang

Contoh akses algoritma

Input(P)

P 🡨 123

Output(P)

### Tipe Real

Real merupakan tipe yang memiliki skop terbatas bilangan rasional (bergantung struktur bit yang merepresentasikan bilangan *floating point* pada komputer). Banyaknya digit dibelakang titik desimal bergantung pada sistem bilangan *floating point* yang dipakai komputer. Tipe ini ditandai dengan adanya titik desimal.

Penjabaran tipe :

Konvensi konstanta Konstanta C = 2.75 atau C = 5.0

Operator yang digunakan sehingga menghasilkan skop hasil *real*:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Jenis**  **Operator** | **Simbol**  **Operator** | **Arti**  **Operator** | **Contoh**  **pernyataan/ ekspresi** | **Hasil** |
| Aritmetika | \* | kali | 3.0 \* 5.0 | 15.0 |
|  | / | bagi | 5 /2 | 2.5 |
|  | + | tambah | 5.0 + 3.0 | 8.0 |
|  | - | kurang | 5.0 – 3.0 | 2.0 |
| Contoh  Fungsi standar | abs (x) | absoulut | abs(-5.0) | 5.0 |
| sqr(x) | kuadrat | sqr(5.0) | 25.0 |

Operator yang digunakan sehingga menghasilkan skop hasil *boolean*:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Jenis**  **Operator** | **Simbol**  **Operator** | **Arti**  **Operator** | **Contoh**  **pernyataan/ ekspresi** | **Hasil** |
| Relasional | > | lebih besar | 5.0 > 3.0 | True |
|  | ≥ | lebih besar atau sama dengan | 5.0 ≥ 5.0 | True |
|  | < | lebih kecil | 5.0 < 3.0 | False |
|  | ≤ | lebih kecil atau sama dengan | 5.0 ≤ 5.0 | True |

Contoh variasi cakupan tipe data Real dalam bahasa Turbo Pascal 7.0

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No** | **Nama Variasi Tipe** | **Lebar Data** | **Ruang Lingkup** | **Ketelitian (0, ...)** |
| 1. | Real | 6 byte | -2.9 x 10-39 s/d 1.7 x 1038 | 12 digit |
| 2. | Single | 4 byte | -1.5 x 10-45 s/d 3.4 x 1038 | 8 digit |
| 3. | Double | 8 byte | -5.0 x 10-324 s/d 1.7 x 10308 | 15 digit |
| 4. | Extended | 10 byte | -3.4 x 10-4932 s/d 1.1 x 104932 | 20 digit |
| 5. | Comp | 8 byte | (-263+1) s/d (263-1) | 20 digit |

Pendefinisian teks algoritma tipe dan variabel dengan contoh :

Type NamaTppe = **Real**

Jari\_Jari = **Real**

Variable NamaVar : NamaTipe

R : Jari\_Jari

Contoh akses algoritma

Input(R)

R 🡨 123.5

Output(R)

### Tipe Character

Character merupakan tipe yang memiliki skop alphanumerik dengan panjang satu huruf/ karakter. Tipe ini ditandai dengan adanya tanda kutip. Secara umum tipe ini terdiri dari …, ‘a’, ‘b’, ‘c’, …’z’, ‘A’, ‘B’, ‘C’, …,’Z’, ‘0’, ‘1’, ‘2’,…, ‘9’, ‘@’, ‘#’, ‘$’,’/’, dan seterusnya yaitu merupakan karakter yang ada pada kode ASCII (American Standar Code International Interchange). Rincian skop tipe ini adalah :

1. Abjad/ alpabet yaitu kumpulan huruf yang memiliki keterurutan ‘a’, …’z’, ‘A’,…,’Z’
2. Digit yaitu kumpulan simbol digit desimal ‘0’, ‘1’, ‘2’,…, ‘9’
3. Spasi/ *Blank* yqitu karakter ’ ’
4. Karakter lain yang diapit dalam tanda kutip ‘@’, ‘#’, ‘$’,’/’, dan seterusnya.

Penjabaran tipe :

Konvensi konstanta tipe ini adalah Konstanta C = ‘#’ atau C = ‘9’

Operator yang digunakan sehingga menghasilkan skop hasil *character*:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Jenis**  **Operator** | **Simbol**  **Operator** | **Arti**  **Operator** | **Contoh**  **pernyataan/ ekspresi** | **Hasil** |
| Contoh  Fungsi standar | succ (x) | successor | succ(‘D’)  succ(‘d’) | ‘E’  ‘e’ |
|  | pred(x) | predecessor | pred(‘D’)  pred(‘d’) | ‘C’  ‘c’ |
|  | chr(x) | character | chr(65) | ‘A’ |
|  | ord(x) | ordinat | ord(‘A’) | 65 |

Operator yang digunakan sehingga menghasilkan skop hasil *boolean*:

| **Jenis**  **Operator** | **Simbol**  **Operator** | **Arti**  **Operator** | **Contoh**  **pernyataan/ ekspresi** | **Hasil** |
| --- | --- | --- | --- | --- |
| Relasional | = | sama dengan | ‘a’ = ‘b’ | False |
|  | > | lebih besar | ‘b’ > ‘a’ | True |
|  | ≥ | lebih besar atau sama dengan | ‘b’ ≥ ‘a’ | True |
|  | < | lebih kecil | ‘b’ < ‘a’ | False |
|  | ≤ | lebih kecil atau sama dengan | ‘b’ ≤ ‘a’ | False |
|  | ≠ | tidak sama dengan | ‘b’ ≠ ‘a’ | True |

Pendefinisian teks algoritma tipe dan variabel dengan contoh :

Type NamaTipe = **Character**

Huruf = **Character**

Variable NamaVar : NamaTipe

H : Huruf

Contoh akses algoritma

Input(H)

H 🡨 ’X’

Output(H)

### Tipe Boolean

Boolean merupakan tipe yang memiliki skop logika dengan hanya dua nilai saja yaitu True atau False. Nilai *true* jika ekspresi tersebut bernilai benar dan *false* jika salah.

Penjabaran tipe :

Konvensi konstanta tipe ini adalah Konstanta C = *False* atau C = *True*

Operator yang digunakan sehingga menghasilkan skop hasil *boolean*:

| **Jenis**  **Operator** | **Simbol**  **Operator** | **Arti**  **Operator** | **Contoh**  **pernyataan/ ekspresi** | **Hasil** |
| --- | --- | --- | --- | --- |
| Logika | and | dan | True and True  True and False  False and True  False and False | True  False  False  False |
|  | or | atau | True or True  True or False  False and True  False or False | True  True  True  False |
|  | xor | ekslusif or | True xor True  True xor False  False xor True  False xor False | False  True  True  False |
|  | not | negasi | not True  nat False | False  True |
| Relasional | = | sama dengan | ‘a’ = ‘b’, 2 =1 | False |
|  | > | lebih besar | ‘b’ > ‘a’, 2 > 1 | True |
|  | ≥ | lebih besar atau sama dengan | ‘b’ ≥ ‘a’, 2 ≥ 2 | True |
|  | < | lebih kecil | ‘b’ < ‘a’, 2 < 1 | False |
|  | ≤ | lebih kecil atau sama dengan | ‘b’ ≤ ‘a’, 2 ≤ 1 | False |
|  | ≠ | tidak sama dengan | ‘b’ ≠ ‘a’, 2 ≠ 1 | True |
| Contoh  Fungsi standar | odd(x) | ganjil | odd(10)  odd(5) | True  False |
|  | eof(x) | end of file | eof(f) ? | True/False |
|  | eoln(x) | end of line | eoln(f) ? | True/False |

Kompleksitas logika akan meningkat jika ekspresi semakin panjang dan bervariasi.

Misalkan X, Y : boolean, dan X ← True, Y← False, maka

* (X AND Y) bernilai False,
* (X OR Y) bernilai True,
* dan NOT (X) bernilai False,
* NOT (Y) bernilai True,
* NOT (X AND Y) bernilai True,
* NOT (X OR Y) bernilai False,
* ( (X AND Y) OR (X OR Y) ) bernilai True
* dan seterusnya

Pendefinisian teks algoritma tipe dan variabel dengan contoh :

Type NamaTipe = **Boolean**

Benar = **Boolean**

Variable NamaVar : NamaTipe

B : Benar

Contoh akses algoritma

~~Input(B): tidak digunakan instruksi ini~~

B 🡨 FALSE

Output(B)

### Tipe String

String merupakan tipe yang memiliki skop kumpulan satu atau lebih karakter dengan panjang maksimum 255 atau kata kosong. Tipe ini ditandai dengan adanya tanda kutip sebagaimana *character*. Contoh tipe ini adalah ‘Anak’, ‘Apa%$@saja/kl’, ‘Oke007’, ‘AYAH’, dan lain-lain.

Dewasa ini string hampir semua bahasa pemrograman mendefinisikan string sebagai paket tabel (*array*) dari karakter, sehingg setiap karakter dalam sebuah string dapat diidentifikasi berdasar indeks dari string tersebut dimulai dengan indeks 1 untuk huruf pertama. Definisi tesebut dtuliskan sebagai berikut :

*String = packed array [1..N] of character,* dimana N adalah jumlah karakter dalam string.

Misalkan X = string[10] berarti himpunan kosong atau kumpulan satu atau lebih karakter dengan panjang maksimum kumpulan adalah 10. Misal X diisi dengan string ’APA KABAR?’, maka ilustrasi X dapat disajikan sebagai berikut.

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| X ==> | A | P | A |  | K | A | B | A | R | ? |
| indeks | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |

Nilai X dapat diidentifikasi dengan X = ’APA KABAR?’

Nilai X[1]=‘A’, X[2]=‘P’, X[3]=‘A’, X[4]=‘ ’,

X[5]=‘K, X[6]=‘A’, X[7]=‘B’, X[8]=‘A’, X[9]=‘R’, X[10]=‘?’

Penjabaran tipe :

Konvensi konstanta tipe ini adalah Konstanta C =’HALLO FRIEND’

Operator yang digunakan menghasilkan skop hasil *string, boolean,* atau *integer*:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Jenis**  **Operator** | **Simbol**  **Operator** | **Arti**  **Operator** | **Contoh**  **pernyataan/ ekspresi** | **Hasil** |
| Relasional | = | sama dengan | ‘apa’ = ‘APA’ | False |
|  | ≠ | tidak sama dengan | ‘apa’ ≠ ‘APA’ | True |
| Contoh  Fungsi standar | ⊕ | Concatenate/  Sambungkan | ‘apa’ ⊕ ‘KABAR’ | ‘apa KABAR’ |
|  | CopyString  (x,n,k) | copy string dari x mulai dari indeks ke-n sampai ke-k | CopyString(‘AKU MAU PULANG’, 5,3) | ‘MAU’ |
|  | trim(x) | buang spasi | trim(‘ I loveYou ’) | ‘IloveYou’ |
|  | pos(x,y) | posisi substring x relatif terhadap y | pos(‘Y’, I Love You’) | 8 |
|  | length(x) | panjang string | length(‘I Love You’) | 10 |

Pendefinisian teks algoritma tipe dan variabel dengan contoh :

Type NamaTipe = **String[N]**

Nama = **String[25]**

Variable NamaVar : NamaTipe

Nm : Nama

Contoh akses algoritma

Input(Nm)

Nm 🡨 ’Robert Sanyata’

Output(Nm)

### Tipe Enumerasi

Enumerasi (*Enumeration*) merupakan tipe data skalar yang tidak terstruktur atau yang ditentukan sendiri oleh pemakai (*programmer*) dengan cara mendaftar satu per satu nilainya secara berurutan. Dengan kata lain, tipe data enumurasi berupa deretan daftar nilai/nama terbatas yang didefinisikan pemrogram. Sebenarnya dasar pembuatan tipe skalar adalah melakukan enumerasi. Contoh-contoh tipe data enumerasi disajikan sebagai berikut.

1. Kelamin = enumerasi (Laki\_laki, Perempuan)
2. Hari = enumerasi (senin, selasa, rabu, kamis, jumat, sabtu, minggu)
3. Warna1= enumerasi (merah, jingga, kuning, hijau, biru, nila, ungu)
4. Warna2 = enumerasi (kuning, merah, hijau, biru, coklat)
5. Sayuran = enumurasi (kangkung, bayam, wortel, kol, kemangi, brokoli)
6. Boolean = enumerasi (false, true)
7. character = enumerasi (’a’, ’b’, ...), seluruh karakter didaftar menurut urutan tertentu. Urutan standar adalah mengikuti urutan dalam kode ASCII.
8. operator = enumerasi (plus, minus, kali, bagi)

Penjabaran tipe :

Konvensi konstanta tipe ini adalah Konstanta C = senin atau C = false

Operator yang digunakan dengan contoh data tersebut di atas sehingga menghasilkan skop hasil *character* adalah:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Jenis**  **Operator** | **Simbol**  **Operator** | **Arti**  **Operator** | **Contoh**  **pernyataan/ ekspresi** | **Hasil** |
| Contoh  Fungsi standar | succ (x) | Successor | succ(senin) | selasa |
|  | pred(x) | Predecessor | pred(selasa) | senin |
|  | ord(x) | Ordinat | ord(rabu) | 2 (senin=0) |

Operator yang digunakan sehingga menghasilkan skop hasil *boolean*:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Jenis**  **Operator** | **Simbol**  **Operator** | **Arti**  **Operator** | **Contoh**  **pernyataan/ ekspresi** | **Hasil** |
| Relasional | = | sama dengan | senin = Selasa | False |
|  | > | lebih besar | selasa > senin | True |
|  | ≥ | lebih besar atau sama dengan | selasa ≥ senin | True |
|  | < | lebih kecil | selasa < senin | False |
|  | ≤ | lebih kecil atau sama dengan | selasa ≤ senin | False |
|  | ≠ | tidak sama dengan | selasa ≠ senin | True |

Pendefinisian teks algoritma tipe dan variabel dengan contoh :

Type NamaTipe= **Enumeration(E1, E2, E3, ..., En)**

Warna = **Enumeration**(kuning, merah, hijau, biru, coklat)

Variable NamaVar : NamaTipe

W : Warna

Contoh akses algoritma

~~Input(W): tidak digunakan instruksi ini~~

W 🡨 Kuning

~~Output(W): tidak digunakan instruksi ini~~

### Tipe Subrange

Subrange merupakan merupakan tipe dengan skop data yang merupakan bagian (*subrange*) dari tipe data skalar standar *integer*, *character,* atauenumerasi yang dinyatakan nilainya dari terkecil sampai terbesarnya. Nilai terkecil dengan nilai terbesar dipisahkan dengan tanda ”..”. Contoh-contoh tipe data enumerasi disajikan sebagai berikut.

1. Nilai\_Matakuliah = 0..100
2. Hari\_kerja = senin .. jumat {dengan catatan enumerasi (senin, selasa, rabu, kamis, jumat, sabtu, minggu) terdefinisi sebelumnya }
3. Data\_kedalaman= -100..100
4. Tanggal = 1..31
5. Bulan = 1..12
6. Tahun = 1900..2050

Penjabaran tipe :

Konvensi konstanta tipe ini adalah Konstanta C = senin atau C = 10

Tipe subrange memiliki operator seperti tipe data dengan skop diatasnya yaitu integer atau character. Operator yang digunakan sehingga menghasilkan skop hasil *subrange* *integer* atau *boolean* adalah :

| **Jenis**  **Operator** | **Simbol**  **Operator** | **Arti**  **Operator** | **Contoh**  **pernyataan/ ekspresi** | **Hasil** |
| --- | --- | --- | --- | --- |
| Aritmetika | \* | kali | 3 \* 5 | 15 |
|  | div | bagi & potong pecahan | 5 div 2  5 div 3 | 2  1 |
|  | mod | sisa pembagian | 5 mod 2  5 mod 3 | 1  2 |
|  | ^ | pangkat | 5 ^ 3 | 125 |
|  | + | tambah | 5 + 3 | 8 |
|  | - | kurang | 5 – 3 | 2 |
| Relasional | = | sama dengan | senin = Selasa | False |
|  | > | lebih besar | selasa > senin | True |
|  | ≥ | lebih besar atau sama dengan | selasa ≥ senin | True |
|  | < | lebih kecil | selasa < senin | False |
|  | ≤ | lebih kecil atau sama dengan | selasa ≤ senin | False |
|  | ≠ | tidak sama dengan | selasa ≠ senin | True |

Pendefinisian teks algoritma tipe dan variabel dengan contoh :

Type NamaTipe= **S1..S2** (subrange-1 s/d subrange-2)

Nilai = **1..100**

Variable NamaVar : NamaTipe

N : Nilai

Contoh akses algoritma

Input(N)

N 🡨 123

Output(N)

### Tipe Set

Set merupakan data yang berbentuk himpunan dan merupakan bagian dari tipe data skalar, sehingga dituliskan sebagai *set of* tipe skalar. Untuk menyatakan keanggotaan himpunan digunakan antara dua tanda kurung siku ”[ ... ]” .

Contoh-contoh tipe data ini disajikan sebagai berikut.

1. DataSet1 = Set of character
2. DataSet2 = Set of integer
3. DataSet3 = Set of boolean
4. DataSet4 = Set of Warna, dengan ketentuan enumerasi Warna= enumerasi (merah, jingga, kuning, hijau, biru, nila, ungu) sudah terdefinisi sebelumnya.
5. Huruf\_Kecil = Set Of ‘a’..’z’
6. Huruf\_Besar = Set Of ‘A’..’Z’
7. Digit = Set Of ’0’..’9’

Contoh untuk menyatakan himpunan adalah :

1. Himpunan\_Kosong = [ ]
2. Himpunan\_HurufKecil = [’a’, ’b’, ’x’,’z’]
3. Himpunan\_HurufBesar = [’A’, ’P’, ’Q’,’R’]
4. Himpunan\_Digit = [’5’, ’6’, ’7’]

Penjabaran tipe :

Konvensi konstanta tipe ini adalah Konstanta C = [senin, minggu] atau C = [ ]

Operator yang digunakan sehingga menghasilkan skop hasil *set* atau *boolean*:

| **Jenis**  **Operator** | **Simbol**  **Operator** | **Arti**  **Operator** | **Contoh**  **pernyataan/ ekspresi** | **Hasil** |
| --- | --- | --- | --- | --- |
| Himpunan | + | *union* | [‘a’,’b’] + [‘1’,’2’] | [‘a’,’b’,‘1’,’2’] |
|  | \* | *intersection* | [‘a’,’z’] + [‘z’,’2’] | [‘z’] |
|  | - | set difference | [‘a’,’z’] - [‘z’,’2’] | [‘a’] |
|  | in | anggota | ‘ x’ in [‘a’,’b’,‘1’,’2’] | False |
| Relasional | = | sama dengan | [‘a’,’b’] = [‘1’,’2’] | False |
|  | > | lebih besar | [‘x’,’z’] > [‘a’,’b’] | True |
|  | ≥ | lebih besar atau sama dengan | [‘x’,’z’] ≥ [‘a’,’b’] | True |
|  | < | lebih kecil | [‘x’,’z’]< [‘a’,’b’] | False |
|  | ≤ | lebih kecil atau sama dengan | [‘x’,’z’] ≤ [‘a’,’b’] | False |
|  | ≠ | tidak sama dengan | [‘x’,’z’] ≠ [‘a’,’b’] | True |

Pendefinisian teks algoritma tipe dan variabel dengan contoh :

Type NamaTipe= **Set Of ...**

Letter = **Set Of ’A’..’Z’**

Variable NamaVar : NamaTipe

Ltr : Letter

Contoh akses algoritma

~~Input(Ltr) : tidak digunakan instruksi ini~~

Ltr 🡨 [’A’,’G’,’H’] union [’X’]

Output(Ltr)

### Tipe Pointer

Pointer merupakan tipe data yang bersifat dinamik dan memiliki skop data berupa penunjukkan ke sebuah alamat (*address*) memori tertentu. Tipe dengan sifat dinamik artinya pembentukan atau alokasi data di dalam memori dilakukan saat eksekusi program. Tipe alamat disimbolkan ”@Tipe” atau ”Pointer To Tipe”. Penunjukkan isi alamat memori disimbolkan dengan ” ↑” diikuti dengan titik ”.” (dot) menjadi ” ↑**.**”

Tipe data ini akan menjadi kompleks jika struktur data juga menjadi kompleks. Contoh-contoh sederhana dari tipe data ini disajikan sebagai berikut.

1. PointerInteger = Pointer to Integer
2. PointerReal = Pointer to Real
3. PointerCharacter = Pointer to Character
4. PointerBoolean = Pointer to Boolean
5. PointerString = Pointer to String

Ilustrasi pointer

|  |  |  |
| --- | --- | --- |
| **RAM** | | Berdasarkan RAM di samping ini.  Misal P bertipe Pointer to character dan ada intruksi Alokasi(P)  P 🡨 3  Output(P) // *hasilnya 3*  Output(P↑.) // *hasilnya ’H’* |
| **Alamat** | **Isi RAM** |
| 1 | ’A’ |
| 2 | ’X’ |
| 3 | ’H’ |
| 4 | ’B’ |
| : |  |
| N | ’Z’ |

Penjabaran tipe :

Konvensi konstanta tipe ini adalah Konstanta C = null atau nil

Operator yang digunakan sehingga menghasilkan skop hasil *pointer*:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Jenis**  **Operator** | **Simbol**  **Operator** | **Arti**  **Operator** | **Contoh**  **pernyataan/ ekspresi** | **Hasil** |
| Contoh  Fungsi standar | allocate (x) | Alokasi pointer baru | allocate(p) | p@ |
| dispose(x) | Menghapus alokasi pointer | dispose(p) | nil |

Pendefinisian teks algoritma tipe dan variabel dengan contoh :

Type NamaTipe = **Pointer to ... atau @...**

pointerInt = **Pointer to Integer atau @Integer**

Variable NamaVar : NamaTipe

P : PointerInt

Contoh akses algoritma

Allocate(P)

P↑ 🡨 123 // atau P↑ 🡨 Null

Output(P, P↑.)

## Identitas dan Ekspresi Data

### Identitas Data

Identitas data direpresentasikan dengan pemberian nama data yang digunakan untuk acuan dan pemrosesan data. Nama data juga disebut *indentifier* data terdiri dari dua kategori yaitu kategori sebagai acuan yang diwakili dengan nama tipe dan kategori yang digunakan untuk keperluan proses diwakili dengan nama konstanta, variabel, dan parameter.

Aturan penamaan data (*identifier*) adalah :

* Minimal satu karakter dan harus dimulai dengan huruf (selanjutnya boleh kombinasi huruf & angka)
* Nama tidak boleh ada spasi (biasanya spasi diganti dengan tanda *underscore* “\_” sehingga tetap menyambung sebagai satu kata)
* Tidak boleh ada karakter-karakter yang merupakan tanda operator matematika ( +, -, \*, &, %, ^) dan tidak boleh ada karakter-karakter lain yang berarti khusus (!, ?, # . ~, ”,”,(,{,[, ],},), dan lain-lain)

Untuk mempermudah memahami dan mendefinisikan segala hal yang berkaitan penamaan maupun istilah dalam dunia pemrograman komputer digunakan alat bantu yang disebut diagram sintaks. Smbol-simbol yang digunakan dalam diagram sintaks dapat dilihat dalam tabel berikut ini.

Simbol diagram sintaks

|  |  |  |
| --- | --- | --- |
| **Jenis Simbol** | **Simbol** | **Keterangan** |
| Terminal |  | Simbol yang tidak menurunkan simbol baru |
| Non-Terminal | NT | Simbol yang menurunkan simbol baru |
| Himpunan Terminal | H | Simbol yang terdiri dari beberapa anggota terminal atau nilai tertentu |

*Identifier* atau penamaan data dinyatakan dalam aturan yang digambarkan dalam diagram berikut ini.

*Identifier*

Letter

Letter

Digit

*Letter* **:**

*Digit* **:**

Macam-macam dan perbedaaan *identifier* data tersebut dijelaskan sebagai berikut :

1. **Tipe (Type**) merupakan nama skop atau jenis data tertentu yang digunakan sebagai acuan (referensi) untuk pendefinisian variabel dan parameter. Nama sebuah tipe tidak boleh (tidak bisa) diproses atau dioperasikan dalam ekspresi perhitungan aritmetika maupun logika. Penjelasan tentang tipe data sudah diuraikan sebelumnya, sedangkan contoh penulisan tipe dilakukan dengan format :

Type Polinom = Real

Nama = String

Bilangan = Integer

1. **Konstanta (Constanta)** merupakan nama untuk sebuah tetapan yang nilainya tidak boleh dan tidak pernah berubah selama proses berlangsung dari awal sampai selesai. Contoh penulisan konstanta dilakukan dengan format :

Constanta Phi = 3.14 (untuk komputasi lingkaran)

g = 9.8 (untuk komputasi gravitasi bumi)

C = 300.000 (dalam km/detik untuk komputasi tetapan cahaya)

1. **Variabel (Variable**) merupakan nama data yang digunakan untuk proses (komputasi, kalkulasi, maupun konversi ) sehingga nilai-nilainya dapat berubah-ubah selama proses belum selesai. Setiap nama variabel harus memiliki tipe data. Umumnya variabel untuk tipe data sederhana dapat didefinisikan secara langsung bersama tipenya, sedangkan untuk tipe data kompleks didefinisikan tidak langsung (menggunakan nama acuan tipe). Contoh penulisan variabel dilakukan dengan format:

Type DataInteger = Integer;

Mahasiswa = <NIM : string; NAMA : String; NILAI : Integer>

Variable X : DataInteger // Acuan tidak langsung didefinisikan pemakai

Y : Integer // Acuan langsung tipe sederhana

Mhs : Mahasiswa // Acuan tidak langsung didefinisikan pemakai

1. Parameter merupakan nama data yang digunakan sebagai variabel argumen *input* dan atau *output*  yang akan diproses pada pembentukan prosedur atau fungsi. Penjelasan lebih detil tentang parameter umumnya jelaskan dalam pembahasan prosedur dan fungsi. Contoh penulisan parameter dilakukan dengan format:

Type Infoparameter = Integer

Info\_hasilfungsi = string

Procedure TukarNilai(in/out A, B : Infoparameter)

{Status Inisial : A, B parameter integer bernilai sembarang}

{Status Akhir : Nilai A bertukar dengan nilai B }

Fuction integerToStr(in X : Infoparameter)→ Info\_hasilfungsi

{diberikan X parameter integer bernilai sembarang}

{dikirimkan hasilkonversi integer X menjadi string ‘X’ }

### Ekspresi Data

Suatu data yang akan diproses dinyatakan dengan sebuah ekspresi, baik ekspresi aritmetika maupun ekspresi logika. Untuk ekspresi yang cenderung aritmetika, contoh format penulisan ekspresi data tersebut dengan menggunakan definisi dan beberapa diagram sintaks berturut-turut mulai dari pengertian faktor, *term*, dan ekspresi diuraiakan berikut ini.

* **Faktor** adalah suatu nilai yang terkandung dalam bentuk variabel, *number*, dan ekspresi yang dinyatakan dengan diagram sintaks berikut.

Variabel

Number

Ekspresi

*Faktor*

***Veriabel*****:** identifier yang dibuat yang memiliki nilai tertentu

***Number :*** simboldigitbilangan yang mewakili nilai tertentu

Contoh : Id 🡨 Nilai Data (berbentuk Faktor)

Faktor Variabel : X, Y, Jumlah, dan seterusnya (Id 🡨 X)

Faktor Number : 125, 50, 99, dan terusnya (Id 🡨 125)

Faktor Ekspresi : (125+X\*10/2), (X+Y\*A-10), dan seterusnya (Id 🡨 (125+X\*10/2))

* **Term** adalah suatu nilai yang terkandung dalam faktor atau ekspresi antar faktor yang menggunakan operator *term* yang dinyatakan dengan diagram sintaks berikut.

Operator\_

Term

Faktor

Faktor

*Term*

*Operator\_Term* **:**

Contoh : Id 🡨 Nilai Data (berbentuk Term)

Term Faktor : X, Y, 125, 50, 99, (125+X\*10/2), dan terusnya (Id🡨 X)

Term Ekspresi : X \* Y, X / 50, 100 div 2, 5 mod 3,

(125+X\*10/2) / X, dan terusnya ( Id🡨(125+X\*10/2) / X )

* **Ekspresi (expression)** adalah suatu peryataan yang memiliki nilai tertentu dalam skop tipe data tertentu pula. Penulisan ekspresi dinyatakan dengan diagram sintaks berikut.

Term

Term

*Ekspresi \_*

*Sederhana*

Ekspresi\_

Sederhana

*Ekspresi*

*kompleks*

Operator\_

Relasional

Ekspresi\_

Sederhana

*Operator\_Relasional* **:**

Contoh : Id 🡨 Nilai Data (berbentuk Ekspresi)

Ekpsresi sederhana : X, Y, 125, 50, 99, (125+X\*10/2), X \* Y, X / 50, 100 div 2,

5 mod 3, (125+X\*10/2) / X, dan terusnya

Ekspresi kompleks : (125+X\*10/2) > (X / 50), dan terusnya

Contoh : Id 🡨 (125+X\*10/2) > (X / 50) //Id: Boolean

# SKEMA ALGORITMA MODULAR TIPE NON-REKURSIF

## Skema Algoritma Modular

Skema algoritma program modular adalah skema algoritma membuat program utuh yang dibangun dengan teknik menggunakan sekumpulan skema subprogram yang saling berhubungan satu dengan subprogram lain. Teknik ini yang bertujuan memberikan menyederhanakan kompleksitas program yang besar, meningkatkan keterbacaan (Readability) program, dan memberikan kemudahan memodifikasi (modifiability) program secara utuh dan terintegrasi.

Subprogram adalah bagian program yang memiliki nama dan karakteristik proses atau fungsi spesifik yang dimanfaatkan sebagai satu kesatuan lojik dari program secara keseluruhan dengan cara pemanggilan subprogram (subprogram call) oleh program utama maupun subprogram lain. Skema algoritma subprogram dalam hal ini akan disebut sebagai skema algoritma prosedur (skema **procedure)** atau fungsi ( skema **function)**. Skema untuk mengelompokkan sekumpulan subprogram (prosedur dan atau fungsi) yang sejenis disebut **modul**. Jadi skema algoritma secara keseluruhan dapat menjadi empat jenis, meliputi :

1. Skema **Program**, untuk membuat program dalam pengertian yang lengkap dan utuh.
2. Skema **Modul**, untuk membuat pustaka bagi kumpulan subprogram sejenis
3. Skema **Procedure**, untuk mendefinisikan elemen subprogram sebagai sebuah proses
4. Skema **Function**, untuk mendefinisikan elemen subprogram sebagai sebuah fungsi

Struktur Program Modular

**Skema** **Program Utama**

**ALGORITMA**

**: CALL** Subprogram-1

**: CALL** Suprogram-11

**:**

**End.**

Link-1

Link-N

**Link** : Modul-1…Modul-N

Subprogram **utama (internal)**

Subprogram-1

Subprogram-N

Subprogram-2

**Modul-1 :**

Subprogram-11

Subprogram-1N

Subprogram-12

**Modul-N**

Subprogram-N1

Subprogram-NN

Subprogram-N2

File Program Utama

File Modul-1 (mode eksternal)

File Modul-N (mode eksternal)

* 1. Diagram skema algoritma modular

Struktur program modular diperlihatkan pada gambar di atas. Berdasar lokasi keberadaan subprogram relatif terhadap program utama (induknya), maka subprogram dikelompokkan dalam dua mode yaitu :

1. Subprogram mode eksternal, merupakan paket modul kumpulan subprogram yang memiliki karakteristik proses/fungsi yang sejenis dan diletakkan sebagai *file* terpisah dari *file* program induk (program utama). Modul disebut jiga UNIT atau MODUL atau LIBRARY.

Karena modul merupakan bagian dari program induknya maka agar subprogram tersebut bisa di-*call*, unit/modul tersebut harus di-*link* terlebih dulu. Penggunaan modul dari program pemanggilnya didefinisikan dengan membuat *link* dengan notasi :

**LINK** Modul-1, Modul-2, dst.

Contoh untuk *link* dalam bahasa pascal menggunakan kata dicadangkan (reserved word) USES, dan untuk bahasa C menggunakan kata INCLUDE.

1. Subprogram mode internal, merupakan subprogram yang berupa PROCEDURE (SUBRUTIN) dan FUNCTION, yang diletakkan dalam *file* yang sama dengan program induknya.

Pemanfaatan prosedur dilakukan dengan melakukan pemanggilan prosedur (procedure call), sedangkan pemanfaatan fungsi dilakukan dengan mengganggap fungsi sebagai sebuah nilai (value) dengan pemanggilan fungsi (function call).

Contoh : **Skema Program Utama Teknik Modular:**

Judul : **Program** Utama;

// dokumentasi program utama

Kamus :

// definisi Link

LINK modul-1, modul-2, ...

// Deklarasi definisi Constanta, Type, Variable global

...

// Deklarasi definisi subprogram utama

**Procedure** P1 (Parameter\_Formal);

**Procedure** P2 (Parameter\_Formal);

...

**Procedure** PN (Parameter\_Formal);

**Function** F1 (Parameter\_Formal)🡪 tipe\_hasil\_F1;

**Function** F2 (Parameter\_Formal)🡪 tipe\_hasil\_F2;

...

**Function** FM (Parameter\_Formal)🡪 tipe\_hasil\_FM;

//Algoritma utama

**Algoritma**;

...

CALL P1 (Parameter\_Aktual)

...

CALL F1 (Parameter\_Aktual)

...

...

end.

// Realisasi Skema Subprogram utama

**Procedure** P1(Parameter\_Formal);

{IS. : uraikan kondisi awal parameter formal yang diperbolehkan }

{IF. : uraikan kondisi hasil akhir tentang parameter formal }

Kamus lokal :

// berisi varibel lokal untuk proses dalam prosedur

...

Algoritma

// algoritma proses untuk mencapai kondisi FS. dari kondisi IS.

...

End;

*... dan setrusnya untuk prosedur lainnya*

**Function** F1 (Parameter\_Formal)🡪 tipe\_hasil\_F1;

{diberikan kondisi awal parameter formal yang diperbolehkan }

{dikirimkan *return\_value* hasil akhir perhitungan fungsi F1 }

Kamus lokal :

// berisi varibel lokal untuk fungsi

...

Algoritma

// algoritma proses komputasi F1 .

...

🡪(return\_value)

End;

*... dan setrusnya untuk Fungsi lainnya*

------------------------------------------------------

Contoh : **Skema Modul:**

Judul : **Modul** Modul-1;

// dokumentasi modul-1

Kamus :

// definisi Link

LINK ...

// Deklarasi definisi Constanta, Type, Variable Modul

...

// Deklarasi definisi subprogram modul-1

**Procedure** P11 (Parameter\_Formal);

**Procedure** P12 (Parameter\_Formal);

...

**Function** F11 (Parameter\_Formal)🡪 tipe\_hasil\_F11;

**Function** F12 (Parameter\_Formal)🡪 tipe\_hasil\_F12;

...

**Algoritma**;

...

end.

//Realisasi detil subprogram modul-1

**Procedure** P11(Parameter\_Formal);

{IS. : uraikan kondisi awal parameter formal yang diperbolehkan }

{IF. : uraikan kondisi hasil akhir tentang parameter formal }

... dan seterusnya sebagaimana deskripsi prosedur seperti subprogram utama

**Function** F1M (Parameter\_Formal)🡪 tipe\_hasil\_F11;

{diberikan kondisi awal parameter formal yang diperbolehkan }

{dikirimkan *return\_value* hasil akhir perhitungan fungsi F1M }

... dan seterusnya sebagaimana deskripsi fungsi seperti subprogram utama

## Skema Prosedur

**Prosedur** adalah subprogram yang dirancang untuk melakukan proses subtugas tertentu dari keseluruhan program berdasarkan parameter dan variabel yang ditetapkan apabila dipanggil (di-*call)*. Prosedur boleh tidak memakai parameter (parameter kosong). Karena secara konseptual prosedur merupakan bagian dari program maka prosedur memerlukan kerangka acuan proses yang berupa kondisi awal parameter sebelum diproses (Initial State disingkat I.S.) dan kondisi parameter hasil proses yang menyatakan keadaan akhir setelah prosedur diproses (Final State yang disingkat F.S.). IS dan FS merupakan “kontrak tugas” yang harus dijamin tercapai oleh algoritma prosedur tersebut.

Nama Prosedur (Parameter

Formal)

Definisi

deskriptif

Prosedur

Isi skema

algoritma

prosedur

Procedure **PX** (In F11,…,F1K; Out F21,…,F2M; In/Out F31,…,F3N)

{I.S. : ………………………………………………………….….}

{F.S. : …………………………………………………………….}

Realisasi algoritma prosedur, memproses I.S. menjadi F.S

Algoritma

proses prosedur dimulai dengan inisiasi CALL

transfer nilai paramater Aktual ke parameter Formal

F11 🡨 A11, … F1K 🡨 A1K

F21 🡨 A21, … F2M 🡨 A1M

F31 🡨 A31, … F2N 🡨 A1N

. . . {proses)

proses diakhiri dengan mengembalikan nilai paramater

Formal ke Aktual sesuai dengan karakteristik

definisi passing parameter IN, OUT, atau IN/OUT

end;

Parameter Aktual

CALL **PX** (A11,…,A1K, A21,…,A2M, A31,…,F3N)

* 1. Diagram skema algoritma procedure

1. **Deklarasi formal skema algoritma prosedur**

Cara mendeklarasikan atau mendefinisikan secara formal skema algoritma prosedur dilakukan dengan mendaftar nama prosedur dan merealisasikan instruksi algoritma ke dalam prosedur tersebut, seperti diilustrasikan dalam gambar. Sebuah prosedur dideklarasikan cukup dengan menuliskan nama prosedur beserta parameternya dilengkapi dengan deskripsi status awal atau *Initial State* (I.S) dan status akhir atau *Final State* (F.S). Sedangkan untuk merealisasikan algoritma lengkap dilakukan dengan menulis lengkap instruksi algoritmanya secara detil. **Catatan** bahwa deklarasi dan realisasi prosedur dapat dilakukan sebagai satu kesatuan (bersamaan) sebagai subprogram internal atau secara terpisah eksternal sebagai deklarasi sebagai interface dan realisasi sebagai implementasi .

Parameter formal memiliki tiga tipe, yaitu tipe **In** (input) yang menyatakan bahwa parameter tersebut akan diproses hanya sebagai input dan nilainya tidak berubah setelah akhir proses. Tipe **Out** (Output) menyatakan bahwa parameter tersebut digunakan untuk menaruh hasil proses tanpa menghiraukan adanya input sebelumnya. Tipe **In/Out** (Input-Output) menyatakan parameter yang memiliki nilai input tertentu dan sekaligus digunakan untuk menaruh hasil akhir proses.

Format deklarasi nama prosedur:

**Procedure** Nama\_prosedur (Parameter\_Formal);

{IS. : uraikan kondisi awal parameter formal yang diperbolehkan }

{IF. : uraikan kondisi hasil akhir tentang parameter formal }

Format deklarasi realisasi/implementasi skema prosedur :

**Procedure** Nama\_prosedur (Parameter\_Formal);

{IS. : uraikan kondisi awal parameter formal yang diperbolehkan }

{IF. : uraikan kondisi hasil akhir tentang parameter formal }

Kamus lokal :

: // berisi varibel lokal untuk proses dalam prosedur

:

Algoritma

: // algoritma proses untuk mencapai kondisi FS. dari kondisi IS.

:

End;

Contoh deklarasi :

**Procedure** TukarNilai(in/out A, B : Integer);

{IS. : A, B sembarang integer a dan b }

{IS. : A berisi nilai b dan B berisi nilai a }

atau

**Procedure** TukarNilai(in A : integer; Out B : Integer);

{IS. : A, B sembarang integer }

{IS. : B berisi nilai A }

atau

**Procedure** TukarNilai(Out A : integer; In B : Integer);

{IS. : A, B sembarang integer }

{IS. : A berisi nilai B }

Contoh Realisasi:

**Procedure** TukarNilai(in/out A, B : Integer);

{IS. : A, B sembarang integer a dan b }

{IS. : A berisi nilai b dan B berisi nilai a }

Kamus lokal :

Temp : Integer

Algoritma

Temp 🡨 A

A 🡨 B

B 🡨 Temp

End;

1. **Pemanggilan prosedur (Procedure Calling)**

Pelaksanaan prosedur dilakukan dengan cara pemanggilan prosedur tersebut oleh prosedur/program pemanggil. Cara pemanggilan dilakukan dengan menyebut **Call** diikuti nama prosedur yang dipanggil dilengkapi parameter aktualnya. Parameter aktual adalah parameter yang disebutkan pada pemanggilan prosedur. Berkaitan dengan proses adanya komunikasi data antara parameter aktual dengan parameter formal, pemanggilan prosedur menggenal tiga jenis Call yang diuraikan pada subbagian pelewatan parameter (parameter passing), terdiri dari: Call By Reference, Call By Copy, dan Call By Name.

Format pemanggilan prosedur :

**Call** Nama\_Prosedur (Parameter\_Aktual);

Contoh :

Berdasarkan deklarasi prosedur TukarNilai yang telah dijelaskan sebelumnya maka contoh pemanggilan prosedur tersebut adalah :

**Procedure** TukarNilai(in/out A, B : Integer);

{IS. : A, B sembarang integer a dan b }

{IS. : A berisi nilai b dan B berisi nilai a }

**var** X,Y : Integer

X 🡨 50

Y 🡨 100

**Call** TukarNilai(X,Y); // contoh Call By Reference

atau

**Call** TukarNilai(10,100); // contoh Call By Value (Copy)

atau

**Call** TukarNilai(X,100); // contoh by Reference dan By Value

## Skema Fungsi

**Fungsi** adalah subprogram yang dirancang untuk melakukan komputasi (aritmetika dan logika) dari suatu ekspresi atau pernyataan tertentu sehingga menghasilkan **satu buah nilai** dari tipe data yang ditetapkan.

1. **Deklarasi formal skema algoritma fungsi:**

Cara mendeklarasikan atau mendefinisikan secara formal skema algoritma fungsi dilakukan dengan mendaftar nama fungsi dan merealisasikan instruksi algoritma ke dalam fungsi tersebut. Sebuah fungsi dideklarasikan sama seperti prosedur dengan cara cukup menuliskan nama fungsi beserta parameternya dilengkapi dengan deskripsi parameter yang menjadi *input* dan tipe data yang dikirimkan sebagai hasil fungsi (*Return Value*). Sedangkan cara merealisasikan algoritmanya adalah menulis lengkap instruksi algoritmanya secara detil. **Catatan** bahwa fungsi sebagai subprogram deklarasi dan realisasi fungsi dilakukan sebagai sesuatu yang sama dengan prosedur.

Parameter formal untuk fungsi hanya memiliki tipe **In** (input), sehingga dalam penulisanya tidak perlu disebut tetapi sudah dianggap default.

Format deklarasi nama Fungsi

**Function** Nama\_Fungsi (Parameter\_Formal)🡪 tipe\_data\_hasil;

{Diinput : batasan parameter formal }

{Dikirim : hasil fungsinya}

Format deklarasi realisasi skema fungsi :

**Function** Nama\_Fungsi (Parameter\_Formal)🡪 tipe\_data\_hasil;

{Diinput : batasan parameter formal }

{Dikirim : hasil fungsinya}

Kamus lokal :

: // berisi varibel lokal untuk proses

:

Algoritma

: // algoritma proses untuk komputasi

:

🡪 (nilai hasil komputasi/fungsi) //return value

End;

Contoh deklarasi :

**Function** Kuadrat(k1,k2,k3: Integer; X :Real)🡪 Real;

{Diinput: koefisien k1,k2,k3 sembarang integer, X sembarang real}

{Dikirim: k1\*X2+ k2\*X+ k3 }

**Function** IsEmpty(X : String) 🡪Boolean;

{Diinput: X string sembarang }

{Dikirim: true jika X=’’, false untuk lainnya}

**Function** AngkaToHari(X : Integer) 🡪String;

{Diinput: X integer 1..7 }

{Dikirim: Senin jika X=1, Selasa jika X=2, dst}

Contoh Realisasi:

**Function Kuadrat(k1,k2,k3: Integer; X :Real)🡪 Real;**

{Diinput: koefisien k1,k2,k3 sembarang integer, X sembarang real}

{Dikirim: k1\*X2+ k2\*X+ k3 }

Kamus lokal :

H : Real;

Algoritma

H 🡨 k1\*X2+ k2\*X+ k3

🡪(H)

End;

**Function IsEmpty(X : String) 🡪Boolean;**

{Diinput: X string sembarang }

{Dikirim: true jika X=’’, false untuk lainnya}

Kamus lokal : -

Algoritma

If (X=’’) then

🡪(True)

else

🡪(False)

endif

End;

**Function AngkaToHari(X : Integer) 🡪String;**

{Diinput: X integer 1..7 }

{Dikirim: Senin jika X=1, Selasa jika X=2, dst}

Kamus lokal :

H :String;

Algoritma

Depend on X

1 : H 🡨 ‘Senin’

2 : H 🡨 ‘Selasa’

3 : H 🡨 ‘Rabu’

4 : H 🡨 ‘Kamis’

5 : H 🡨 ‘Jumat’

6 : H 🡨 ‘Sabtu’

7 : H 🡨 ‘Minggu’

endDepend

🡪(H)

End;

1. **Pemanfaatan Fungsi (Function Using)**

Pemanfaatan atau pemanggilan fungsi tidak sama dengan pemanggilan prosedur. Karena hasil fungsi adalah sebuah nilai maka fungsi harus diperlakukan sebagai nilai (value) degan tipe data tertentu, sehingga fungsi tidak dapat berdiri sendiri melainkan harus diperlakukan sebagaimana data. Sebagai data, fungsi dapat diproses dengan instruksi *assignment* untuk nilai variabel atau output sebagai sebuah nilai.

Format pemanfaatan/pemanggilan prosedur :

**Variabel 🡨** Nama\_Fungsi(Parameter\_Aktual)

**Output**(Nama\_Fungsi(Parameter\_Aktual))

Contoh :

Berdasarkan deklarasi fungsi sebelumnya maka contoh pemanggilan fungsi dapat dilakukan sebagai berikut :

**Function** Kuadrat(k1,k2,k3: Integer; X :Real)🡪 Real;

{Diinput: koefisien k1,k2,k3 sembarang integer, X sembarang real}

{Dikirim: k1\*X2+ k2\*X+ k3 }

**var** X : Real; Y : String; Z : Boolean;

X 🡨 **kuadrat**(1,2,3,5.0) // X = 38

Output(X) // tercetak hasil 38

Output(**kuadrat**(1,2,3,5.0)) // tercetak hasil 38

**Function** AngkaToHari(X : Integer) 🡪String;

{Diinput: X integer 1..7 }

{Dikirim: Senin jika X=1, Selasa jika X=2, dst}

Y 🡨 **AngkaToHari**(3) // Y = ‘Rabu’

Output(Y) // tercetak hasil ‘Rabu’

Output(**AngkaToHari**(5)) // tercetak hasil ‘Jumat’

**Function** IsEmpty(X : String) 🡪Boolean;

{Diinput: X string sembarang }

{Dikirim: true jika X=’’, false untuk lainnya}

**Z 🡨 IsEmpty(‘’)** // Z = True

Output(Z) // Tercetak True

Output(IsEmpty(‘Sanyata’)) // Tercetak False

## Pelewatan Parameter (Passing Parameter)

**Parameter Passing** antara subprogram/program pemanggil dan subprogram yang dipanggil bertujuan untuk komunikasi data antar unit-unit atau subprogram-subprogram melalui pemanggilan subprogram (subprogram call) sehingga memberikan keuntungan berupa keterbacaan program (readability) dan kemudahan modifikasi/kesalahan program (modifiability). Parameter yang dilewatkan pada pemanggilan subprogram terbagi dalan tiga kelas yaitu data, subprogram, dan type.

**A. Parameter Binding**

Parameter passing umumnya menggunakan metode positional binding (positional metode for binding actual to formal parameter) dalam subprogram call. Metode ini menekankan pada data dan urutan atau posisi parameter aktual terikat secara posisi dengan data dan urutan parameter formal.

Misalkan diketahui subprogram SP dengan parameter formal F1, F2, F3, …, Fn sbb:

Subprogram SP(F1, F2, F3, …,Fn)

// misal jenis parameter F1 = IN, F2 = OUT, F3 = IN/OUT

:

:

EndSubProgram

Pemanggilan subprogram SP dengan parameter aktual A1,A2,A3,…,An dilakukan dengan notasi :

CALL SP(A1, A2, A3,…,An)

Terdapat dua efek komunikasi pertukaran data akibat adanya pemanggilan yaitu :

1. Saat awal pemanggilan : terjadi proses transfer pertukaran data dari parameter aktual ke paramtere formal sbb :

F1 🡨 A1, F2 🡨 A2, F3 🡨 A3, ... , Fn 🡨 An.

1. Saat akhir/selesai peoses prosedur : terjadi proses transfer pengembalian nilai data dari parameter formal ke paramtere aktual sesuai dengan definisi jenis parameter formalnya apakah tipe IN, OUT, IN/OUT dan jenis pemanggilannya (jenis Call).

* jika tipe IN maka tidak ada transfer pengembalian data.
* jika tipe OUT atau IN/OUT maka ada transfer pengembalian nilai yaitu data hasil proses akhir parameter formal dicopy ke parameter aktual. Misal tipe parameter aktual F1 = IN, F2 = OUT, F3=IN/OUT maka transfer pengembalian nilainya adalah :

A2 🡨 F2, A3 🡨 F3, dan A1 tidak ada transfer.

**B. Jenis Call**

Jenis pemanggilan subprogram atau jenis Call dibedakan dalam tiga jenis sebagai berikut :

1. **Call By Reference (By Sharing)**

Unit pemanggil melewatkan alamat (address) memori dari parameter aktual untuk diacu atau di-*share* oleh parameter formal untuk proses. Jadi, setelah proses selesai maka parameter aktual berubah nilainnya sesuai dengan proses perubahan yang terjadi pada parameter formal. **Catatan** : parameter formal sesuai dengan tipe **In/Out**.

Contoh :

**Procedure** TukarNilai(in/out A, B : Integer);

{IS. : A, B sembarang integer a dan b }

{IS. : A berisi nilai b dan B berisi nilai a }

Kamus lokal :

Temp : Integer

Algoritma

Temp 🡨 A

A 🡨 B

B 🡨 Temp

End;

Misal **var** X,Y:Integer, dan X 🡨 5, Y 🡨 7 maka Call By Reference,

**Call TukarNilai**(X,Y)terjadi penggunaan variabel di memory (RAM)

diilustrasikan sbb:

Memori (RAM) : Temp

m1?

variabel aktual: X Y

variabel fomal: A B

saat awal I.S.:

* m1 = X = 5 dan m2 = Y = 7
* A mengacu alamat nilai m1 dan B ke alamat nilai m2

?

m2?

A, B, dan Temp bersifat sementara

hanya ada saat Procedure diproses,

setelah proses A, B, Temp dihapus.

Jadi tinggal X dan Y masih mengacu

alamat memori m1 dan m2

saat akhir F.S.:

* m1 = A = 7 dan m2 = B = 5 ,
* X mengacu alamat m1 dan Y mengacu alamat m2
* A , B lalu di hapus (dealokasi)

contoh-contoh pemanggilan lain yang mungkin dilakukan dijelaskan

sebagai berikut :

X 🡨 50

Y 🡨 100

**Output**(X,Y) // X=50 dan Y100

**Call** TukarNilai(X,Y) // True, Call By Reference

**Output**(X,Y) // X=100 dan Y=50

**Call** TukarNilai(50,100) // Error, 50,100 not call by reference

**Call** TukarNilai(50,Y) // Error, 50 not call by reference

**Call** TukarNilai(X,100) // Error, 100 not call by reference

1. **Call By Copy**

Call By Copy terbagi lagi menjadi tiga jenis yaitu Call By Value, Call By Result, dan Call By Value-Result.

1. **Call By Value**

Unit pemamnggil mengevaluasi nilai parameter aktual untuk menginisialisasi parameter formal untuk proses. Jadi, setelah proses selesai nilai parameter formal tidak bisa dipulangbalikkan pada parameter aktual. **Catatan** : parameter formal sesuai dengan tipe **In**.

**Procedure** TukarNilai(in A, B : Integer);

Kamus lokal :

Temp : Integer

Algoritma

Temp 🡨 A

A 🡨 B

B 🡨 Temp

End;

Misal **var** X,Y:Integer, dan X 🡨 5, Y 🡨 7 maka Call By Value,

**Call TukarNilai**(X,Y)terjadi penggunaan variabel di memory (RAM) diilustrasikan sbb:

Memori (RAM) :

Temp

m1?

variabel aktual: X Y

variabel fomal: A B

saat awal I.S.:

* X mengacu nilai 5, Y mengacu nilai 7
* A = 5 dicopy dari X dan B = 7 dicopy dari Y

?

m2?

A, B, dan Temp bersifat sementara hanya ada saat Procedure diproses, setelah proses A, B, Temp dihapus.

saat akhir F.S.:

* X dan Y tetap mengacu alamat 5 dan 7
* m1 = 7 diubah oleh A dan m2 = 5 diubah oleh B, tapi

kemudian semua A , B, Temp di hapus (dealokasi)

5

7

contoh-contoh pemanggilan lain yang mungkin dilakukan dijelaskan sebagai berikut :

X 🡨 50

Y 🡨 100

**Output**(X,Y) // X=50 dan Y=100

**Call** TukarNilai(X,Y) // Call By Value dengan variabel

**Call** TukarNilai(50,100) // Call By Value dengan nilai

**Call** TukarNilai(100,50) // Call By Value dengan nilai

**Call** TukarNilai(X,100) // Call By Value dengan variabel & nilai

**Call** TukarNilai(50,Y) // Call By Value dengan nilai & variabel

**Output**(X,Y) // X=50 dan Y=100

1. **Call By Result**

Unit pemanggil tidak menginisialisasi parameter formal, tetapi hasil akhir proses nilai parameter formal dicopykan kepada parameter aktual. **Catatan** : parameter formal sesuai dengan tipe **Out**.

**Procedure** TukarNilai(Out A, B : Integer);

Kamus lokal :

Temp : Integer

Algoritma

Temp 🡨 A

A 🡨 B

B 🡨 Temp

End;

Misal **var** X,Y:Integer, dan X 🡨 5, Y 🡨 7 maka Call By Result,

**Call TukarNilai**(X,Y) terjadi penggunaan variabel di memory (RAM) diilustrasikan sbb:

Memori (RAM) :

Temp

m1?

variabel aktual: X Y

variabel fomal: A B

saat awal I.S.:

* X mengacu nilai 5, Y mengacu nilai 7

?

m2?

A, B, dan Temp bersifat sementara hanya ada saat Procedure diproses, setelah proses A, B, Temp dihapus.

saat akhir F.S.:

* A mengacu nilai m1 dan B mengacu nilai m2
* m1 = 7 diproses A dan m2 = 5 diproses oleh B
* X = 7 dicopy dari A dan B= 5 dicopy dari B
* lalu semua A , B, Temp di hapus (dealokasi)

5

7

Contoh-contoh pemanggilan lain yang mungkin dilakukan dijelaskan sebagai berikut :

X 🡨 50

Y 🡨 100

**Output**(X,Y) // X=50 dan Y=100

**Call** TukarNilai(X,Y) // True, Call By Result

**Output**(X,Y) // X=100 dan Y=50

**Call** TukarNilai(50,100) // error, 50,100 not Call By Result

**Call** TukarNilai(X,100) // error, 100 not Call By Result

**Call** TukarNilai(50,Y) // error, 50 not Call By Result

1. **Call By Value-Result**

Unit pemanggil menginisialisasi parameter formal dan hasil akhir parameter formal dicopykan kepada parameter aktual. Meskipun mekanisme berbeda, efek parameter aktual dari mekanisme ini sama dengan Call by Reference. **Catatan** : parameter formal sesuai dengan tipe **In**/**Out**.

1. **Call By Name**

Unit pemanggil melewatkan nama dari parameter aktual untuk mengganti nama parameter formal secara tekstual (Replaced textually). **Catatan** : parameter formal sesuai dengan tipe **In**/**Out**.

Contoh :

**Procedure** TukarNilai(In/Out A, B : Integer);

Kamus lokal :

Temp : Integer

Algoritma

Temp 🡨 A

A 🡨 B

B 🡨 Temp

End;

Misal **var** X,Y : Integer

X 🡨 50

Y 🡨 100

**Output**(X,Y) // X=50 dan B=100

**Call** TukarNilai(X,Y) // Call By Name, dgn proses :

// Saat call :

// A diganti X, B diganti Y

// Saat proses

// Temp 🡨 X

// X 🡨 Y

// Y 🡨 Temp

**Output**(X,Y) // X=100 dan B=50

## Skop dan Penamaan Variabel

**A. Skop Variabel**

Variabel yang ditulis dalam kamus data program disebut variabel global, sedangkan variabel yang ditulis dalam kamus data subprogram (prosedur/fungsi) disebut variabel lokal. Variabel global berarti variabel yang memilki skop (ruang lingkup) global atau nama/identitas variabel tersebut dikenal pada seluruh bagian program termasuk di dalam prosedur dan fungsi. Variabel lokal memilki skop lokal atau hanya dikenal pada prosedur tersebut. (lihat gambar skema skop variabel)

**Variabel global** memilki siklus hidup (life time) selama program aktif dari awal sampai dengan selesai (terminated). **Variabel lokal** memiliki siklus hidup selama subprogram tersebut aktif (di-*call* ). Variabel lokal akan muncul saat subprogram diaktifkan dan setelah subprogram tidak aktif maka variabel lokal tersebut ikut hilang. Identifikasi dan pemakaian yang tepat variabel global dan lokal akan mempengaruhi penggunaan memori setiap saat.

Skop Variabel

**Skema** **Program**

**Subprogram-1**

Algoritma

:

:

End;

**Kamus Lokal**

**Var** VL1 : {type}

**Algoritma**

**:**

**:**

**End.**

**Kamus Global**

**Var VG :** {type}

**Subprogram-2**

**Algoritma**

**:**

**:**

**End;**

**Kamus Lokal**

**Var** VL2 : {type}

**Subprogram**-2.1

Algoritma

:

:

End;

**Kamus Lokal**

**Var** VL2.1 : {type}

* 1. Diagram skop variabel

Berdasarkan contoh skema yang terlihat pada gambar tersebut di atas, variabel memiliki skop sebagai berikut :

* + 1. VG memiliki skop global untuk seluruh program dan subprogram sehingga VG dikenal di subprogram-1, subprogram-2, subprogram-2.1
    2. VL1 memiliki skop lokal terhadap subprogram-1, sehingga variabel yang dikenal adalah VG dan VL1.
    3. VL2 memiliki skop lokal terhadap subprogram-2, tetapi memiliki skop global terhadap subprogram-2.1. Varibel yang dikenal dalam subprogram ini adalah VG dan VL2.
    4. VL2.1 memiliki skop lokal terhadap subprogram-2.1, tetapi memiliki induk pada subprogram-2. Variabel yang dikenal VG, VL2, dan VL2.1.

**B. Penamaan Parameter dan Variabel Lokal**

Penamaan parameter dan variabel lokal memiliki kriteria yang sama dengan cara penamaan identifikasi data (*indentifier*) ditambah satu aturan lagi, secara lengkap sebagai berikut :

* Minimal satu karakter dan harus dimulai dengan huruf (selanjutnya boleh kombinasi huruf & angka)
* Nama tidak boleh ada spasi (biasanya spasi diganti dengan tanda *underscore* “\_” sehingga tetap menyambung sebagai satu kata)
* Tidak boleh ada karakter-karakter yang merupakan tanda operator matematika ( +, -, \*, &, %, ^)
* Tidak boleh ada karakter-karakter lain yang berarti khusus (!, ?, # . ~, ”,”,(,{,[, ],},), dan lain-lain)
* **Tambahan** : nama parameter pada prinsipnya adalah variabel lokal, sehingga nama paramater BOLEH sama dengan nama variabel global, tetapi nama parameter TIDAK BOLEH sama dengan nama variabel lokal pada satu subprogram. Contoh :

**Contoh :**

Program Penamaan\_Parameter\_Variabel;

Kamus {global}

Type : Data1=Integer; Data2=String; Data3=Char; Data4:real;

Var : X:Data1; Y:Data2; Z:Data3;

Procedure Penamaan(X:Data1; Y:Data2; A:Data3; B:Data4)

Kamus {lokal}

Var X:Data1; Z:Data2; B:Data1; C:Data4;

Algoritma

Output(X,Y,Z,A,B,C);

End;

Algoritma {program}

X 🡨 1; Y🡨’Satu’; Z🡨’1’;

CALL Penamaan(X,Y,Z)

End.

**Keterangan**

X = global=parameter=lokal 🡪 *error*: X duplikat parameter=lokal

Y = global=parameter 🡪 boleh

Z = global=lokal 🡪 boleh

A = parameter 🡪 boleh

B = parameter = lokal 🡪 *error*: X duplikat parameter=lokal

C = lokal 🡪 boleh

## Studi Kasus Implementasi Skema Turbo Pascal

Contoh implementasi skema algoritma program menggunakan bahasa turbo pascal disajikan sebagai berikut :

**A. Contoh Skema Program Utama**

{\* ===== JUDUL PROBLEM ===== \*}

Program Nama\_Program\_Template;

{\* deskripsi input \*}

{\* deskripsi output \*}

{\* deskripsi proses \*}

{\* ===== KAMUS GLOBAL ===== \*}

{\* --- Deklarasi Link --- \*}

Uses CRT;

{\* --- Deklarasi Data --- \*}

Type NamaTipe = InformasiTipeData;

Var VG : NamaTipe;

{\* --- Deklarasi Prosedur/Fungsi --- \*}

Procedure NamaProsedure(Parameter\_Formal);

Function NamaFungsi(Parameter\_Formal) : InformatiTipeData;

{\* ===== ALGORITMA UTAMA ===== \*}

BEGIN

Clrscr;

GotoXY(20,12); Write('Template program sukses ! ...');

END.

**B. Contoh Program dengan Subprogram Mode Internal**

{\* ===== JUDUL PROBLEM ===== \*}

**Program Analisis\_Skop\_Variabel;**

{\* Menganalisis Skop Variabel Global vs Varibal Lokal \*}

{\* Menganalisis Nama Variabel Global, Parameter, Lokal \*}

{\* Memberikan intuisi penamaan variabel/parameter yang baik \*}

{\* Nama Parameter tidak boleh sama dengan Variabel global/lokal \*}

{\* ===== KAMUS GLOBAL ===== \*}

**Uses CRT;**

**Type InfoType = String;**

**Var G1 : InfoType;**

**G2 : InfoType;**

{\* --- Deklarasi PROSEDUR/FUNGSI --- \*}

**Function Nilai\_Skop(Fx : Integer) : String;**

**begin**

**If (Fx=0) then**

**Nilai\_Skop := '999'**

**else**

**Nilai\_Skop := '000';**

**end;**

**Procedure Analisis\_Skop(var Px : InfoType); {/Call by Reference/}**

**var G1 : InfoType; { nama G1 lokal=G1 global }**

**{ Lokal terbentuk saat di-call, hilang setelah call}**

**begin**

**Writeln('--G1/G2 Global, G1 juga nama Lokal, Call ANALISIS\_SKOP(G1) :');**

**Writeln('--G1/G2 TANPA perubahan di dalam prosedur yang diCall: ');**

**Writeln('5. Variabel G1 = ', G1); {/skop lokal tidak terinisialisasi/}**

**Writeln('6. Variabel G2 = ', G2); {/skop global dalam prosedur/}**

**Writeln('--G1/G2 DENGAN perubahan di dalam prosedur yang diCall : ');**

**G1 := Nilai\_Skop(1);**

**G2 := Nilai\_Skop(1);**

**Writeln('7. Variabel G1 = ', G1); {/nilai diubah dalam prosedur/}**

**Writeln('8. Variabel G2 = ', G2); {/nilai diubah dalam prosedur/}**

**end;**

{\* ===== ALGORITMA UTAMA ===== \*}

**BEGIN**

Clrscr;

Writeln('--G1/G2 global tanpa inisialisasi : ');

Writeln('1. Variabel G1 = ', G1); {/skop global tidak terinisialisasi/}

Writeln('2. Variabel G2 = ', G2); {/skop global tidak terinisialisasi/}

Writeln('--G1/G2 global setelah inisialisasi fungsi NILAI\_SKOP(0) : ');

G1 := Nilai\_Skop(0);

G2 := Nilai\_Skop(0);

Writeln('3. Variabel G1 = ', G1); {/skop global terinisialisasi/}

Writeln('4. Variabel G2 = ', G2); {/skop global terinisialisasi/}

Analisis\_Skop(G1); {CALL By Reference}

Writeln('--G1/G2 global setelah pemanggilan prosedur : ');

Writeln('9. Variabel G1 = ', G1); {/skop global setelah call prosedur/}

Writeln('10.Variabel G2 = ', G2); {/skop global setelah call prosedur/}

G2 := Readkey;

**END.**

Hasil/Output :

--G1/G2 global tanpa inisialisasi :

1. Variabel G1 = ? (tidak dapat dipastikan)

2. Variabel G2 = ? (tidak dapat dipastikan)

--G1/G2 global setelah inisialisasi fungsi NILAI\_SKOP = 999 :

3. Variabel G1 = 999

4. Variabel G2 = 999

--G1/G2 global, G1 juga nama lokal, Call ANALISIS\_SKOP(G1) :

--G1/G2 TANPA perubahan di dalam prosedur yang diCall :

5. Variabel G1 = ? (tidak dapat dipastikan)

6. Variabel G2 = 999

--G1/G2 DENGAN perubahan di dalam prosedur yang diCall:

7. Variabel G1 = 000

8. Variabel G2 = 000

--G1/G2 global setelah pemanggilan prosedur : ');

9. Variabel G1 = 999

10.Variabel G2 = 000

**C. Contoh Program dengan Subprogram Mode Eksternal**

{\* ===== JUDUL UNIT ===== \*}

**Unit T\_UNIT;**

{\* Nama unit harus sama dengan nama file \_.pas \*}

{\* T\_UNIT = T\_UNIT.PAS \*}

{\* ===== KAMUS DATA ===== \*}

**INTERFACE**

**USES CRT;**

Type INFOTYPE = Integer;

{\*--- Deklarasi PROSEDUR / FUNGSI--- \*}

Procedure Print(PX : INFOTYPE);

Function IntToDayOfWeek(FX : INFOTYPE) : String;

{\* ===== REALISASI ALGORITMA ===== \*}

**IMPLEMENTATION**

Procedure Print(PX : INFOTYPE);

begin

Writeln(IntToDayOfWeek(PX));

end;

Function IntToDayOfWeek(FX : INFOTYPE) : String;

var Hasil : string;

begin

Hasil := '';

Case FX Of

1 : Hasil := 'Senin';

2 : Hasil := 'Selasa';

3 : Hasil := 'Rabu';

4 : Hasil := 'Kamis';

5 : Hasil := 'Jumat';

6 : Hasil := 'Sabut';

7 : Hasil := 'Minggu';

end;

IntToDayOfWeek:= Hasil;

end;

**END.**

<<< =================================================== >>>

{\* ===== JUDUL PROBLEM ===== \*}

Program Link\_Unit;

{\* ===== KAMUS GLOBAL ===== \*}

Uses CRT, T\_UNIT; {\* --- Link --- \*}

Type InfoType = Integer;

Var G : InfoType;

BEGIN {\* ===== ALGORITMA UTAMA ===== \*}

Clrscr;

G := 3;

{Call} Print(G);

END.

STUDI KASUS ALGORITMA dan IMPLEMENTASI

**1. Tuliskan Output Algoritma Program Di Bawah Ini !**

**Judul Program** LATIHAN\_1 // Amati Output Algoritma modular ini

**Kamus**

Type Nilai = Integer;

Var X,Y,Z : Nilai

// Deklarasi prosedur dan fungsi

**Procedure** Hitung1(in Awal, Akhir: Integer; out Hasil : Nilai);

{I.S.: Awal < akhir sembarang integer positif }

{F.S.: Hasil = Awal + (Awal+1) + (Awal+2} + … + Akhir }

Kamus : X, K : Integer;

Algoritma

X 🡨0;

K Traversal Awal..Akhir

X 🡨 X + K

endK

Hasil 🡨 X

End;

**Procedure** Hitung2(in M,N:Integer; out Hasil : Nilai);

{I.S.: M<N sembarang integer positif, Hasil sembarang integer }

{F.S.: Hasil = M + 1 + 2 + … + N }

Kamus : X : Integer;

Algoritma

Hasil🡨M; X🡨1

While (X<=N) Do

Hasil 🡨 Hasil + X

X 🡨 X + 1

endWhile

End;

**Function** Hitung3(in N:Integer)🡪 Nilai;

{Diberikan N sembarang integer positif }

{Dikrimkan Nilai = 1 + 2 + 3 + … + N }

Kamus : H,X : Integer;

Algoritma

H 🡨 0; X 🡨 1

While (X<=N) Do

H 🡨 H + X

X 🡨 X + 1

endWhile

🡪(H)

End;

**Function** Hitung4(in N:Integer)🡪String;

Kamus : -

Algoritma

If (N>0) Then

🡪(’Benar’)

else

🡪(’Salah’)

End;

**ALGORITMA UTAMA**

X🡨3; Y🡨8; Z🡨5

CALL Hitung1(X, Y, Z) ; Output (X,Y,Z)

CALL Hitung2(10, X, Z); Output (X,Y,Z)

Output(Hitung3(Z))

Output(Hitung4(Z))

**END.**

**2. Buatkan Algoritma Dan Program Kasus Di Bawah Ini !**

Buatlah algoritma untuk membuat program aplikasi sederhana yang bisa membantu adik-adik SD kelas 1, 2, atau 3 untuk belajar berhitung penjumlahan, pengurangan, dan perkalian dua buah bilangan integer.

Proses komputer akan memunculkan pernyataan dua bilangan misalkan A dan B dalam sajian

A + B = ? \_ untuk penjumlahan, atau

A - B = ? \_ untuk pengurangan, atau

A x B = ? \_ untuk Perkalian

Setelah dijawab hasilnya di belakang tanda ?, lalu dienter keluar pesan sbb :

Hasil = …(salah/benar)

Komentar/Saran = ……(teks yang berkaitan)

Mengulangi dengan kembali ke MENU atau Ingin mencoba Lagi (Y/T)? \_

Proses dapat berulang, sesuai menu atau konfirmasi ?, jika Y maka ulang lagi.

Aplikasi tersebut disajikan dengan menggunakan interaksi berupa menu pilihan sbb :

MENU BELAJAR MATEMATIKA SD

Pilih menu huruf = ?\_

[ A ]. PENJUMLAHAN

[ B ]. PENGURANGAN

[ C ]. PERKALIAN

[ Q ]. SELESAI

**TUGAS :**

1. Pelajari terlebih dahulu rancangan algoritma program BERTIGA dibawah ini.
2. Pelajari dan tuliskan contoh implementasi PROGRAM ”BERTIGA” MODEL-1 dan MODEL-2 dalam bahasa turbo pascal, dan amati hasil-hasilnya.
3. Tuliskan algoritma versi lain (MODEL-3) untuk menyelesaikan masalah tersebut, selanjutnya buatkan programnnya dengan turbo pascal dan demokan hasilnya di depan kelas.

**Rancangan Algoritma Program :**

**Program**  BERTIGA;

// Aplikasi latihan berhitung untuk SD kelas 3 - bertiga

**Kamus**

Constanta Selesai = ‘Q’

Type KomSar = String;

Bilangan = Integer;

MenuPilihan = Character;

Jawaban = boolean

Var Bil1, Bil2 : Bilangan;

MP : MenuPilihan;

Jwb : Jawaban;

// Deklarasi prosedur dan fungsi

**Procedure** MenuBertiga(in/out pilihan: Character);

{I.S.: pilihan sembarang, layar menu akan ditampilkan}

{F.S.: pilhan = A, B, C, atau Q}

**Procedure** ProsesJumlah(in A,B : integer; out H:Jawaban);

{I.S.: A, B random integer untuk sajian soal, C jawaban temporari}

{F.S.: H true jika jawaban C = A + B, H false untuk jawaban salah}

**Procedure** ProsesKurang (in A,B : integer; out H:Jawaban);

{I.S.: A, B random integer untuk sajian soal, C jawaban temporari}

{F.S.: H true jika jawaban C = A - B, H false untuk jawaban salah}

**Procedure** ProsesKali (in A,B : integer; out H:Jawaban);

{I.S.: A, B random integer untuk sajian soal, C jawaban temporari}

{F.S.: H true jika jawaban C = A x B, H false untuk jawaban salah}

**Function** Komentar\_Saran(H:Jawaban)🡪KomSar;

{Diinput : H boolean sembarang }

{Dikirim : teks komentar atau saran yang diperlukan}

**ALGORITMA UTAMA**

**Repeat**

**Call** MenuBertiga(MP);

Bil1🡨 Random;

Bil2🡨 Random;

Depend on MP

‘A’ : ProsesJumlah(Bil1,Bil2,Jwb);

‘B’ : ProsesKurang (Bil1,Bil2,Jwb);

‘C’ : ProsesKali(Bil1,Bil2,Jwb);

EndDepend

Output(mn,’’,Komentar\_Saran(Jwb))

**Until(MP=Selesai);**

**END.**

**IMPLEMETASI PROGRAM DALAM TURBO PASCAL**

**A. PROGRAM BERTGA MODEL-1**

Program BERTIGA\_Model1;

{ ----------------------------------------------------------------------}

{ Program ini menggunakan subprogram yang terdiri : }

{ (1) 4 prosedur : MenuBertiga, ProsesJumlah, ProsesKurang, ProsesKali }

{ (2) 1 fungsi : Komentar\_Saran } {-----------------------------------------------------------------------}

**{<<< Link >>>}**

uses Crt;

**{<<< KAMUS GLOBAL >>>}**

Const Selesai = 'Q';

Type KomSar = String; { Komentar dan Saran }

Bilangan = Integer; { Bilangan yang dioperasikan}

MenuPilihan=Char; { Pilihan huruf pada menu}

Jawaban=Boolean; { Pengetesan jawaban}

Var Bil1, Bil2 : Bilangan;

MP : MenuPilihan;

Jwb : Jawaban;

**{<<< DEKLARASI PROSEDUR DAN FUNGSI >>>}**

Procedure MenuBertiga( var Pilihan : MenuPilihan);

var X, Y : Integer;

begin

ClrScr;

X := 20; Y := 3;

GotoXY(X,Y); Write('\* ----------------------------------\*');

GotoXY(X,Y+01); Write('| MENU MATEMATIKA SD KELAS 1-3 |');

GotoXY(X,Y+02); Write('\* ----------------------------------\*');

GotoXY(X,Y+03); Write('| [A] PENJUMLAHAN ( M + N ) |');

GotoXY(X,Y+04); Write('| [B] PENGURANGAN ( M - N ) |');

GotoXY(X,Y+05); Write('| [C] PERKALIAN ( M x N ) |');

GotoXY(X,Y+06); Write('| |');

GotoXY(X,Y+07); Write('| [Q] SELESAI |');

GotoXY(X,Y+08); Write('\* ----------------------------------\*');

GotoXY(X,Y+09); Write('| Pilih menu huruf = \_ |');

GotoXY(X,Y+10); Write('\* ----------------------------------\*');

GotoXY(X+25,Y+09); Read(Pilihan);

end;

Procedure ProsesJumlah(A,B:Bilangan; var H:Jawaban);

var C : Bilangan;

begin

ClrScr;

GotoXY(30,10); Write(A,' + ',B,' = '); Readln(C);

If (A+B=C) then H:=True else H:=False;

end;

Procedure ProsesKurang(A,B:Bilangan; var H:Jawaban);

var C : Bilangan;

begin

ClrScr;

GotoXY(30,10); Write(A,' - ',B,' = '); Readln(C);

If (A-B=C) then H:=True else H:=False;

end;

Procedure ProsesKali(A,B:Bilangan; var H:Jawaban);

var C : Bilangan;

begin

ClrScr;

GotoXY(30,10); Write(A,' x ',B,' = '); Readln(C);

If (A\*B=C) then H:=True else H:=False;

end;

Function Komentar\_Saran(H:Jawaban): KomSar;

var ks : String;

begin

If H then

ks := 'Jawaban Anda BENAR ! Anda dapat BINTANG \*'

else

ks := 'Jawaban SALAH! Coba lagi, jangan putus asa !';

Komentar\_Saran := ks;

end;

**Begin {<<< ALGORITMA UTAMA >>>}**

Repeat

MenuBertiga(MP);

If (MP<> Selesai) then begin

Randomize; {Bangkitkan bilangan random antara 0 dan 1 }

Bil1 := Abs(Trunc(Random\*100)); {Ambil bilangan 0 s/d 100}

Bil2 := Abs(Trunc(Random\*100));

Case UpCase(MP) Of

'A' : ProsesJumlah(Bil1, Bil2, Jwb);

'B' : ProsesKurang(Bil1, Bil2, Jwb);

'C' : ProsesKali(Bil1, Bil2, Jwb);

end;

GotoXY(20,12); Writeln(Komentar\_Saran(Jwb));

GotoXY(20,16); Writeln('Tekan <ENTER> untuk melanjutkan ...');

Readln;

end;

Until (Upcase(MP)=Selesai);

**End.**

**B. PROGRAM BERTIGA MODEL-2**

Program BERTIGA\_Model2;

{ ----------------------------------------------------------------------}

{ Program ini menggunakan subprogram yang terdiri : }

{ (1) 1 prosedur : MenuBertiga, ProsesJumlah, ProsesKurang, ProsesKali }

{ (2) 1 fungsi : Komentar\_Saran } {-----------------------------------------------------------------------}

**{<<< Link >>>}**

uses Crt;

**{<<< KAMUS GLOBAL >>>}**

Const Selesai = 'Q';

Type KomSar = String; { Komentar dan Saran }

Bilangan = Integer; { Bilangan yang dioperasikan}

MenuPilihan=Char; { Pilihan huruf pada menu}

Jawaban=Boolean; { Pengetesan jawaban}

TandaSoal =Char; { Tanda model soal +, -, x}

Var Bil1, Bil2 : Bilangan;

MP : MenuPilihan;

Jwb : Jawaban;

**{<<< DEKLARASI PROSEDUR DAN FUNGSI >>>}**

Procedure MenuBertiga( var Pilihan : MenuPilihan);

var X, Y : Integer;

begin

ClrScr;

X := 20; Y := 3;

GotoXY(X,Y); Write('\* ----------------------------------\*');

GotoXY(X,Y+01); Write('| MENU MATEMATIKA SD KELAS 1-3 |');

GotoXY(X,Y+02); Write('\* ----------------------------------\*');

GotoXY(X,Y+03); Write('| [A] PENJUMLAHAN ( M + N ) |');

GotoXY(X,Y+04); Write('| [B] PENGURANGAN ( M - N ) |');

GotoXY(X,Y+05); Write('| [C] PERKALIAN ( M x N ) |');

GotoXY(X,Y+06); Write('| |');

GotoXY(X,Y+07); Write('| [Q] SELESAI |');

GotoXY(X,Y+08); Write('\* ----------------------------------\*');

GotoXY(X,Y+09); Write('| Pilih menu huruf = \_ |');

GotoXY(X,Y+10); Write('\* ----------------------------------\*');

GotoXY(X+25,Y+09); Read(Pilihan);

end;

Procedure ProsesSoalJawab(T:TandaSoal; A,B:Bilangan; var H:Jawaban);

var C : Bilangan;

begin

ClrScr;

GotoXY(30,10); Write(A:10:0,' ',T,' ',B:10:0,' = '); Readln(C);

Case T Of

'+' : If (A+B=C) then H:=True else H:=False;

'-' : If (A-B=C) then H:=True else H:=False;

'x' : If (A\*B=C) then H:=True else H:=False;

end;

end;

Function Komentar\_Saran(H:Jawaban): KomSar;

var ks : String;

begin

If H then

ks := 'Jawaban Anda BENAR ! Anda dapat BINTANG \*'

else

ks := 'Jawaban SALAH! Coba lagi, jangan putus asa !';

Komentar\_Saran := ks;

end;

**Begin {<<< Algoritma Utama >>>}**

Repeat

MenuBertiga(MP);

If (MP<> Selesai) then begin

Randomize; {Bangkitkan bilangan random antara 0 dan 1 }

Bil1 := Abs(Trunc(Random\*100)); {Ambil bilangan 0 s/d 100}

Bil2 := Abs(Trunc(Random\*100));

Case UpCase(MP) Of

'A' : ProsesSoalJawab('+', Bil1, Bil2, Jwb);

'B' : ProsesSoalJawab('-', Bil1, Bil2, Jwb);

'C' : ProsesSoalJawab('x', Bil1, Bil2, Jwb);

end;

GotoXY(20,12); Writeln(Komentar\_Saran(Jwb));

GotoXY(20,16); Writeln('Tekan <ENTER> untuk melanjutkan ...');

Readln;

end;

Until (Upcase(MP)=Selesai);

**End.**

## Studi Kasus Aplikasi Spreadsheet Excel

Definisi algoritma modular pada studi kasus aplikasi Ms-Excel 2007 dinyatakan bahwa Ms-Excel 2007 adalah sebuah skema algoritma “Progam” aplikasi yang memiliki 11 “Modul” dasar dimana setiap modul dinyatakan dalam bentuk “Fungsi” dengan referensi kelengkapan 337 elemen fungsi. Komposisi tersebut diuraikan dalam tabel dibawah ini.

Komposisi modul aplikasi Ms-Excel 2007

|  |  |  |
| --- | --- | --- |
| No | Nama Modul | Jumlah Fungsi |
| 1 | Statistical | 83 |
| 2 | Lookup & Reference | 17 |
| 3 | Text & Data | 21 |
| 4 | Logical | 7 |
| 5 | Math & Trigonometry | 60 |
| 6 | Database | 12 |
| 7 | Financial | 53 |
| 8 | Date & Time | 20 |
| 9 | Information | 17 |
| 10 | Egineering | 40 |
| 11 | Cube | 7 |
| **Total Fungsi =** | | **337** |

### Modul “Statistical”

|  |  |
| --- | --- |
| **Function** | **Description** |
| [AVEDEV](ms-help://MS.EXCEL.12.1033/EXCEL/content/HP10062481.htm) | Returns the average of the absolute deviations of data points from their mean |
| [AVERAGE](ms-help://MS.EXCEL.12.1033/EXCEL/content/HP10062482.htm) | Returns the average of its arguments |
| [AVERAGEA](ms-help://MS.EXCEL.12.1033/EXCEL/content/HP10062483.htm) | Returns the average of its arguments, including numbers, text, and logical values |
| [AVERAGEIF](ms-help://MS.EXCEL.12.1033/EXCEL/content/HA10047433.htm) | Returns the average (arithmetic mean) of all the cells in a range that meet a given criteria |
| [AVERAGEIFS](ms-help://MS.EXCEL.12.1033/EXCEL/content/HA10047493.htm) | Returns the average (arithmetic mean) of all cells that meet multiple criteria. |
| [BETADIST](ms-help://MS.EXCEL.12.1033/EXCEL/content/HP10062484.htm) | Returns the beta cumulative distribution function |
| [BETAINV](ms-help://MS.EXCEL.12.1033/EXCEL/content/HP10062485.htm) | Returns the inverse of the cumulative distribution function for a specified beta distribution |
| [BINOMDIST](ms-help://MS.EXCEL.12.1033/EXCEL/content/HP10062486.htm) | Returns the individual term binomial distribution probability |
| [CHIDIST](ms-help://MS.EXCEL.12.1033/EXCEL/content/HP10062487.htm) | Returns the one-tailed probability of the chi-squared distribution |
| [CHIINV](ms-help://MS.EXCEL.12.1033/EXCEL/content/HP10062488.htm) | Returns the inverse of the one-tailed probability of the chi-squared distribution |
| [CHITEST](ms-help://MS.EXCEL.12.1033/EXCEL/content/HP10062489.htm) | Returns the test for independence |
| [CONFIDENCE](ms-help://MS.EXCEL.12.1033/EXCEL/content/HP10062490.htm) | Returns the confidence interval for a population mean |
| [CORREL](ms-help://MS.EXCEL.12.1033/EXCEL/content/HP10062491.htm) | Returns the correlation coefficient between two data sets |
| [COUNT](ms-help://MS.EXCEL.12.1033/EXCEL/content/HP10062492.htm) | Counts how many numbers are in the list of arguments |
| [COUNTA](ms-help://MS.EXCEL.12.1033/EXCEL/content/HP10062493.htm) | Counts how many values are in the list of arguments |
| [COUNTBLANK](ms-help://MS.EXCEL.12.1033/EXCEL/content/HP10062393.htm) | Counts the number of blank cells within a range |
| [COUNTIF](ms-help://MS.EXCEL.12.1033/EXCEL/content/HP10069840.htm) | Counts the number of cells within a range that meet the given criteria |
| [COUNTIFS](ms-help://MS.EXCEL.12.1033/EXCEL/content/HA10047494.htm) | Counts the number of cells within a range that meet multiple criteria |
| [COVAR](ms-help://MS.EXCEL.12.1033/EXCEL/content/HP10062494.htm) | Returns covariance, the average of the products of paired deviations |
| [CRITBINOM](ms-help://MS.EXCEL.12.1033/EXCEL/content/HP10062495.htm) | Returns the smallest value for which the cumulative binomial distribution is less than or equal to a criterion value |
| [DEVSQ](ms-help://MS.EXCEL.12.1033/EXCEL/content/HP10062496.htm) | Returns the sum of squares of deviations |
| [EXPONDIST](ms-help://MS.EXCEL.12.1033/EXCEL/content/HP10062497.htm) | Returns the exponential distribution |
| [FDIST](ms-help://MS.EXCEL.12.1033/EXCEL/content/HP10062498.htm) | Returns the F probability distribution |
| [FINV](ms-help://MS.EXCEL.12.1033/EXCEL/content/HP10062499.htm) | Returns the inverse of the F probability distribution |
| [FISHER](ms-help://MS.EXCEL.12.1033/EXCEL/content/HP10062500.htm) | Returns the Fisher transformation |
| [FISHERINV](ms-help://MS.EXCEL.12.1033/EXCEL/content/HP10062501.htm) | Returns the inverse of the Fisher transformation |
| [FORECAST](ms-help://MS.EXCEL.12.1033/EXCEL/content/HP10062502.htm) | Returns a value along a linear trend |
| [FREQUENCY](ms-help://MS.EXCEL.12.1033/EXCEL/content/HP10062503.htm) | Returns a frequency distribution as a vertical array |
| [FTEST](ms-help://MS.EXCEL.12.1033/EXCEL/content/HP10062504.htm) | Returns the result of an F-test |
| [GAMMADIST](ms-help://MS.EXCEL.12.1033/EXCEL/content/HP10062505.htm) | Returns the gamma distribution |
| [GAMMAINV](ms-help://MS.EXCEL.12.1033/EXCEL/content/HP10062506.htm) | Returns the inverse of the gamma cumulative distribution |
| [GAMMALN](ms-help://MS.EXCEL.12.1033/EXCEL/content/HP10062507.htm) | Returns the natural logarithm of the gamma function, Γ(x) |
| [GEOMEAN](ms-help://MS.EXCEL.12.1033/EXCEL/content/HP10062508.htm) | Returns the geometric mean |
| [GROWTH](ms-help://MS.EXCEL.12.1033/EXCEL/content/HP10062509.htm) | Returns values along an exponential trend |
| [HARMEAN](ms-help://MS.EXCEL.12.1033/EXCEL/content/HP10062510.htm) | Returns the harmonic mean |
| [HYPGEOMDIST](ms-help://MS.EXCEL.12.1033/EXCEL/content/HP10062511.htm) | Returns the hypergeometric distribution |
| [INTERCEPT](ms-help://MS.EXCEL.12.1033/EXCEL/content/HP10062512.htm) | Returns the intercept of the linear regression line |
| [KURT](ms-help://MS.EXCEL.12.1033/EXCEL/content/HP10062513.htm) | Returns the kurtosis of a data set |
| [LARGE](ms-help://MS.EXCEL.12.1033/EXCEL/content/HP10062514.htm) | Returns the k-th largest value in a data set |
| [LINEST](ms-help://MS.EXCEL.12.1033/EXCEL/content/HP10069838.htm) | Returns the parameters of a linear trend |
| [LOGEST](ms-help://MS.EXCEL.12.1033/EXCEL/content/HP10069839.htm) | Returns the parameters of an exponential trend |
| [LOGINV](ms-help://MS.EXCEL.12.1033/EXCEL/content/HP10062515.htm) | Returns the inverse of the lognormal distribution |
| [LOGNORMDIST](ms-help://MS.EXCEL.12.1033/EXCEL/content/HP10062516.htm) | Returns the cumulative lognormal distribution |
| [MAX](ms-help://MS.EXCEL.12.1033/EXCEL/content/HP10062517.htm) | Returns the maximum value in a list of arguments |
| [MAXA](ms-help://MS.EXCEL.12.1033/EXCEL/content/HP10062518.htm) | Returns the maximum value in a list of arguments, including numbers, text, and logical values |
| [MEDIAN](ms-help://MS.EXCEL.12.1033/EXCEL/content/HP10062519.htm) | Returns the median of the given numbers |
| [MIN](ms-help://MS.EXCEL.12.1033/EXCEL/content/HP10062520.htm) | Returns the minimum value in a list of arguments |
| [MINA](ms-help://MS.EXCEL.12.1033/EXCEL/content/HP10062521.htm) | Returns the smallest value in a list of arguments, including numbers, text, and logical values |
| [MODE](ms-help://MS.EXCEL.12.1033/EXCEL/content/HP10062522.htm) | Returns the most common value in a data set |
| [NEGBINOMDIST](ms-help://MS.EXCEL.12.1033/EXCEL/content/HP10062523.htm) | Returns the negative binomial distribution |
| [NORMDIST](ms-help://MS.EXCEL.12.1033/EXCEL/content/HP10062524.htm) | Returns the normal cumulative distribution |
| [NORMINV](ms-help://MS.EXCEL.12.1033/EXCEL/content/HP10062525.htm) | Returns the inverse of the normal cumulative distribution |
| [NORMSDIST](ms-help://MS.EXCEL.12.1033/EXCEL/content/HP10062526.htm) | Returns the standard normal cumulative distribution |
| [NORMSINV](ms-help://MS.EXCEL.12.1033/EXCEL/content/HP10062527.htm) | Returns the inverse of the standard normal cumulative distribution |
| [PEARSON](ms-help://MS.EXCEL.12.1033/EXCEL/content/HP10062528.htm) | Returns the Pearson product moment correlation coefficient |
| [PERCENTILE](ms-help://MS.EXCEL.12.1033/EXCEL/content/HP10062529.htm) | Returns the k-th percentile of values in a range |
| [PERCENTRANK](ms-help://MS.EXCEL.12.1033/EXCEL/content/HP10062530.htm) | Returns the percentage rank of a value in a data set |
| [PERMUT](ms-help://MS.EXCEL.12.1033/EXCEL/content/HP10062531.htm) | Returns the number of permutations for a given number of objects |
| [POISSON](ms-help://MS.EXCEL.12.1033/EXCEL/content/HP10062532.htm) | Returns the Poisson distribution |
| [PROB](ms-help://MS.EXCEL.12.1033/EXCEL/content/HP10062533.htm) | Returns the probability that values in a range are between two limits |
| [QUARTILE](ms-help://MS.EXCEL.12.1033/EXCEL/content/HP10062534.htm) | Returns the quartile of a data set |
| [RANK](ms-help://MS.EXCEL.12.1033/EXCEL/content/HP10062535.htm) | Returns the rank of a number in a list of numbers |
| [RSQ](ms-help://MS.EXCEL.12.1033/EXCEL/content/HP10062536.htm) | Returns the square of the Pearson product moment correlation coefficient |
| [SKEW](ms-help://MS.EXCEL.12.1033/EXCEL/content/HP10062537.htm) | Returns the skewness of a distribution |
| [SLOPE](ms-help://MS.EXCEL.12.1033/EXCEL/content/HP10062538.htm) | Returns the slope of the linear regression line |
| [SMALL](ms-help://MS.EXCEL.12.1033/EXCEL/content/HP10062539.htm) | Returns the k-th smallest value in a data set |
| [STANDARDIZE](ms-help://MS.EXCEL.12.1033/EXCEL/content/HP10062540.htm) | Returns a normalized value |
| [STDEV](ms-help://MS.EXCEL.12.1033/EXCEL/content/HP10062541.htm) | Estimates standard deviation based on a sample |
| [STDEVA](ms-help://MS.EXCEL.12.1033/EXCEL/content/HP10062542.htm) | Estimates standard deviation based on a sample, including numbers, text, and logical values |
| [STDEVP](ms-help://MS.EXCEL.12.1033/EXCEL/content/HP10062543.htm) | Calculates standard deviation based on the entire population |
| [STDEVPA](ms-help://MS.EXCEL.12.1033/EXCEL/content/HP10062544.htm) | Calculates standard deviation based on the entire population, including numbers, text, and logical values |
| [STEYX](ms-help://MS.EXCEL.12.1033/EXCEL/content/HP10062545.htm) | Returns the standard error of the predicted y-value for each x in the regression |
| [TDIST](ms-help://MS.EXCEL.12.1033/EXCEL/content/HP10062546.htm) | Returns the Student's t-distribution |
| [TINV](ms-help://MS.EXCEL.12.1033/EXCEL/content/HP10062547.htm) | Returns the inverse of the Student's t-distribution |
| [TREND](ms-help://MS.EXCEL.12.1033/EXCEL/content/HP10062548.htm) | Returns values along a linear trend |
| [TRIMMEAN](ms-help://MS.EXCEL.12.1033/EXCEL/content/HP10062549.htm) | Returns the mean of the interior of a data set |
| [TTEST](ms-help://MS.EXCEL.12.1033/EXCEL/content/HP10062550.htm) | Returns the probability associated with a Student's t-test |
| [VAR](ms-help://MS.EXCEL.12.1033/EXCEL/content/HP10062551.htm) | Estimates variance based on a sample |
| [VARA](ms-help://MS.EXCEL.12.1033/EXCEL/content/HP10062552.htm) | Estimates variance based on a sample, including numbers, text, and logical values |
| [VARP](ms-help://MS.EXCEL.12.1033/EXCEL/content/HP10062553.htm) | Calculates variance based on the entire population |
| [VARPA](ms-help://MS.EXCEL.12.1033/EXCEL/content/HP10062554.htm) | Calculates variance based on the entire population, including numbers, text, and logical values |
| [WEIBULL](ms-help://MS.EXCEL.12.1033/EXCEL/content/HP10062555.htm) | Returns the Weibull distribution |
| [ZTEST](ms-help://MS.EXCEL.12.1033/EXCEL/content/HP10062556.htm) | Returns the one-tailed probability-value of a z-test |

### Modul “Lookup & Reference”

|  |  |
| --- | --- |
| **Function** | **Description** |
| [ADDRESS](ms-help://MS.EXCEL.12.1033/EXCEL/content/HP10062407.htm) | Returns a reference as text to a single cell in a worksheet |
| [AREAS](ms-help://MS.EXCEL.12.1033/EXCEL/content/HP10062408.htm) | Returns the number of areas in a reference |
| [CHOOSE](ms-help://MS.EXCEL.12.1033/EXCEL/content/HP10069830.htm) | Chooses a value from a list of values |
| [COLUMN](ms-help://MS.EXCEL.12.1033/EXCEL/content/HP10062409.htm) | Returns the column number of a reference |
| [COLUMNS](ms-help://MS.EXCEL.12.1033/EXCEL/content/HP10062410.htm) | Returns the number of columns in a reference |
| [HLOOKUP](ms-help://MS.EXCEL.12.1033/EXCEL/content/HP10062411.htm) | Looks in the top row of an array and returns the value of the indicated cell |
| [HYPERLINK](ms-help://MS.EXCEL.12.1033/EXCEL/content/HP10062412.htm) | Creates a shortcut or jump that opens a document stored on a network server, an intranet, or the Internet |
| [INDEX](ms-help://MS.EXCEL.12.1033/EXCEL/content/HP10069831.htm) | Uses an index to choose a value from a reference or array |
| [INDIRECT](ms-help://MS.EXCEL.12.1033/EXCEL/content/HP10062413.htm) | Returns a reference indicated by a text value |
| [LOOKUP](ms-help://MS.EXCEL.12.1033/EXCEL/content/HP10069832.htm) | Looks up values in a vector or array |
| [MATCH](ms-help://MS.EXCEL.12.1033/EXCEL/content/HP10062414.htm) | Looks up values in a reference or array |
| [OFFSET](ms-help://MS.EXCEL.12.1033/EXCEL/content/HP10062415.htm) | Returns a reference offset from a given reference |
| [ROW](ms-help://MS.EXCEL.12.1033/EXCEL/content/HP10069833.htm) | Returns the row number of a reference |
| [ROWS](ms-help://MS.EXCEL.12.1033/EXCEL/content/HP10062416.htm) | Returns the number of rows in a reference |
| [RTD](ms-help://MS.EXCEL.12.1033/EXCEL/content/HP10062417.htm) | Retrieves real-time data from a program that supports [COM automation (Automation: A way to work with an application's objects from another application or development tool. Formerly called OLE Automation, Automation is an industry standard and a feature of the Component Object Model (COM).)](javascript:AppendPopup(this,'ofAutomation_1')) |
| [TRANSPOSE](ms-help://MS.EXCEL.12.1033/EXCEL/content/HP10069834.htm) | Returns the transpose of an array |
| [VLOOKUP](ms-help://MS.EXCEL.12.1033/EXCEL/content/HP10069835.htm) | Looks in the first column of an array and moves across the row to return the value of a cell |

### Modul “Text & Data”

|  |  |
| --- | --- |
| **Function** | **Description** |
| [ASC](ms-help://MS.EXCEL.12.1033/EXCEL/content/HP10062557.htm) | Changes full-width (double-byte) English letters or katakana within a character string to half-width (single-byte) characters |
| [BAHTTEXT](ms-help://MS.EXCEL.12.1033/EXCEL/content/HP10062558.htm) | Converts a number to text, using the ß (baht) currency format |
| [CHAR](ms-help://MS.EXCEL.12.1033/EXCEL/content/HP10062559.htm) | Returns the character specified by the code number |
| [CLEAN](ms-help://MS.EXCEL.12.1033/EXCEL/content/HP10062560.htm) | Removes all nonprintable characters from text |
| [CODE](ms-help://MS.EXCEL.12.1033/EXCEL/content/HP10062561.htm) | Returns a numeric code for the first character in a text string |
| [CONCATENATE](ms-help://MS.EXCEL.12.1033/EXCEL/content/HP10062562.htm) | Joins several text items into one text item |
| [DOLLAR](ms-help://MS.EXCEL.12.1033/EXCEL/content/HP10062563.htm) | Converts a number to text, using the $ (dollar) currency format |
| [EXACT](ms-help://MS.EXCEL.12.1033/EXCEL/content/HP10062564.htm) | Checks to see if two text values are identical |
| [FIND, FINDB](ms-help://MS.EXCEL.12.1033/EXCEL/content/HP10069836.htm) | Finds one text value within another (case-sensitive) |
| [FIXED](ms-help://MS.EXCEL.12.1033/EXCEL/content/HP10062565.htm) | Formats a number as text with a fixed number of decimals |
| [JIS](ms-help://MS.EXCEL.12.1033/EXCEL/content/HP10062567.htm) | Changes half-width (single-byte) English letters or katakana within a character string to full-width (double-byte) characters |
| [LEFT, LEFTB](ms-help://MS.EXCEL.12.1033/EXCEL/content/HP10062568.htm) | Returns the leftmost characters from a text value |
| [LEN, LENB](ms-help://MS.EXCEL.12.1033/EXCEL/content/HP10062569.htm) | Returns the number of characters in a text string |
| [LOWER](ms-help://MS.EXCEL.12.1033/EXCEL/content/HP10062570.htm) | Converts text to lowercase |
| [MID, MIDB](ms-help://MS.EXCEL.12.1033/EXCEL/content/HP10062571.htm) | Returns a specific number of characters from a text string starting at the position you specify |
| [PHONETIC](ms-help://MS.EXCEL.12.1033/EXCEL/content/HP10062572.htm) | Extracts the phonetic (furigana) characters from a text string |
| [PROPER](ms-help://MS.EXCEL.12.1033/EXCEL/content/HP10062573.htm) | Capitalizes the first letter in each word of a text value |
| [REPLACE, REPLACEB](ms-help://MS.EXCEL.12.1033/EXCEL/content/HP10062574.htm) | Replaces characters within text |
| [REPT](ms-help://MS.EXCEL.12.1033/EXCEL/content/HP10062575.htm) | Repeats text a given number of times |
| [RIGHT, RIGHTB](ms-help://MS.EXCEL.12.1033/EXCEL/content/HP10062576.htm) | Returns the rightmost characters from a text value |
| [SEARCH, SEARCHB](ms-help://MS.EXCEL.12.1033/EXCEL/content/HP10062577.htm) | Finds one text value within another (not case-sensitive) |
| [SUBSTITUTE](ms-help://MS.EXCEL.12.1033/EXCEL/content/HP10062578.htm) | Substitutes new text for old text in a text string |
| [T](ms-help://MS.EXCEL.12.1033/EXCEL/content/HP10062579.htm) | Converts its arguments to text |
| [TEXT](ms-help://MS.EXCEL.12.1033/EXCEL/content/HP10062580.htm) | Formats a number and converts it to text |
| [TRIM](ms-help://MS.EXCEL.12.1033/EXCEL/content/HP10062581.htm) | Removes spaces from text |
| [UPPER](ms-help://MS.EXCEL.12.1033/EXCEL/content/HP10062582.htm) | Converts text to uppercase |
| [VALUE](ms-help://MS.EXCEL.12.1033/EXCEL/content/HP10062583.htm) | Converts a text argument to a number |

### Modul “Logical”

|  |  |
| --- | --- |
| **Function** | **Description** |
| [AND](ms-help://MS.EXCEL.12.1033/EXCEL/content/HP10069828.htm) | Returns TRUE if all of its arguments are TRUE |
| [FALSE](ms-help://MS.EXCEL.12.1033/EXCEL/content/HP10062401.htm) | Returns the logical value FALSE |
| [IF](ms-help://MS.EXCEL.12.1033/EXCEL/content/HP10069829.htm) | Specifies a logical test to perform |
| [IFERROR](ms-help://MS.EXCEL.12.1033/EXCEL/content/HA01231765.htm) | Returns a value you specify if a formula evaluates to an error; otherwise, returns the result of the formula |
| [NOT](ms-help://MS.EXCEL.12.1033/EXCEL/content/HP10062402.htm) | Reverses the logic of its argument |
| [OR](ms-help://MS.EXCEL.12.1033/EXCEL/content/HP10062403.htm) | Returns TRUE if any argument is TRUE |
| [TRUE](ms-help://MS.EXCEL.12.1033/EXCEL/content/HP10062404.htm) | Returns the logical value TRUE |

### Modul “Math & Trigonometry”

|  |  |
| --- | --- |
| **Function** | **Description** |
| [ABS](ms-help://MS.EXCEL.12.1033/EXCEL/content/HP10062418.htm) | Returns the absolute value of a number |
| [ACOS](ms-help://MS.EXCEL.12.1033/EXCEL/content/HP10062419.htm) | Returns the arccosine of a number |
| [ACOSH](ms-help://MS.EXCEL.12.1033/EXCEL/content/HP10062420.htm) | Returns the inverse hyperbolic cosine of a number |
| [ASIN](ms-help://MS.EXCEL.12.1033/EXCEL/content/HP10062421.htm) | Returns the arcsine of a number |
| [ASINH](ms-help://MS.EXCEL.12.1033/EXCEL/content/HP10062422.htm) | Returns the inverse hyperbolic sine of a number |
| [ATAN](ms-help://MS.EXCEL.12.1033/EXCEL/content/HP10062423.htm) | Returns the arctangent of a number |
| [ATAN2](ms-help://MS.EXCEL.12.1033/EXCEL/content/HP10062424.htm) | Returns the arctangent from x- and y-coordinates |
| [ATANH](ms-help://MS.EXCEL.12.1033/EXCEL/content/HP10062425.htm) | Returns the inverse hyperbolic tangent of a number |
| [CEILING](ms-help://MS.EXCEL.12.1033/EXCEL/content/HP10062426.htm) | Rounds a number to the nearest integer or to the nearest multiple of significance |
| [COMBIN](ms-help://MS.EXCEL.12.1033/EXCEL/content/HP10062427.htm) | Returns the number of combinations for a given number of objects |
| [COS](ms-help://MS.EXCEL.12.1033/EXCEL/content/HP10062428.htm) | Returns the cosine of a number |
| [COSH](ms-help://MS.EXCEL.12.1033/EXCEL/content/HP10062429.htm) | Returns the hyperbolic cosine of a number |
| [DEGREES](ms-help://MS.EXCEL.12.1033/EXCEL/content/HP10062430.htm) | Converts radians to degrees |
| [EVEN](ms-help://MS.EXCEL.12.1033/EXCEL/content/HP10062431.htm) | Rounds a number up to the nearest even integer |
| [EXP](ms-help://MS.EXCEL.12.1033/EXCEL/content/HP10062432.htm) | Returns *e* raised to the power of a given number |
| [FACT](ms-help://MS.EXCEL.12.1033/EXCEL/content/HP10062433.htm) | Returns the factorial of a number |
| [FACTDOUBLE](ms-help://MS.EXCEL.12.1033/EXCEL/content/HP10062434.htm) | Returns the double factorial of a number |
| [FLOOR](ms-help://MS.EXCEL.12.1033/EXCEL/content/HP10062435.htm) | Rounds a number down, toward zero |
| [GCD](ms-help://MS.EXCEL.12.1033/EXCEL/content/HP10062436.htm) | Returns the greatest common divisor |
| [INT](ms-help://MS.EXCEL.12.1033/EXCEL/content/HP10062437.htm) | Rounds a number down to the nearest integer |
| [LCM](ms-help://MS.EXCEL.12.1033/EXCEL/content/HP10062438.htm) | Returns the least common multiple |
| [LN](ms-help://MS.EXCEL.12.1033/EXCEL/content/HP10062439.htm) | Returns the natural logarithm of a number |
| [LOG](ms-help://MS.EXCEL.12.1033/EXCEL/content/HP10062440.htm) | Returns the logarithm of a number to a specified base |
| [LOG10](ms-help://MS.EXCEL.12.1033/EXCEL/content/HP10062441.htm) | Returns the base-10 logarithm of a number |
| [MDETERM](ms-help://MS.EXCEL.12.1033/EXCEL/content/HP10062442.htm) | Returns the matrix determinant of an array |
| [MINVERSE](ms-help://MS.EXCEL.12.1033/EXCEL/content/HP10069841.htm) | Returns the matrix inverse of an array |
| [MMULT](ms-help://MS.EXCEL.12.1033/EXCEL/content/HP10069842.htm) | Returns the matrix product of two arrays |
| [MOD](ms-help://MS.EXCEL.12.1033/EXCEL/content/HP10062443.htm) | Returns the remainder from division |
| [MROUND](ms-help://MS.EXCEL.12.1033/EXCEL/content/HP10062444.htm) | Returns a number rounded to the desired multiple |
| [MULTINOMIAL](ms-help://MS.EXCEL.12.1033/EXCEL/content/HP10062445.htm) | Returns the multinomial of a set of numbers |
| [ODD](ms-help://MS.EXCEL.12.1033/EXCEL/content/HP10062446.htm) | Rounds a number up to the nearest odd integer |
| [PI](ms-help://MS.EXCEL.12.1033/EXCEL/content/HP10062447.htm) | Returns the value of pi |
| [POWER](ms-help://MS.EXCEL.12.1033/EXCEL/content/HP10062448.htm) | Returns the result of a number raised to a power |
| [PRODUCT](ms-help://MS.EXCEL.12.1033/EXCEL/content/HP10062449.htm) | Multiplies its arguments |
| [QUOTIENT](ms-help://MS.EXCEL.12.1033/EXCEL/content/HP10062450.htm) | Returns the integer portion of a division |
| [RADIANS](ms-help://MS.EXCEL.12.1033/EXCEL/content/HP10062451.htm) | Converts degrees to radians |
| [RAND](ms-help://MS.EXCEL.12.1033/EXCEL/content/HP10062452.htm) | Returns a random number between 0 and 1 |
| [RANDBETWEEN](ms-help://MS.EXCEL.12.1033/EXCEL/content/HP10062453.htm) | Returns a random number between the numbers you specify |
| [ROMAN](ms-help://MS.EXCEL.12.1033/EXCEL/content/HP10062454.htm) | Converts an arabic numeral to roman, as text |
| [ROUND](ms-help://MS.EXCEL.12.1033/EXCEL/content/HP10062455.htm) | Rounds a number to a specified number of digits |
| [ROUNDDOWN](ms-help://MS.EXCEL.12.1033/EXCEL/content/HP10062456.htm) | Rounds a number down, toward zero |
| [ROUNDUP](ms-help://MS.EXCEL.12.1033/EXCEL/content/HP10062457.htm) | Rounds a number up, away from zero |
| [SERIESSUM](ms-help://MS.EXCEL.12.1033/EXCEL/content/HP10062458.htm) | Returns the sum of a power series based on the formula |
| [SIGN](ms-help://MS.EXCEL.12.1033/EXCEL/content/HP10062459.htm) | Returns the sign of a number |
| [SIN](ms-help://MS.EXCEL.12.1033/EXCEL/content/HP10062460.htm) | Returns the sine of the given angle |
| [SINH](ms-help://MS.EXCEL.12.1033/EXCEL/content/HP10069843.htm) | Returns the hyperbolic sine of a number |
| [SQRT](ms-help://MS.EXCEL.12.1033/EXCEL/content/HP10062461.htm) | Returns a positive square root |
| [SQRTPI](ms-help://MS.EXCEL.12.1033/EXCEL/content/HP10062462.htm) | Returns the square root of (number \* pi) |
| [SUBTOTAL](ms-help://MS.EXCEL.12.1033/EXCEL/content/HP10062463.htm) | Returns a subtotal in a list or database |
| [SUM](ms-help://MS.EXCEL.12.1033/EXCEL/content/HP10062464.htm) | Adds its arguments |
| [SUMIF](ms-help://MS.EXCEL.12.1033/EXCEL/content/HP10062465.htm) | Adds the cells specified by a given criteria |
| [SUMIFS](ms-help://MS.EXCEL.12.1033/EXCEL/content/HA10047504.htm) | Adds the cells in a range that meet multiple criteria |
| [SUMPRODUCT](ms-help://MS.EXCEL.12.1033/EXCEL/content/HP10062466.htm) | Returns the sum of the products of corresponding array components |
| [SUMSQ](ms-help://MS.EXCEL.12.1033/EXCEL/content/HP10062467.htm) | Returns the sum of the squares of the arguments |
| [SUMX2MY2](ms-help://MS.EXCEL.12.1033/EXCEL/content/HP10062468.htm) | Returns the sum of the difference of squares of corresponding values in two arrays |
| [SUMX2PY2](ms-help://MS.EXCEL.12.1033/EXCEL/content/HP10062469.htm) | Returns the sum of the sum of squares of corresponding values in two arrays |
| [SUMXMY2](ms-help://MS.EXCEL.12.1033/EXCEL/content/HP10062470.htm) | Returns the sum of squares of differences of corresponding values in two arrays |
| [TAN](ms-help://MS.EXCEL.12.1033/EXCEL/content/HP10062471.htm) | Returns the tangent of a number |
| [TANH](ms-help://MS.EXCEL.12.1033/EXCEL/content/HP10062472.htm) | Returns the hyperbolic tangent of a number |
| [TRUNC](ms-help://MS.EXCEL.12.1033/EXCEL/content/HP10062473.htm) | Truncates a number to an integer |

### Modul “Database”

|  |  |
| --- | --- |
| **Function** | **Description** |
| [DAVERAGE](ms-help://MS.EXCEL.12.1033/EXCEL/content/HP10062266.htm) | Returns the average of selected database entries |
| [DCOUNT](ms-help://MS.EXCEL.12.1033/EXCEL/content/HP10062267.htm) | Counts the cells that contain numbers in a database |
| [DCOUNTA](ms-help://MS.EXCEL.12.1033/EXCEL/content/HP10062268.htm) | Counts nonblank cells in a database |
| [DGET](ms-help://MS.EXCEL.12.1033/EXCEL/content/HP10062269.htm) | Extracts from a database a single record that matches the specified criteria |
| [DMAX](ms-help://MS.EXCEL.12.1033/EXCEL/content/HP10062270.htm) | Returns the maximum value from selected database entries |
| [DMIN](ms-help://MS.EXCEL.12.1033/EXCEL/content/HP10062271.htm) | Returns the minimum value from selected database entries |
| [DPRODUCT](ms-help://MS.EXCEL.12.1033/EXCEL/content/HP10062272.htm) | Multiplies the values in a particular field of records that match the criteria in a database |
| [DSTDEV](ms-help://MS.EXCEL.12.1033/EXCEL/content/HP10062273.htm) | Estimates the standard deviation based on a sample of selected database entries |
| [DSTDEVP](ms-help://MS.EXCEL.12.1033/EXCEL/content/HP10062274.htm) | Calculates the standard deviation based on the entire population of selected database entries |
| [DSUM](ms-help://MS.EXCEL.12.1033/EXCEL/content/HP10062275.htm) | Adds the numbers in the field column of records in the database that match the criteria |
| [DVAR](ms-help://MS.EXCEL.12.1033/EXCEL/content/HP10062276.htm) | Estimates variance based on a sample from selected database entries |
| [DVARP](ms-help://MS.EXCEL.12.1033/EXCEL/content/HP10062277.htm) | Calculates variance based on the entire population of selected database entries |

### Modul “Financial”

|  |  |
| --- | --- |
| **Function** | **Description** |
| [ACCRINT](ms-help://MS.EXCEL.12.1033/EXCEL/content/HP10062343.htm) | Returns the accrued interest for a security that pays periodic interest |
| [ACCRINTM](ms-help://MS.EXCEL.12.1033/EXCEL/content/HP10062344.htm) | Returns the accrued interest for a security that pays interest at maturity |
| [AMORDEGRC](ms-help://MS.EXCEL.12.1033/EXCEL/content/HP10062345.htm) | Returns the depreciation for each accounting period by using a depreciation coefficient |
| [AMORLINC](ms-help://MS.EXCEL.12.1033/EXCEL/content/HP10062346.htm) | Returns the depreciation for each accounting period |
| [COUPDAYBS](ms-help://MS.EXCEL.12.1033/EXCEL/content/HP10062347.htm) | Returns the number of days from the beginning of the coupon period to the settlement date |
| [COUPDAYS](ms-help://MS.EXCEL.12.1033/EXCEL/content/HP10062348.htm) | Returns the number of days in the coupon period that contains the settlement date |
| [COUPDAYSNC](ms-help://MS.EXCEL.12.1033/EXCEL/content/HP10062349.htm) | Returns the number of days from the settlement date to the next coupon date |
| [COUPNCD](ms-help://MS.EXCEL.12.1033/EXCEL/content/HP10062350.htm) | Returns the next coupon date after the settlement date |
| [COUPNUM](ms-help://MS.EXCEL.12.1033/EXCEL/content/HP10062351.htm) | Returns the number of coupons payable between the settlement date and maturity date |
| [COUPPCD](ms-help://MS.EXCEL.12.1033/EXCEL/content/HP10062352.htm) | Returns the previous coupon date before the settlement date |
| [CUMIPMT](ms-help://MS.EXCEL.12.1033/EXCEL/content/HP10062353.htm) | Returns the cumulative interest paid between two periods |
| [CUMPRINC](ms-help://MS.EXCEL.12.1033/EXCEL/content/HP10062354.htm) | Returns the cumulative principal paid on a loan between two periods |
| [DB](ms-help://MS.EXCEL.12.1033/EXCEL/content/HP10062355.htm) | Returns the depreciation of an asset for a specified period by using the fixed-declining balance method |
| [DDB](ms-help://MS.EXCEL.12.1033/EXCEL/content/HP10062356.htm) | Returns the depreciation of an asset for a specified period by using the double-declining balance method or some other method that you specify |
| [DISC](ms-help://MS.EXCEL.12.1033/EXCEL/content/HP10062357.htm) | Returns the discount rate for a security |
| [DOLLARDE](ms-help://MS.EXCEL.12.1033/EXCEL/content/HP10062358.htm) | Converts a dollar price, expressed as a fraction, into a dollar price, expressed as a decimal number |
| [DOLLARFR](ms-help://MS.EXCEL.12.1033/EXCEL/content/HP10062359.htm) | Converts a dollar price, expressed as a decimal number, into a dollar price, expressed as a fraction |
| [DURATION](ms-help://MS.EXCEL.12.1033/EXCEL/content/HP10062360.htm) | Returns the annual duration of a security with periodic interest payments |
| [EFFECT](ms-help://MS.EXCEL.12.1033/EXCEL/content/HP10062361.htm) | Returns the effective annual interest rate |
| [FV](ms-help://MS.EXCEL.12.1033/EXCEL/content/HP10069823.htm) | Returns the future value of an investment |
| [FVSCHEDULE](ms-help://MS.EXCEL.12.1033/EXCEL/content/HP10062362.htm) | Returns the future value of an initial principal after applying a series of compound interest rates |
| [INTRATE](ms-help://MS.EXCEL.12.1033/EXCEL/content/HP10062363.htm) | Returns the interest rate for a fully invested security |
| [IPMT](ms-help://MS.EXCEL.12.1033/EXCEL/content/HP10062364.htm) | Returns the interest payment for an investment for a given period |
| [IRR](ms-help://MS.EXCEL.12.1033/EXCEL/content/HP10062365.htm) | Returns the internal rate of return for a series of cash flows |
| [ISPMT](ms-help://MS.EXCEL.12.1033/EXCEL/content/HP10062366.htm) | Calculates the interest paid during a specific period of an investment |
| [MDURATION](ms-help://MS.EXCEL.12.1033/EXCEL/content/HP10062367.htm) | Returns the Macauley modified duration for a security with an assumed par value of $100 |
| [MIRR](ms-help://MS.EXCEL.12.1033/EXCEL/content/HP10062368.htm) | Returns the internal rate of return where positive and negative cash flows are financed at different rates |
| [NOMINAL](ms-help://MS.EXCEL.12.1033/EXCEL/content/HP10062369.htm) | Returns the annual nominal interest rate |
| [NPER](ms-help://MS.EXCEL.12.1033/EXCEL/content/HP10062370.htm) | Returns the number of periods for an investment |
| [NPV](ms-help://MS.EXCEL.12.1033/EXCEL/content/HP10069824.htm) | Returns the net present value of an investment based on a series of periodic cash flows and a discount rate |
| [ODDFPRICE](ms-help://MS.EXCEL.12.1033/EXCEL/content/HP10062371.htm) | Returns the price per $100 face value of a security with an odd first period |
| [ODDFYIELD](ms-help://MS.EXCEL.12.1033/EXCEL/content/HP10062372.htm) | Returns the yield of a security with an odd first period |
| [ODDLPRICE](ms-help://MS.EXCEL.12.1033/EXCEL/content/HP10062373.htm) | Returns the price per $100 face value of a security with an odd last period |
| [ODDLYIELD](ms-help://MS.EXCEL.12.1033/EXCEL/content/HP10062374.htm) | Returns the yield of a security with an odd last period |
| [PMT](ms-help://MS.EXCEL.12.1033/EXCEL/content/HP10069825.htm) | Returns the periodic payment for an annuity |
| [PPMT](ms-help://MS.EXCEL.12.1033/EXCEL/content/HP10069826.htm) | Returns the payment on the principal for an investment for a given period |
| [PRICE](ms-help://MS.EXCEL.12.1033/EXCEL/content/HP10062375.htm) | Returns the price per $100 face value of a security that pays periodic interest |
| [PRICEDISC](ms-help://MS.EXCEL.12.1033/EXCEL/content/HP10062376.htm) | Returns the price per $100 face value of a discounted security |
| [PRICEMAT](ms-help://MS.EXCEL.12.1033/EXCEL/content/HP10062377.htm) | Returns the price per $100 face value of a security that pays interest at maturity |
| [PV](ms-help://MS.EXCEL.12.1033/EXCEL/content/HP10062378.htm) | Returns the present value of an investment |
| [RATE](ms-help://MS.EXCEL.12.1033/EXCEL/content/HP10062379.htm) | Returns the interest rate per period of an annuity |
| [RECEIVED](ms-help://MS.EXCEL.12.1033/EXCEL/content/HP10062380.htm) | Returns the amount received at maturity for a fully invested security |
| [SLN](ms-help://MS.EXCEL.12.1033/EXCEL/content/HP10062381.htm) | Returns the straight-line depreciation of an asset for one period |
| [SYD](ms-help://MS.EXCEL.12.1033/EXCEL/content/HP10062382.htm) | Returns the sum-of-years' digits depreciation of an asset for a specified period |
| [TBILLEQ](ms-help://MS.EXCEL.12.1033/EXCEL/content/HP10062383.htm) | Returns the bond-equivalent yield for a Treasury bill |
| [TBILLPRICE](ms-help://MS.EXCEL.12.1033/EXCEL/content/HP10062384.htm) | Returns the price per $100 face value for a Treasury bill |
| [TBILLYIELD](ms-help://MS.EXCEL.12.1033/EXCEL/content/HP10062385.htm) | Returns the yield for a Treasury bill |
| [VDB](ms-help://MS.EXCEL.12.1033/EXCEL/content/HP10062386.htm) | Returns the depreciation of an asset for a specified or partial period by using a declining balance method |
| [XIRR](ms-help://MS.EXCEL.12.1033/EXCEL/content/HP10062387.htm) | Returns the internal rate of return for a schedule of cash flows that is not necessarily periodic |
| [XNPV](ms-help://MS.EXCEL.12.1033/EXCEL/content/HP10062388.htm) | Returns the net present value for a schedule of cash flows that is not necessarily periodic |
| [YIELD](ms-help://MS.EXCEL.12.1033/EXCEL/content/HP10062389.htm) | Returns the yield on a security that pays periodic interest |
| [YIELDDISC](ms-help://MS.EXCEL.12.1033/EXCEL/content/HP10062390.htm) | Returns the annual yield for a discounted security; for example, a Treasury bill |
| [YIELDMAT](ms-help://MS.EXCEL.12.1033/EXCEL/content/HP10062391.htm) | Returns the annual yield of a security that pays interest at maturity |

### Modul “Date & Time”

|  |  |
| --- | --- |
| **Function** | **Description** |
| [DATE](ms-help://MS.EXCEL.12.1033/EXCEL/content/HP10062283.htm) | Returns the serial number of a particular date |
| [DATEVALUE](ms-help://MS.EXCEL.12.1033/EXCEL/content/HP10062284.htm) | Converts a date in the form of text to a serial number |
| [DAY](ms-help://MS.EXCEL.12.1033/EXCEL/content/HP10062285.htm) | Converts a serial number to a day of the month |
| [DAYS360](ms-help://MS.EXCEL.12.1033/EXCEL/content/HP10062286.htm) | Calculates the number of days between two dates based on a 360-day year |
| [EDATE](ms-help://MS.EXCEL.12.1033/EXCEL/content/HP10062287.htm) | Returns the serial number of the date that is the indicated number of months before or after the start date |
| [EOMONTH](ms-help://MS.EXCEL.12.1033/EXCEL/content/HP10062288.htm) | Returns the serial number of the last day of the month before or after a specified number of months |
| [HOUR](ms-help://MS.EXCEL.12.1033/EXCEL/content/HP10062289.htm) | Converts a serial number to an hour |
| [MINUTE](ms-help://MS.EXCEL.12.1033/EXCEL/content/HP10062290.htm) | Converts a serial number to a minute |
| [MONTH](ms-help://MS.EXCEL.12.1033/EXCEL/content/HP10062291.htm) | Converts a serial number to a month |
| [NETWORKDAYS](ms-help://MS.EXCEL.12.1033/EXCEL/content/HP10062292.htm) | Returns the number of whole workdays between two dates |
| [NOW](ms-help://MS.EXCEL.12.1033/EXCEL/content/HP10062293.htm) | Returns the serial number of the current date and time |
| [SECOND](ms-help://MS.EXCEL.12.1033/EXCEL/content/HP10062294.htm) | Converts a serial number to a second |
| [TIME](ms-help://MS.EXCEL.12.1033/EXCEL/content/HP10062295.htm) | Returns the serial number of a particular time |
| [TIMEVALUE](ms-help://MS.EXCEL.12.1033/EXCEL/content/HP10062296.htm) | Converts a time in the form of text to a serial number |
| [TODAY](ms-help://MS.EXCEL.12.1033/EXCEL/content/HP10062297.htm) | Returns the serial number of today's date |
| [WEEKDAY](ms-help://MS.EXCEL.12.1033/EXCEL/content/HP10062298.htm) | Converts a serial number to a day of the week |
| [WEEKNUM](ms-help://MS.EXCEL.12.1033/EXCEL/content/HP10062299.htm) | Converts a serial number to a number representing where the week falls numerically with a year |
| [WORKDAY](ms-help://MS.EXCEL.12.1033/EXCEL/content/HP10062300.htm) | Returns the serial number of the date before or after a specified number of workdays |
| [YEAR](ms-help://MS.EXCEL.12.1033/EXCEL/content/HP10062301.htm) | Converts a serial number to a year |
| [YEARFRAC](ms-help://MS.EXCEL.12.1033/EXCEL/content/HP10062302.htm) | Returns the year fraction representing the number of whole days between start\_date and end\_date |

### Modul “Information”

|  |  |
| --- | --- |
| **Function** | **Description** |
| [CELL](ms-help://MS.EXCEL.12.1033/EXCEL/content/HP10062392.htm) | Returns information about the formatting, location, or contents of a cell |
| [ERROR.TYPE](ms-help://MS.EXCEL.12.1033/EXCEL/content/HP10062394.htm) | Returns a number corresponding to an error type |
| [INFO](ms-help://MS.EXCEL.12.1033/EXCEL/content/HP10062395.htm) | Returns information about the current operating environment |
| [ISBLANK](ms-help://MS.EXCEL.12.1033/EXCEL/content/HP10069827.htm) | Returns TRUE if the value is blank |
| [ISERR](ms-help://MS.EXCEL.12.1033/EXCEL/content/HP10069827.htm) | Returns TRUE if the value is any error value except #N/A |
| [ISERROR](ms-help://MS.EXCEL.12.1033/EXCEL/content/HP10069827.htm) | Returns TRUE if the value is any error value |
| [ISEVEN](ms-help://MS.EXCEL.12.1033/EXCEL/content/HP10062396.htm) | Returns TRUE if the number is even |
| [ISLOGICAL](ms-help://MS.EXCEL.12.1033/EXCEL/content/HP10069827.htm) | Returns TRUE if the value is a logical value |
| [ISNA](ms-help://MS.EXCEL.12.1033/EXCEL/content/HP10069827.htm) | Returns TRUE if the value is the #N/A error value |
| [ISNONTEXT](ms-help://MS.EXCEL.12.1033/EXCEL/content/HP10069827.htm) | Returns TRUE if the value is not text |
| [ISNUMBER](ms-help://MS.EXCEL.12.1033/EXCEL/content/HP10069827.htm) | Returns TRUE if the value is a number |
| [ISODD](ms-help://MS.EXCEL.12.1033/EXCEL/content/HP10062397.htm) | Returns TRUE if the number is odd |
| [ISREF](ms-help://MS.EXCEL.12.1033/EXCEL/content/HP10069827.htm) | Returns TRUE if the value is a reference |
| [ISTEXT](ms-help://MS.EXCEL.12.1033/EXCEL/content/HP10069827.htm) | Returns TRUE if the value is text |
| [N](ms-help://MS.EXCEL.12.1033/EXCEL/content/HP10062398.htm) | Returns a value converted to a number |
| [NA](ms-help://MS.EXCEL.12.1033/EXCEL/content/HP10062399.htm) | Returns the error value #N/A |
| [TYPE](ms-help://MS.EXCEL.12.1033/EXCEL/content/HP10062400.htm) | Returns a number indicating the data type of a value |

### Modul “Engineering”

|  |  |
| --- | --- |
| **Function** | **Description** |
| [BESSELI](ms-help://MS.EXCEL.12.1033/EXCEL/content/HP10062303.htm) | Returns the modified Bessel function In(x) |
| [BESSELJ](ms-help://MS.EXCEL.12.1033/EXCEL/content/HP10062304.htm) | Returns the Bessel function Jn(x) |
| [BESSELK](ms-help://MS.EXCEL.12.1033/EXCEL/content/HP10062305.htm) | Returns the modified Bessel function Kn(x) |
| [BESSELY](ms-help://MS.EXCEL.12.1033/EXCEL/content/HP10062306.htm) | Returns the Bessel function Yn(x) |
| [BIN2DEC](ms-help://MS.EXCEL.12.1033/EXCEL/content/HP10062307.htm) | Converts a binary number to decimal |
| [BIN2HEX](ms-help://MS.EXCEL.12.1033/EXCEL/content/HP10062308.htm) | Converts a binary number to hexadecimal |
| [BIN2OCT](ms-help://MS.EXCEL.12.1033/EXCEL/content/HP10062309.htm) | Converts a binary number to octal |
| [COMPLEX](ms-help://MS.EXCEL.12.1033/EXCEL/content/HP10062310.htm) | Converts real and imaginary coefficients into a complex number |
| [CONVERT](ms-help://MS.EXCEL.12.1033/EXCEL/content/HP10062311.htm) | Converts a number from one measurement system to another |
| [DEC2BIN](ms-help://MS.EXCEL.12.1033/EXCEL/content/HP10062312.htm) | Converts a decimal number to binary |
| [DEC2HEX](ms-help://MS.EXCEL.12.1033/EXCEL/content/HP10062313.htm) | Converts a decimal number to hexadecimal |
| [DEC2OCT](ms-help://MS.EXCEL.12.1033/EXCEL/content/HP10062314.htm) | Converts a decimal number to octal |
| [DELTA](ms-help://MS.EXCEL.12.1033/EXCEL/content/HP10062315.htm) | Tests whether two values are equal |
| [ERF](ms-help://MS.EXCEL.12.1033/EXCEL/content/HP10062316.htm) | Returns the error function |
| [ERFC](ms-help://MS.EXCEL.12.1033/EXCEL/content/HP10062317.htm) | Returns the complementary error function |
| [GESTEP](ms-help://MS.EXCEL.12.1033/EXCEL/content/HP10062318.htm) | Tests whether a number is greater than a threshold value |
| [HEX2BIN](ms-help://MS.EXCEL.12.1033/EXCEL/content/HP10062319.htm) | Converts a hexadecimal number to binary |
| [HEX2DEC](ms-help://MS.EXCEL.12.1033/EXCEL/content/HP10062321.htm) | Converts a hexadecimal number to decimal |
| [HEX2OCT](ms-help://MS.EXCEL.12.1033/EXCEL/content/HP10062322.htm) | Converts a hexadecimal number to octal |
| [IMABS](ms-help://MS.EXCEL.12.1033/EXCEL/content/HP10062323.htm) | Returns the absolute value (modulus) of a complex number |
| [IMAGINARY](ms-help://MS.EXCEL.12.1033/EXCEL/content/HP10062324.htm) | Returns the imaginary coefficient of a complex number |
| [IMARGUMENT](ms-help://MS.EXCEL.12.1033/EXCEL/content/HP10062325.htm) | Returns the argument theta, an angle expressed in radians |
| [IMCONJUGATE](ms-help://MS.EXCEL.12.1033/EXCEL/content/HP10062326.htm) | Returns the complex conjugate of a complex number |
| [IMCOS](ms-help://MS.EXCEL.12.1033/EXCEL/content/HP10062327.htm) | Returns the cosine of a complex number |
| [IMDIV](ms-help://MS.EXCEL.12.1033/EXCEL/content/HP10062328.htm) | Returns the quotient of two complex numbers |
| [IMEXP](ms-help://MS.EXCEL.12.1033/EXCEL/content/HP10062329.htm) | Returns the exponential of a complex number |
| [IMLN](ms-help://MS.EXCEL.12.1033/EXCEL/content/HP10062330.htm) | Returns the natural logarithm of a complex number |
| [IMLOG10](ms-help://MS.EXCEL.12.1033/EXCEL/content/HP10062331.htm) | Returns the base-10 logarithm of a complex number |
| [IMLOG2](ms-help://MS.EXCEL.12.1033/EXCEL/content/HP10062332.htm) | Returns the base-2 logarithm of a complex number |
| [IMPOWER](ms-help://MS.EXCEL.12.1033/EXCEL/content/HP10062333.htm) | Returns a complex number raised to an integer power |
| [IMPRODUCT](ms-help://MS.EXCEL.12.1033/EXCEL/content/HP10062334.htm) | Returns the product of from 2 to 29 complex numbers |
| [IMREAL](ms-help://MS.EXCEL.12.1033/EXCEL/content/HP10062335.htm) | Returns the real coefficient of a complex number |
| [IMSIN](ms-help://MS.EXCEL.12.1033/EXCEL/content/HP10062336.htm) | Returns the sine of a complex number |
| [IMSQRT](ms-help://MS.EXCEL.12.1033/EXCEL/content/HP10062337.htm) | Returns the square root of a complex number |
| [IMSUB](ms-help://MS.EXCEL.12.1033/EXCEL/content/HP10062338.htm) | Returns the difference between two complex numbers |
| [IMSUM](ms-help://MS.EXCEL.12.1033/EXCEL/content/HP10062339.htm) | Returns the sum of complex numbers |
| [OCT2BIN](ms-help://MS.EXCEL.12.1033/EXCEL/content/HP10062340.htm) | Converts an octal number to binary |
| [OCT2DEC](ms-help://MS.EXCEL.12.1033/EXCEL/content/HP10062341.htm) | Converts an octal number to decimal |
| [OCT2HEX](ms-help://MS.EXCEL.12.1033/EXCEL/content/HP10062342.htm) | Converts an octal number to hexadecimal |

### Modul “Cube”

|  |  |
| --- | --- |
| **Function** | **Description** |
| [CUBEKPIMEMBER](ms-help://MS.EXCEL.12.1033/EXCEL/content/HA10083021.htm) | Returns a key performance indicator (KPI) property and displays the KPI name in the cell. A KPI is a quantifiable measurement, such as monthly gross profit or quarterly employee turnover, that is used to monitor an organization's performance. |
| [CUBEMEMBER](ms-help://MS.EXCEL.12.1033/EXCEL/content/HA10083017.htm) | Returns a member or tuple from the cube. Use to validate that the member or tuple exists in the cube. |
| [CUBEMEMBERPROPERTY](ms-help://MS.EXCEL.12.1033/EXCEL/content/HA10083023.htm) | Returns the value of a member property from the cube. Use to validate that a member name exists within the cube and to return the specified property for this member. |
| [CUBERANKEDMEMBER](ms-help://MS.EXCEL.12.1033/EXCEL/content/HA10083020.htm) | Returns the nth, or ranked, member in a set. Use to return one or more elements in a set, such as the top sales performer or the top 10 students. |
| [CUBESET](ms-help://MS.EXCEL.12.1033/EXCEL/content/HA10083019.htm) | Defines a calculated set of members or tuples by sending a set expression to the cube on the server, which creates the set, and then returns that set to Microsoft Office Excel. |
| [CUBESETCOUNT](ms-help://MS.EXCEL.12.1033/EXCEL/content/HA10083024.htm) | Returns the number of items in a set. |
| [CUBEVALUE](ms-help://MS.EXCEL.12.1033/EXCEL/content/HA10083018.htm) | Returns an aggregated value from the cube. |

## SOAL-SOAL LATIHAN.

1. Buatlah algoritma program modular, yang didalamnya terdapat subprogram yang berupa **prosedur** untuk memproses :

Hasil = 1+2+3+4+5+ … +N, dimana N sebagai parameter

Procedure Jumlah1N(in N : Integer; out Hasil : Integer);

{I.S. : N integer >= 1, jika N < 1 maka error }

{F.S. : Hasil = 1 + 2 + 3 + 4 + 5 + … + N }

1. Buatlah algoritma program modular, yang didalamnya terdapat subprogram yang berupa **fungsi** untuk menghitung :

Hasil = 1+2+3+4+5+ … + N, dimana N sebagai parameter

Function HitungJumlah1N(in N : Integer): Integer;

{Diberikan N integer >= 1, jika N < 1 maka error }

{Dikirim nilai hasil = 1 + 2 + 3 + 4 + 5 + … + N }

1. Buatlah algoritma program modular, yang didalamnya terdapat subprogram yang berupa **prosedur** dan juga **fungsi** ( inisiatif sendiri) untuk memproses :

Hasil = 1x2x3x4x5x … x N, dimana N sebagai parameter

1. Buatlah algoritma program modular, yang didalamnya terdapat subprogram yang berupa **prosedur** untuk mencetak label/tulisan/teks ditengah (center) sepanjang area yang ditentukan (panjang satuan karakter) dan ditulis mulai koordinat monitor (X,Y). Tulisan yang dicetak berwarna FC pada latarbelakang warna BC.

Procedure PrintTeks(in X,Y,L,FC,FB : Integer; in T : String);

{I.S. : X,Y koordinat layar, L panjang area maks 50 dari }

{ tulisan T dengan warna FC dan latar belakang BG }

{F.S. : Tulisan T tercetak ditengah area L sesuai kriteria I.S }

1. Buatlah algoritma program modular, yang didalamnya terdapat subprogram yang berupa **prosedur** atau **fungsi** (bebas menurut pilihan Anda) untuk mencetak tulisan menjadi terbalik ejaannya. Misal AYAH menjadi HAYA, SELAMAT menjadi TAMALES, dst.
2. Buatlah algoritma program modular, yang didalamnya terdapat subprogram yang berupa **prosedur** atau **fungsi** (bebas menurut pilihan Anda) yang dapat menentukan banyaknya kata dalam kalimat yang diketikkan (titik dan tanda baca bukan kata). Misal “ayah pergi ke kantor, dan Ibu pergi ke pasar.” ada 9 kata.
3. Buatlah algoritma program modular, yang didalamnya terdapat subprogram yang berupa **prosedur** atau **fungsi** (bebas menurut pilihan Anda) yang dapat menentukan banyaknya huruf dalam sebuah kata yang diketikkan.
4. Buatlah algoritma program modular, yang dapat menentukan mencetak semua bilangan prima dari 1 sampai dengan N, dimana N parameter yang diinginkan.
5. Buatlah algoritma program modular, yang dapat menentukan mencetak semua bilangan prima dari 1 sampai dengan N, dimana N parameter yang diinginkan.
6. Buatlah algoritma program modular, yang dapat menentukan mencetak N bilangan fibbonacci, dimana N parameter banyaknya bilangan yang diinginkan.
7. Buatlah algoritma program modular, yang dapat membangkitkan bilangan random integer antara 1 dan 100 sebanyak 500 buah.

# SKEMA ALGORITMA PROSEDUR DAN FUNGSI MODEL REKURSIF

## Pengertian

Susuatu dikatakan rekurrsif jika definisi sesuatu tersebut mengandung terminologi dirinya. Algoritma prosedur atau fungsi disebut rekursif jika di dalam pendefinisian prosedur atau fungsi tersebut mengandung pemanggilan nama prosedur/fungsi dirinya.

Kelebihan dari algoritma rekursif adalah sintaksnya yang sederhana dan mudah dibaca, sedangkan kelemahannya adlah memerlukan waktu yang eksekusi yang relatif lama dibandingkan versi iteratif (pengulangan standar). Prinsip algoritma rekursif mengandung dua bagian, yaitu bagian dasar (*Base*) dan bagian pengulangan (*Reccurent=Recc*), yang didefiniskan sebagai berikut :

* Base : merupakan kondisi yang menjadi terminal pengulangan dan dasar perhitungan.
* Recc : merupakan bagian yang mendefinisikan proses pengulangan.

**Contoh : 1.** Problem Perhitungan Faktorial

Analsis

Problem : Faktorial(N) = N ! = N x (N-1) x (N-2) x … x 1

Base : Faktorial (0) = 1

Recc : Faktorial (N) = N x Faktorial (N-1)

Konstruksi : If (N=0) Then

Eksekusi (Base)

Else

Eksekusi (Recc)

Algoritma : 1. Pendefinisian Fungsi rekursif

**Function** Faktorial(N : Integer)🡪 Integer;

{diberikan integer N >= 0 }

{dikirimkan hasil N! }

Kamus : -

Algoritma

If (N=0) Then

🡪 (1)

Else

🡪 N \* Faktorial(N-1)

Endif

End

2. Pendefinisian Prosedur rekursif

**Procedure** Faktorial(out H:Integer; in N:Integer);

{I.S.: sembarang integer N >= 0 }

{F.S.: H = N! }

Kamus : -

Algoritma

If (N=0) Then

H 🡨 1

Else

**Call** Faktorial(H, N-1)

Endif

End

**Contoh : 2.** Problem Bilangan Fibonacci

Analsis

Problem : Fibonacci(N) = 1,2,3,5,8,13, ..., Fibonacci ke-N

Base : Fibonacci(0) = 1

Fibonacci(1) = 1

Recc : Fibonacci(N) = Fibonacci(N-1)+ Fibonacci(N-2)

Konstruksi : Depend On N

0 : Eksekusi(Base0)

1 : Eksekusi(Base1)

N>1 : Eksekusi(Recc)

EndDepend

Algoritma : 1. Pendefinisian Fungsi rekursif

**Function** Fibonacci(N : Integer)🡪 Integer;

{diberikan integer N >= 0}

{dikirimkan nilai bilangan Fibonacci ke-N}

Kamus : -

Algoritma

Depend On N

0 : 🡪 (1)

1 : 🡪 (1)

N>1 : 🡪 Fibonacci(N-1)+Fibonacci(N-2)

EndDepend

End

2. Pendefinisian Prosedur rekursif

**Procedure** Fibonaccci(out H:Integer; in N:Integer);

{I.S.: sembarang integer N >= 0}

{F.S.: H = bilangan Fibonacii ke-N}

Kamus : -

H1, H2 : Integer;

Algoritma

Depend On N

0 : H 🡨 1

1 : H 🡨 1

N>1 : Call Fibonacci(H, N-1)

H1 🡨 H

Call Fibonacci(H, N-2)

H2 🡨 H

H 🡨 H1 + H2

EndDepend

End

## Contoh Implementasi Turbo Pascal

Implementasi program rekursif dari dua problem yang telah diuraikan sebelumnya ke dalam Turbo Pascal diuraikan sebagai berikut :

**A. Problem Faktorial**

{\* ===== JUDUL PROBLEM ===== \*}

**Program** AlgoritmaRekursif\_Procedure\_Function\_Untuk\_Faktorial;

{\* input : N ditentukan = 10 \*}

{\* output : N ! \*}

{\* proses : Procedure dan Function Rekursif \*}

{\* ===== KAMUS GLOBAL ===== \*}

{\* --- Deklarasi Link --- \*}

**Uses** Crt;

{\* --- Deklarasi Data --- \*}

**Type** DataX = Integer;

**Var** N,H,i : DataX;

{\* --- Deklarasi Prosedur/Fungsi --- \*}

**Procedure** PFaktorial(Var H:DataX; N:DataX);

{I.S.: sembarang integer N >= 0 }

{F.S.: H = N! }

begin

If (N=0) then

H:=1

else begin

PFaktorial(H,N-1);

H := H \* N;

end;

end;

**Function** FFaktorial(N:DataX):DataX;

{diberikan integer N >= 0 }

{dikirimkan hasil N! }

begin

If (N=0) then

FFaktorial:=1

else

FFaktorial:=N\*FFaktorial(N-1);

end;

{\* ===== ALGORITMA UTAMA ===== \*}

Begin

Clrscr;

N:=10;

Writeln('------------------------------');

Writeln('> Output PROCEDURE FAKTORIAL :');

PFaktorial(H,N);

Writeln(N,'! = ',H);

Writeln('------------------------------');

Writeln('> Output FUNCTION FAKTORIAL :');

Writeln(N,'! = ',FFaktorial(N));

readln;

end.

**B. Problem Fibonacci**

{\* ===== JUDUL PROBLEM ===== \*}

**Program** AlgoritmaRekursif\_Procedure\_Function\_Untuk\_Fibonacci;

{\* input : N ditentukan =10 \*}

{\* output : tercetak deret bilangan fibonacci sampai bilangan ke-N\*}

{\* proses : Procedure dan Function Rekursif \*}

{\* ===== KAMUS GLOBAL ===== \*}

{\* --- Deklarasi Link --- \*}

**Uses** Crt;

{\* --- Deklarasi Data --- \*}

**Type** DataX = Integer;

**Var** N,H,i : DataX;

{\* --- Deklarasi Prosedur/Fungsi --- \*}

**Procedure** PFibonacci(var H:DataX; N:DataX);

{I.S.: sembarang integer N >= 0}

{F.S.: H = bilangan Fibonacii ke-N}

var H1,H2:DataX;

begin

Case N Of

0 : H:=1;

1 : H:=1;

else begin

PFibonacci(H,N-1); H1:=H;

PFibonacci(H,N-2); H2:=H;

H:=H1+H2;

end;

end;

end;

**Function** FFibonacci(N:DataX):DataX;

{diberikan integer N >= 0}

{dikirimkan nilai bilangan Fibonacci ke-N}

begin

Case N Of

0 : FFibonacci:=1;

1 : FFibonacci:=1;

else begin

FFibonacci:= FFibonacci(N-1)+FFibonacci(N-2);

end;

end;

end;

{\* ===== ALGORITMA UTAMA ===== \*}

Begin

Clrscr;

N:=10;

Writeln('------------------------------');

Writeln('> Output PROCEDURE FIBONACCI :');

For i:= 1 To N do begin

PFibonacci(H,i);

Write(H,',');

end;

Writeln('------------------------------');

Writeln('> Output FUNCTION FIBONACCI :');

For i:= 1 To N do begin

Write(FFibonacci(i),',');

end;

readln;

end.

**SOAL-SOAL LATIHAN.**

1. Buatlah algoritma program modular, yang didalamnya terdapat subprogram yang berupa **Prosedur dan Fungsi Rekursif** untuk mencetak Nama Anda di monitor sebanyak N kali, dimana N diinput dari keyboard !
2. Buatlah algoritma program modular, yang didalamnya terdapat subprogram yang berupa **prosedur** dan juga **fungsi Rekursif** ( inisiatif sendiri) untuk mencetak sederetan bilangan dari 1 s/d N, N diinput!
3. Buatlah algoritma program modular, yang didalamnya terdapat subprogram yang berupa **prosedur** dan juga **fungsi Rekursif** ( inisiatif sendiri) untuk mencetak sederetan bilangan ganjil dari 1 s/d N, N diinput!
4. Buatlah algoritma program modular, yang didalamnya terdapat subprogram yang berupa **prosedur** dan juga **fungsi Rekursif** ( inisiatif sendiri) untuk mencetak sederetan bilangan kelipatan 10 dari 1 s/d 150.000!
5. Buatlah algoritma program modular, yang didalamnya terdapat subprogram yang berupa **Prosedur Rekursif** untuk memproses :

Hasil = 1 + 2 + 3 + 4 + 5 + … + N, dimana N sebagai parameter

Procedure ProsesHasil (in N : Integer; out Hasil : Integer);

{I.S. : N integer >= 1, jika N < 1 maka error }

{F.S. : Hasil = 1 + 2 + 3 + 4 + 5 + … + N }

1. Buatlah algoritma program modular, yang didalamnya terdapat subprogram yang berupa **Fungsi Rekursif** untuk menghitung :

Hasil = 1 + 2 + 3 + 4 + 5 + … + N, dimana N sebagai parameter

Function HitungHasil(in N : Integer): Integer;

{Diberikan N integer >= 1, jika N < 1 maka error }

{Dikirim nilai hasil = 1 + 2 + 3 + 4 + 5 + … + N }

# ALGORITMA PEMROSESAN FILE SEKUENSIAL DAN FILE ACAK

## File Sekuensial

### Definisi File Sekuensial

**File (arsip, berkas) sekuensial** adalah sekumpulan rekaman elemen-elemen data sejenis (bertipe sama) yang disimpan dalam *storage (disk)* komputer dengan nama file tertentu, yang hanya dapat diakses secara berurutan dan searah (sekuensial). Akses berurutan dan searah berarti bahwa elemen data hanya dapat diakses satu per satu ”bergerak ke depan” (*next*) satu elemen, dan tidak dapat ”bergerak mundur” (*prev*). Rekaman elemen-elemen data disimpan mulai dari data pertama sampai dengan data terakhir. Pada beberapa implementasi, secara *bulit-in* akhir sebuah file ditambahkan fungsi rekaman data fiktif sebagai MARK akhir file yang umum diberi nama EOF (End Of File).

Rekaman elemen data dapat berisi tipe data sederhana (integer, real, char, string, boolean) maupun tipe data kompleks (record). Untuk membedakan setiap elemen data satu dengan lainnya dalam sebuah file agar tidak terjadi duplikasi, diciptakan suatu konsep yang disebut KEY. *Key* adalah satu atau beberapa atribut (field) elemen data yang digunakan untuk menjamin keunikan data (tidak ada duplikasi data). Diantara *key* yang dipilih untuk menjadi kunci utama dalam perekaman data disebut PRIMARY KEY.

Ilustrasi file sekuensial dengan sejumlah N elemen data beserta contoh *file integer, file character,* dan *file record* tentang nilai mahasiswa diperlihatkan dalam gambar di bawah ini.

60

40

50

10

30

90

20

10

70

80

**EOF**

File Of Integer

elemen data-1

elemen data-2

elemen data-3

elemen data-4

elemen data-5

elemen data-6

**...**

elemen data-N

**EOF**

File Sekuensial

C

A

B

X

S

A

C

Z

B

D

**EOF**

File Of Char

001, Ali, 60

002, Budi, 40

003, Cika, 50

004, Dedy, 10

005, Eva, 30

006, Ferry, 90

007, Gun Gun, 20

008, Heriawan, 10

009, Iwan, 70

010, Jejaka, 80

**EOF**

File Of Record <NIM, NAMA, NILAI>

Cara mendefinisikan kamus tipe, variabel, dan cara akses data dari contoh-contoh file tersebut adalah :

Type FileSequential = File Of Element

Element = ... // tipe standar atau kompleks

Var F : FileSequential

X : Element

-------------------------------------------------------------

Type FileSequential = File Of Element

Element = Integer

Var F : FileSequential

X : Element

-------------------------------------------------------------

Type FileSequential = File Of Element

Element = Character

Var F : FileSequential

X : Element

-------------------------------------------------------------

Type FileSequential = File Of Element

Element = < NIM : String[3];

NAMA : String[50];

NILAI : Integer >

Var F : FileSequential

X : Element

-------------------------------------------------------------

Berdasarkan fungsi akses, file sekuensial dibedakan menjadi tiga jenis yaitu :

1. INPUT

file sekuensial jenis input adalah file yang elemennya bersifat *read only* dan hanya digunakan untuk input data saja. Instruksi algoritma yang terkait dengan file ini adalah **Input(F,X)**

1. OUTPUT,

file sekuensial jenis output adalah file yang hanya digunakan untuk merekam elemen data saja sehingga bersifat *write only* Instruksi algoritma yang terkait dengan file ini adalah **Output(F,X)**

1. INPUT-OUTPUT

file sekuensial jenis input-output adalah file yang bersifat *read-write* dan umumnya digunakan untuk memproses transaksi data. Instruksi algoritma yang terkait dengan file ini adalah **Input(F,X)** maupun **Output(F,X).**

### Prosedur dan Fungsi Primitif

Prosedur dan fungsi primitifadalah sekumpulan prosedur dan atau fungsi yang terdefinisi dan diasumsikan telah disediakan (*built-in*) oleh bahasa pemrograman. Selanjutnya modul yang menyimpan sekumpulan prosedur dan fungsi primitif ini disebut PRIMITIF saja. Primitif ini dapat kita gunakan sebagai pengkayaan instruksi algoritma dalam pengelolaan file sekuensial. Beberapa primitif yang akan digunakan didefinisikan sebagai berikut :

|  |
| --- |
| **Procedure Assign(in/out vF : VarFile, in nmF : namaFile)**  {I.S. : vF sembarang, nmF sembarang nama fisik di disk }  {F.S. : variabel vF terhubung dengan path\namafile nmF di disk } |
| **Procedure Rewrite(in/out vF : VarFile)**  {I.S. : vF sudah terdefinisi dalam procedure Assign }  {F.S. : file dengan namafile nmF yang terdefinisi dalam procedure  Assign terbentuk dan dalam keaadan aktif/*open* di disk } |
| **Procedure OpenFile(in/out vF : VarFile)**  {I.S. : vF sudah pernah ada/ terdefinisi oleh procedure Rewrite}  {F.S. : file yang terkait dengan vF telah dibuka siap digunakan} |
| **Procedure CloseFile(in/out vF : VarFile)**  {I.S. : vF terbuka oleh procedure OpenFile }  {F.S. : file yang terkait dengan vF telah ditutup } |
| **Procedure Reset(in/out vF : VarFile)**  {I.S. : vF sudah perna terdefinisi dalam procedure OpenFile }  {F.S. : head pembaca data menunjuk lokasi/posisi di awal data } |
| **Function EOF(vF: VarFile)--> boolean**  { diberikan variabel file vF yang terdefinisi}  { dikrimkan status akhir file dengan nilai= TRUE jika head pembaca  data menunjuk MARK akhir file vF, dan nilai = FALSE jika bukan} |
| **Function IOResult(vF: VarFile)--> integer**  { diberikan variabel file vF }  { dikrimkan status keberadaan file dengan nilai = 0 jika file  telah ada dalam disk, dan nilai selain 0 jika file tidak ada} |
| **Prosedur dan fungsi khusus untuk TEXTFILE = File Of Char/String:**  **Procedure Append(in/out vF : VarFile)**  {I.S. : vF terdefinisi }  {F.S. : file dari vF terbuka/open dan head penulis data pada  posisi akhir data yang siap untuk penulisan data baru  yang akan ditambahkan} |

### Algoritma Pengolelolaan File Sekuensial

**A. Algoritma mendefinisikan sebuah file sekuensial**

|  |
| --- |
| **JUDUL : PROGRAM Membuat\_FileSekuensial;**  **KAMUS :**  Link : PRIMITIF  Const NamaFileMhs = ’D:\Data\filemhs.dat’  Type FileSequential = File Of Element  Element = String //atau element = char menjadi textfile  Var F : FileSequential  X : Element  **AGORITMA**  Assign(F,NamaFileMhs)  Rewrite(F)  CloseFile(F)  **END.** |

**B. Algoritma merekam elemen data sebuah file sekuensial**

|  |
| --- |
| **JUDUL : PROGRAM Membuat\_FileSekuensial;**  **{ Membuat file baru sekaligus merekam elemen data integer, Mark=999 }**  **KAMUS :**  Link : PRIMITIF  Const NamaFileMhs = ’D:\Data\filemhs.dat’  MARK = 999  Type FileSequential = File Of Element  Element = Integer  NamaFile = String  Var F : FileSequential  X : Element  STOP : Boolean  **Procedure CreateFile(in vF : NamaFile)**  **Kamus: -**  **Algoritma**  Assign(F,vF)  If (IOResult <> 0 ) then  Rewrite(F)  Else  Reset(F)  endif  **end;**  **AGORITMA**  Call CreateFile(NamaFileMhs)  STOP <-- FALSE  Call Append(F)  Repeat  Input(Keyboard,’Elemen data = ’,X)  If (X=MARK) then STOP <-- TRUE else Output(F,X)  Until (STOP)  CloseFile(F)  **END.** |

**C. Algoritma M*erging* dua buah file sekuensial**

Merging adalah penggabungan elemen-elemen data dari dua buah file sekuensial yang sejenis (bertipe sama) sehingga membentuk file hasil penggabungan. Penggabungan yang dimaksud disini adalah penggabungan yang paling sederhana yaitu file-1 ditambahkan atau ”dikonkatenasi” dengan file-2. Ilustrasi mergin tersebut digambarkan sebagai berikut :

60

40

50

10

30

90

20

10

70

80

**EOF**

File-1

500

400

200

300

100

**EOF**

File-2

60

40

50

10

30

90

20

10

70

80

500

400

200

300

100

**EOF**

File-3

|  |
| --- |
| **JUDUL : PROGRAM Merging\_Dua\_FileSekuensial;**  **{File-3 = File-1 dikokatenasi dengan File-2, file sudah terdefinisi}**  **KAMUS :**  Link : PRIMITIF  Type FileSequential = File Of Element  Element = Integer  Var File1,File2,File3 : FileSequential  **Procedure MergingFile(in F1,F2; out F3: FileSequential)**  **Kamus:** X : Element  **Algoritma**  **Call Append**(F3)  Input(F1,X)  While (X<> EOF(F1)) do  Output(F3,X)  Input(F1,X)  endwhile  Input(F2,X)  While (X<> EOF(F2)) do  Output(F3,X)  Input(F2,X)  endwhile  **end;**  **AGORITMA**  OpenFile(File1); OpenFile(File2); Rewrite(File3)  Call MergingFile(File1,File2,File3)  CloseFile(File1); CloseFile(File2); CloseFile(File3)  **END.** |

**D. Algoritma *Merging* dua buah file sekuensial yang terurut**

Merging jenis ini akan menggabung dua buah file sekuensial dengan data masing-masing sudah terurut dan akan membentuk file hasil penggabungan yang terurut pula. Penggabungan yang dimaksud yaitu file-1 digabung dengan file-2 menjadi file-3 dalam keadaan terurut seperti yang diilustrasi pada gambar di bawah.

Algoritma tidak dibahas disini, agar digunakan sebagai latihan.

90

80

70

60

50

40

30

20

10

10

**EOF**

File-1

50

40

20

30

10

**EOF**

File-2

90

80

70

60

50

50

40

40

30

30

20

20

10

10

10

**EOF**

File-3

## File Acak

### Definisi File Acak

**File (arsip, berkas) Acak** adalah sekumpulan rekaman elemen-elemen data sejenis (bertipe sama) yang disimpan dalam *storage (disk)* komputer dengan nama file tertentu, yang hanya dapat diakses secara acak (random) berdasarkan perhitungan terhadap lokasi data tertentu.

Primitif tambahan yang diperlukan meliputi :

|  |
| --- |
| **Procedure Seek(**in **vF: VarFile;** in **nR :NomorRecord)**  {I.S. : vF sudah terdefinisi, nR sembarang }  {F.S. : Head data aktif menunjuk data di alamat/nomor record nR}  **Function FilePos(**in **vF: VarFile)--> NomorRecord**  {I.S. : vF sudah terdefinisi }  {F.S. : nomor record yang aktif, record ke-1 memiliki nomor 0 }  **Function FileSize(**in **vF: VarFile)--> NomorRecord**  {I.S. : vF sudah terdefinisi }  {F.S. : Head data aktif menunjuk data nomor record terakhir + 1} |

### Algoritma Pengolahan Data

Sebagai latihan pelajari program di bawah ini.

TULISKAN KEMBALI ALGORITMA INI DAN AMATI PROSES KERJANYA

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*}

Program Pengelolaan\_Data\_FileAcak;

{ Kamus }

Uses Crt;

{ Type }

Type DataMhs = Record

Flag : Char;

Nim : String[8];

Nama : String[50];

Nilai : Real;

End;

{ Variable Global }

Var F : File of DataMhs;

T : DataMhs;

Stop : Boolean;

Ch,Pil : Char;

{-layar interaksi-}

Procedure LayarMenu;

Begin

TextBackGround(Black);ClrScr;

TextColor(White);TextBackGround(Blue);

GotoXY(25,05); Writeln('==================================');

GotoXY(25,06); Writeln('| <<< MENU === PILIHAN >>> |');

GotoXY(25,07); Writeln('==================================');

GotoXY(25,08); Writeln('| [1] Input Data Nilai Mahasiswa |');

GotoXY(25,09); Writeln('| [2] List/Lihat Daftar |');

GotoXY(25,10); Writeln('| [3] Delete/Hapus Data |');

GotoXY(25,11); Writeln('| [4] Memperbaiki/Update/Edit |');

GotoXY(25,12); Writeln('| [5] Cari/Searching Data |');

GotoXY(25,13); Writeln('| ======= |');

GotoXY(25,14); Writeln('| [6] == Quit |');

GotoXY(25,15); Writeln('| |');

GotoXY(25,16); Writeln('==================================');

GotoXY(25,17); Writeln('| Pilihan Menu No. \_ |');

GotoXY(25,18); Writeln('==================================');

End;

Procedure LayarEntry;

Begin

TextColor(White);TextBackGround(Brown);

GotoXY(25,05); Writeln('=====================================');

GotoXY(25,06); Writeln('| \*\*\* LAYAR DATA ENTRY \*\*\* |');

GotoXY(25,07); Writeln('=====================================');

GotoXY(25,08); Writeln('| |');

GotoXY(25,09); Writeln('| N I M : \_\_\_\_\_ |');

GotoXY(25,10); Writeln('| |');

GotoXY(25,11); Writeln('| N A M A : °°°°°°°°°°°°°°°°°°°°° |');

GotoXY(25,12); Writeln('| |');

GotoXY(25,13); Writeln('| NILAI : xxxxx |');

GotoXY(25,14); Writeln('| |');

GotoXY(25,15); Writeln('=====================================');

GotoXY(25,16); Writeln('| |');

GotoXY(25,17); Writeln('=====================================');

End;

Procedure LayarHapus;

Begin

TextBackGround(Black);ClrScr;

TextColor(White);TextBackGround(Brown);

GotoXY(25,05); Writeln('=====================================');

GotoXY(25,06); Writeln('| \*\*\* LAYAR HAPUS DATA \*\*\* |');

GotoXY(25,07); Writeln('=====================================');

GotoXY(25,08); Writeln('| Ketikkan NIM yang ingin dihapus ? |');

GotoXY(25,09); Writeln('| |');

GotoXY(25,10); Writeln('| N I M : \_\_\_\_\_ |');

GotoXY(25,11); Writeln('| N A M A : |');

GotoXY(25,12); Writeln('| NILAI : |');

GotoXY(25,13); Writeln('| |');

GotoXY(25,14); Writeln('=====================================');

GotoXY(25,15); Writeln('| |');

GotoXY(25,16); Writeln('=====================================');

End;

Procedure LayarUpDate;

Begin

TextBackGround(Black);ClrScr;

TextColor(White);TextBackGround(Brown);

GotoXY(25,05); Writeln('======================================');

GotoXY(25,06); Writeln('| \*\*\* LAYAR HAPUS DATA \*\*\* |');

GotoXY(25,07); Writeln('======================================');

GotoXY(25,08); Writeln('| Ketikkan NIM yang ingin diUpdate ? |');

GotoXY(25,09); Writeln('| |');

GotoXY(25,10); Writeln('| N I M : \_\_\_\_\_ |');

GotoXY(25,11); Writeln('| N A M A : |');

GotoXY(25,12); Writeln('| NILAI Baru: |');

GotoXY(25,13); Writeln('| |');

GotoXY(25,14); Writeln('======================================');

GotoXY(25,15); Writeln('| |');

GotoXY(25,16); Writeln('======================================');

End;

Procedure LayarSearch;

Begin

TextBackGround(Black);ClrScr;

TextColor(White);TextBackGround(Brown);

GotoXY(25,05); Writeln('=====================================');

GotoXY(25,06); Writeln('| \*\*\* LAYAR MENCARI DATA \*\*\* |');

GotoXY(25,07); Writeln('=====================================');

GotoXY(25,08); Writeln('| Ketikkan NIM yang ingin diCari ? |');

GotoXY(25,09); Writeln('| |');

GotoXY(25,10); Writeln('| N I M : \_\_\_\_\_ |');

GotoXY(25,11); Writeln('| N A M A : |');

GotoXY(25,12); Writeln('| NILAI : |');

GotoXY(25,13); Writeln('| |');

GotoXY(25,14); Writeln('=====================================');

GotoXY(25,15); Writeln('| |');

GotoXY(25,16); Writeln('=====================================');

End;

Procedure OpenFile; {Bagaimana kalau diinput dari keyboard ?}

Begin

Assign(F,'c:\Temp\TempData.dat');

{$i-}

Reset(F);

If(IOResult<>0) Then Rewrite(F);

{$i+}

End;

Procedure DataEntry;

Var Jawab : Char;

Begin

OpenFile; {Siapkan file}

Seek(F,Filesize(F)); {mencari posisi setelah record terakhir}

TextBackGround(Black);ClrScr; {bersihkan layar}

Repeat

LayarEntry; {Siapkan layar masukan}

{-Input data-}

GotoXY(39,09); Readln(T.Nim);

GotoXY(39,11); Readln(T.Nama);

GotoXY(39,13); Readln(T.Nilai);

{-konfirmasi-}

GotoXY(27,16); Write('Apakah Data Sudah Benar (Y/T) ? ');

GotoXY(59,16); Readln(Jawab);

If Jawab in ['y','Y'] Then Write(F,T);

GotoXY(27,16); Write('Apakah Ingin ENTRY lagi (Y/T) ? ');

GotoXY(59,16);Readln(Jawab);

Until Jawab in ['t','T'];

Close(F); {Tutup file}

End;

Procedure DataTampil;

Var No : Integer;

Begin

OpenFIle;

TextBackGround(Black);ClrScr;

TextColor(White);

Writeln('DAFTAR NILAI MAHASISWA - STMIK DCI');

GotoXy(01,02);Writeln('==============================================');

GotoXY(01,03); Writeln('No. Urut'); GotoXY(12,03); Writeln('N I M ');

GotoXY(25,03); Writeln('Nama Mahasiswa'); GotoXY(45,03);

Writeln('Nilai');

GotoXY(01,04);Writeln('==============================================');

No :=1;

While ( Not EOF(F) ) Do Begin

Read(F,T);

If (T.Nim<>'\*') Then Begin

GotoXY(03,No+4);Writeln(No);

GotoXY(12,No+4);Writeln(T.Nim);

GotoXY(25,No+4);Writeln(T.Nama);

GotoXY(45,No+4);Writeln(T.Nilai:5:0);

Inc(No);

End;

End;

GotoXY(01,No+4);

Writeln('==================================================');

Ch:=Readkey;

Close(F);

End;

Procedure DataHapus;

Var H\_Nim : String[5]; Jawab : Char;

Kode : Integer;

Begin

OpenFile;

LayarHapus;

GotoXY(39,10); Readln(H\_Nim);

Kode :=0; Jawab :='T';

While (Not EOF(F)) Do Begin

Read(F,T);

If (H\_Nim=T.Nim) Then Begin

Kode := 1;

GotoXY(39,10); Writeln(T.Nim);

GotoXY(39,11); Writeln(T.Nama);

GotoXY(39,12); Writeln(T.Nilai:3:0);

GotoXY(27,15); Write('Apakah Yakin Dihapus (Y/T) ?');Readln(Jawab);

If Jawab In ['y','Y'] Then Begin

Seek(F,FilePos(F)-1);

T.Nim :='\*';

Write(F,T);

GotoXY(27,18); Write('Proses : Data sedang dihapus.....!');

Delay(1000);

End;

End;

End;

If (Kode=0) Then Begin

GotoXY(27,18); Write('Data Tersebut tidak ada .....!');

Delay(1000);

end else

Begin

If Not (Jawab in ['y','Y']) Then Begin

GotoXY(27,18); Write('Proses Penghapusan batal .....!');

Delay(1000);

End;

End;

Close(F);

End;

Procedure DataUpDate;

Var U : DataMhs; Jawab : Char;

Kode : Integer;

Begin

OpenFile;

LayarUpDate;

GotoXY(39,10); Readln(U.Nim);

Kode := 0;

While (Not EOF(F)) Do Begin

Read(F,T);

If (U.Nim=T.Nim) Then Begin

Kode := 1;

GotoXY(39,10); Writeln(T.Nim);

GotoXY(39,11); Writeln(T.Nama);

GotoXY(39,12); Writeln(' (Nilai lama=',T.Nilai:3:0,')');

GotoXY(39,10); Readln(U.Nim);

If (U.Nim<>'') Then T.Nim:=U.Nim;

GotoXY(39,11); Readln(U.Nama);

If (U.Nama<>'') Then T.Nim:=U.Nama;

GotoXY(39,12); Readln(U.Nilai);

If (U.Nilai<>T.Nilai) Then T.Nilai:=U.Nilai;

GotoXY(27,15); Write('Apakah Data Sudah Benar (Y/T) ?');Readln(Jawab);

If Jawab In ['y','Y'] Then Begin

Seek(F,FilePos(F)-1);

Write(F,T);

GotoXY(27,18); Write('Proses : Data sedang diUpdate....!');

Delay(1000);

End;

End;

End;

If (Kode=0) Then Begin

GotoXY(27,18); Write('Data Tersebut tidak ada .....!');

Delay(1000);

end else

Begin

If Not (Jawab in ['y','Y']) Then Begin

GotoXY(27,18); Write('Proses UpDate batal .....!');

Delay(1000);

End;

End;

Close(F);

End;

Procedure DataSearch;

Var S\_Nim : String[5]; Jawab : Char;

Begin

OpenFile;

LayarSearch;

GotoXY(39,10); Readln(S\_Nim);

While (Not EOF(F)) Do Begin

Read(F,T);

If (S\_Nim=T.Nim) Then Begin

GotoXY(39,10); Writeln(T.Nim);

GotoXY(39,11); Writeln(T.Nama);

GotoXY(39,12); Writeln(T.Nilai:3:0);

Seek(F,FileSize(F)+1);

End;

End;

Ch:=Readkey;

Close(F);

End;

Procedure MenuPilihan;

Begin

LayarMenu;

Repeat

GotoXY(48,17); Pil:= Readkey;

Until (Pil in ['1'..'6']);

End;

{---Program Utama---}

Begin

Stop := False;

Repeat

TextBackGround(Black); ClrScr;

MenuPilihan;

Case Pil Of

'1' : DataEntry;

'2' : DataTampil;

'3' : DataHapus;

'4' : DataUpdate;

'5' : DataSearch;

'6' : Stop := True;

End;

Until (Stop);

TextBackGround(Black); ClrScr;

GotoXY(30,12); Write('Selesai....! <Enter> to continue..');

Ch := Readkey;

End.

# METODE ALGORITMA PADA PROSES SORTING INTERNAL

## Tipe Algoritma Sorting

Sorting adalah proses menyusun data sehingga mendapatkan hasil berupa data yang tersusun secara berurutan baik secara **ascending** (alpabetis atau nilai dari kecil menuju besar) maupun **descending** (ranking atau nilai dari besar menuju kecil) berdasarkan atribut kuncinya.

**Contoh-1 : Data sederhana**

Susunan data awal :

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 60 | 40 | 50 | 10 | 30 | 90 | 20 | 10 | 70 | 80 |

Susunan data hasil sorting ascending :

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 10 | 10 | 20 | 30 | 40 | 50 | 60 | 70 | 80 | 90 |

Susunan data hasil sorting descending :

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 90 | 80 | 70 | 60 | 50 | 40 | 30 | 20 | 10 | 10 |

**Contoh-2 : Data record**

Susunan data awal : Hasil sorting berdasar **Nilai** descending :

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Nama | %Absensi | Nilai |  |  | Nama | %Absensi | Nilai |
| Diana | 30 | 60 |  |  | Susi | 5 | 100 |
| Fery | 50 | 50 |  |  | Ali | 5 | 90 |
| Budi | 10 | 70 |  |  | Soni | 40 | 80 |
| Ali | 5 | 90 |  |  | Budi | 10 | 70 |
| Susi | 5 | 100 |  |  | Diana | 30 | 60 |
| Heri | 15 | 40 |  |  | Fery | 50 | 50 |
| Soni | 40 | 80 |  |  | Heri | 15 | 40 |

Algoritma sorting dibedakan dalam dua tipe yaitu tipe internal dan eksternal. Inti perbedaan tipe ini didasarkan pada lokasi data yang diproses. Tipe sorting internal juga disebut tipe sorting array (larik) dengan berpedoman bahwa data yang diproses bisa ditampung dalam memori utama (RAM) komputer. Tipe sorting eksternal juga disebut tipe sorting sekuensial file yang berpedoman bahwa data yang diproses tidak mencukupi ditampung dalam memori utama melainkan dalam memori sekunder (disk atau storage).

Metode-metode sorting dapat dibedakan dan dicontohkan dengan skema sebagai berikut :

Skema pembagian metode algoritma sorting

1. Natural\_InsertionSort
2. Binary\_InsertionSort
3. Shell\_InsertionSort

(Diminishing Increment)

**Metode**

**Insertion**

1. BubbleSort

(Natural, Optimal, Flag)

1. ShakerSort
2. QuickSort
3. RadixSort
4. MergeSort

**Metode**

**Counting**

1. CountingSort

Metode

**Selection**

**Internal**

**Sorting**

1. Natural\_SelectionSort

(Masimum, Minimum)

1. TreeSort

(Heap, Topological)

**Algoritma**

**Sorting**

**Metode**

**Exchange**

1. Natural\_MergeSort
2. Balanced\_MergeSort
3. Polyphase\_Sort

**Metode**

**Merging**

**Eksternal**

**Sorting**

## Internal Sorting

### Metode Counting

Secara ascending, prinsip metode ini adalah :

(1). Tentukan nilai minimal dan maksimum dari data array asal

(2). Definisikan array baru dengan indeks minimum s/d maksimum & inisialisasi dengan 0

(3). Hitung (Counting) banyaknya item data masing-masing berdasarkan nilai indeks no.(2)

(4). Tuliskan kembali ke array asal no. (1)

**Contoh ilustrasi :**

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| (1) | |  | (2) | |  | (3) | |  | (4) | |
| Asal | : T |  | Baru | :Ttemp |  | Baru | :Ttemp |  | Hasil | :T |
| Indek | **Nilai** | indek | Indek | **Nilai** |  | Indek | **Nilai** |  | Indek | **Nilai** |
| 1 | **60** | Min = | 10 | **0** |  | 10 | **2** |  | 1 | **10** |
| 2 | **40** |  | 11 | **0** |  | 11 | **0** |  | 2 | **10** |
| 3 | **50** |  | 12 | **0** |  | 12 | **0** |  | 3 | **40** |
| 4 | **10** |  | : | **0** |  | : | **0** |  | 4 | **50** |
| 5 | **50** |  | 40 | **0** | Counting | 40 | **1** | Re-write | 5 | **50** |
| 6 | **80** |  | : | **0** |  | : | **0** |  | 6 | **50** |
| 7 | **10** |  | 50 | **0** |  | 50 | **3** |  | 7 | **60** |
| 8 | **60** |  | : | **0** |  | : | **0** |  | 8 | **60** |
| 9 | **50** |  | 60 | **0** |  | 60 | **2** |  | 9 | **70** |
| 10 | **70** |  | : | **0** |  | : | **0** |  | 10 | **80** |
|  |  |  | 70 | **0** |  | 70 | **1** |  |  |  |
|  |  | indek | : | **0** |  | : | **0** |  |  |  |
|  |  | Mak= | 80 | **0** |  | 80 | **1** |  |  |  |

**Skema Algoritma :**

KAMUS DATA

**Type** INDEKS = 1 .. N;

ELEMEN = Integer;

TDATA = Array[1..N] Of ELEMEN;

**Function** MAX(T :TDATA)🡪 Integer; // nilai maksimun data

**Function** MIN(T :TDATA)🡪 Integer; // nilai minimum data

Procedure CountingSort( in/out T : TDATA; in NMin, Nmax :ELEMEN);

{I.S : T terdefinisi T[1..N], tidak ada T[i]=0}

{F.S : T[1..N] terurut Ascending}

Kamus :

Temp : Array[NMin..NMax] Of ELEMEN;

i,j: Integer; {i=indeks traversal, j=indek penulisan ulang}

Algoritma

i traversal NMin..Nmax //inisialisasi

Temp[i] ← 0

Endi

i traversal 1..N //Counting

Temp[T[i]] ← Temp[T[i]]+1

Endi

{Penulisan kembali}

i traversal NMin..Nmax

j ←0

If (Temp[i]<>0) Then

Repeat Temp[i] Times

j ← j + 1

T[j] ← i

Endrepeat

Endif

Endi

End

### Metode Insertion

#### Natural\_InsertionSort

Prinsip metode ini adalah :

(1). Insertkan data yang dianalisis (tanda →) pada posisi indeks yang seharusnya relatif

terhadap data sebelumnya.

(2). Shift (geser) data untuk rekomposisi keterurutan yang benar dari data yang dianalisis

sampai di indek yang tepat. (“-“ data yang sudah diproses atau hasil proses geser)

**Contoh ilustrasi :**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Indek | **Awal** |  | **(1)** |  | **(2)** |  | **(3)** |  | **(4)** |  | **(5)** |  | **(6)** |  | **(7)** |  | **(8)** |  | **(9)** |
| 1 | **60** | - | **40** | - | **40** | - | **10** | - | **10** | - | **10** | - | **10** | - | **10** | - | **10** | - | **10** |
| →**2** | **40** | - | **60** | - | **50** | - | **40** | - | **40** | - | **40** | - | **10** | - | **10** | - | **10** | - | **10** |
| 3 | **50** | → | **50** | - | **60** | - | **50** | - | **50** | - | **50** | - | **40** | - | **40** | - | **40** | - | **40** |
| 4 | **10** |  | **10** | → | **10** | - | **60** | - | **50** | - | **50** | - | **50** | - | **50** | - | **50** | - | **50** |
| 5 | **50** |  | **50** |  | **50** | → | **50** | - | **60** | - | **60** | - | **50** | - | **50** | - | **50** | - | **50** |
| 6 | **80** |  | **80** |  | **80** |  | **80** | → | **80** | - | **80** | - | **60** | - | **60** | - | **50** | - | **50** |
| 7 | **10** |  | **10** |  | **10** |  | **10** |  | **10** | → | **10** | - | **80** | - | **60** | - | **60** | - | **60** |
| 8 | **60** |  | **60** |  | **60** |  | **60** |  | **60** |  | **60** | → | **60** | - | **80** | - | **60** | - | **60** |
| 9 | **50** |  | **50** |  | **50** |  | **50** |  | **50** |  | **50** |  | **50** | → | **50** | - | **80** | - | **70** |
| 10 | **70** |  | **70** |  | **70** |  | **70** |  | **70** |  | **70** |  | **70** |  | **70** | → | **70** | - | **80** |

**Skema Algoritma :**

KAMUS DATA

**Type** INDEKS = 0 .. N;

ELEMEN = Integer;

TDATA = Array[0..N] Of ELEMEN;

Procedure **Natural\_InsertionSort**(in/out T : TDATA);

{I.S : T sudah berisi DATA sembarang }

{F.S : T[1]<T[2]<T[3]…<T[N]

Kamus i,j : INDEKS // indeks untuk elemen, indeks untuk geser

Temp : ELEMEN // temporari item

Algoritma

i Traversal 2.. N

Temp 🡨 T[i]

j 🡨 i-1

While (T[j] > Temp) and (j>=1) do

T[j+1] 🡨 T[j]; //geser elemen

j 🡨 j–1; //cari lokasi yang sesuai

// jika j=0, harus ditangani khusus karena indeks j=0 tidak ada

// Solusinya dengan sentinel membuat indeks 0

// atau dengan menangani j=0 dengan boolean pengganti (j>=0)

Endwhile

T[J+1] 🡨 Temp; {mengisi posisi yang sesuai}

Endi;

End

#### Binary\_InsertionSort

Prinsip metode ini merupakan optimasi Natural\_InsertionSort dengan mengurangi rata-rata pengeseran adalah :

(1). Insertkan data yang dianalisis (tanda →) pada posisi indeks yang seharusnya.

(2). Shift (geser) data untuk rekomposisi yang benar dari data yang dianalisis, berpedoman pada posisi indeks yang benar dengan cara dibagi 2 (biner) area Kiri(Left) dan Kanan(Right) dipisahkan oleh Tengah (Mid). Ilustrasi pembagian biner diperlihatkan pada contoh berikut. Misal proses sudah berlangsung pada elemen ke-9, sehingga elemen-1 s/d elemen-8 sudah terurut, maka proses Binary \_InsertionSort dari array T.

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| T | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | :Fokus tahapan pada indeks-9 =Temp = 25 |
| (1) | **L** |  |  |  | **M** |  |  |  | **R** |  | :M=(L+R)div 2 =5; T[m]> Temp |
| Left | 10 | 10 | 40 | 50 | 50 | 60 | 60 | 80 | **25** | 70 | Right |

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| (2) | **L** |  | **M** |  | **R** |  |  |  |  |  | :M=(L+R)div 2 =3; T[m]> Temp |
| Left | 10 | 10 | 40 | 50 | 50 | 60 | 60 | 80 | **25** | 70 | Right |

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| (3) | **L** | M | **R** |  |  |  |  |  |  |  | :M=(L+R)div 2 =2; T[m]< Temp, siap geser |
| Left | 10 | 10 | 40 | 50 | 50 | 60 | 60 | 80 | **25** | 70 | Right |

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| (4) | **L** | M | **R** |  |  |  |  |  |  |  | :Geser dari indeks-8..indeks-R; T[M]🡨Temp |
| Left | 10 | 10 | **25** | 40 | 50 | 50 | 60 | 60 | 80 | 70 | Right |

**Contoh ilustrasi :**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Indek | **Awal** |  | **(1)** |  | **(2)** |  | **(3)** |  | **(4)** |  | **(5)** |  | **(6)** |  | **(7)** |  | **(8)** |  | **(9)** |
| M=L:1 | **60** | L | **40** | L | **40** | L | **10** | L1 | **10** | L | **10** | L1 | **10** | L | **10** | L1 | **10** |  | **10** |
| →R:2 | **40** | M | **60** | M | **50** |  | **40** |  | **40** | M2=R3 | **40** |  | **10** |  | **10** |  | **10** |  | **10** |
| 3 | **50** | →R | **50** |  | **60** | M | **50** | M1=L2 | **50** |  | **50** |  | **40** |  | **40** |  | **40** |  | **40** |
| 4 | **10** |  | **10** | →R | **10** |  | **60** | M2=L3 | **50** | M1=R2 | **50** | M1=L2 | **50** |  | **50** |  | **50** |  | **50** |
| 5 | **50** |  | **50** |  | **50** | →**R** | **50** | M3=L4 | **60** |  | **60** |  | **50** | M | **50** | M1=L2 | **50** |  | **50** |
| 6 | **80** |  | **80** |  | **80** |  | **80** | →R | **80** |  | **80** | M2=L3 | **60** |  | **60** |  | **50** |  | **50** |
| 7 | **10** |  | **10** |  | **10** |  | **10** |  | **10** | →R1 | **10** |  | **80** |  | **60** | M2=L3 | **60** |  | **60** |
| 8 | **60** |  | **60** |  | **60** |  | **60** |  | **60** |  | **60** | →R | **60** |  | **80** | M3=L4 | **60** |  | **60** |
| 9 | **50** |  | **50** |  | **50** |  | **50** |  | **50** |  | **50** |  | **50** | →R | **50** | M4=L5 | **80** |  | **70** |
| 10 | **70** |  | **70** |  | **70** |  | **70** |  | **70** |  | **70** |  | **70** |  | **70** | →R | **70** |  | **80** |

**Skema Algoritma :**

KAMUS DATA

**Type** INDEKS = 1 .. N;

ELEMEN = Integer;

TDATA = Array[1..N] Of ELEMEN;

Procedure **Binary\_InsertionSort**(in/Out T : TDATA)

{I.S : T sudah berisi DATA sembarang }

{F.S : T[1]<T[2]<T[3]…<T[N]

Kamus i,j : INDEKS; // indeks untuk elemen, indeks untuk geser

L,M,R : INDEKS; // Left-Mid-Right = Top-Mid-Bottom

Temp : ELEMEN; // Temporar item

Algoritma

i traversal 2..N

Temp 🡨 T[i];

L 🡨 1;

R 🡨 i-1;

While (L<=R) do begin

M 🡨 (L+R) div 2;

If (T[M}> Temp) then

R := M - 1

else

L := M + 1

endif

Endwhile

j Traversal i-1..L

T[j+1] := T[j]

endj

T[L] := Temp; {mengisi posisi yang sesuai}

Endi

End

#### Shell\_InsertionSort

Prinsip metode ini adalah :

(1). Insertkan data dengan cara menukarkan.data diposisi ke-i dengan data ke-(i+Step)

(2). Ulangi proses (1) dengan cara menurunkan step (biasanya step div 2)

**Contoh ilustrasi :**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Indek | **Awal** |  | (1)  Step = 5  ⎡10/2⎤ |  | (2)  Step = 3  ⎡5/2⎤ |  | (3)  Step = 2  ⎡3/2⎤ |  | (4)  Step = 1  ⎡2/2⎤ |
| 1 | **60** |  | **60** |  | **10** |  | **10** |  | **10** |
| 2 | **40** |  | **10** |  | **10** |  | **10** |  | **10** |
| 3 | **50** |  | **50** |  | **50** |  | **50** |  | **40** |
| 4 | **10** |  | **10** |  | **40** |  | **40** |  | **50** |
| 5 | **50** |  | **50** |  | **50** |  | **50** |  | **50** |
| 6 | **80** |  | **80** |  | **50** |  | **50** |  | **50** |
| 7 | **10** |  | **40** |  | **60** |  | **60** |  | **60** |
| 8 | **60** |  | **60** |  | **60** |  | **60** |  | **60** |
| 9 | **50** |  | **50** |  | **80** |  | **80** |  | **70** |
| 10 | **70** |  | **70** |  | **70** |  | **70** |  | **80** |

KAMUS DATA

**Type** INDEKS = 1 .. N;

ELEMEN = Integer;

TDATA = Array[1..N] Of ELEMEN;

Procedure **Shell\_InsertionSort**(in/Out T : TDATA)

{I.S : T sudah berisi DATA sembarang }

{F.S : T[1]<T[2]<T[3]…<T[N]

Kamus i,j : INDEKS;

Step : Integer

Temp : ELEMEN;

Algoritma

Step 🡨 ⎡N/2⎤ // pembulatan ke atas jarak antara data yang dianalisis

While (Step > 0) do

i Traversal 1..(N-Step)

j 🡨 i + Step

If (T[i]>T[j]) then

// Tukar

Temp 🡨 T[i]

T[i] 🡨 T[j]

T[j] 🡨 Temp

Endif

Endi

Step 🡨 ⎡N/2⎤ //penurunan untuk jarak selanjutnya

Endwhile

End

## Metode Selection

### Natural\_SelectionSort

Prinsip metode ini adalah :

(1). Seleksi nilai minimum tandai indeksnya dengan (\*) mulai seleksi dari tanda (>)

(2). Tukar data antara data pada indeks tanda (→) dengan data pada indeks tanda (\*) terakhir.

(tanda “-“ data yang sudah diproses)

**Contoh ilustrasi :**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Indek | **Awal** |  | **(1)** |  | **(2)** |  | **(3)** |  | **(4)** |  | **(5)** |  | **(6)** |  | **(7)** |  | **(8)** |  | **(9)** |
| K:→1 | **60** | - | **10** | - | **10** | - | **10** | - | **10** | - | **10** | - | **10** | - | **10** | - | **10** | - | **10** |
| J:\*2 | **40** | → | **40** | - | **10** | - | **10** | - | **10** | - | **10** | - | **10** | - | **10** | - | **10** | - | **10** |
| 3 | **50** | > | **50** | → | **50** | - | **40** | - | **40** | - | **40** | - | **40** | - | **40** | - | **40** | - | **40** |
| \*4 | **10** |  | **60** | > | **60** | → | **60** | - | **50** | - | **50** | - | **50** | - | **50** | - | **50** | - | **50** |
| 5 | **50** |  | **50** |  | **50** | **>\*** | **50** | → | **60** | - | **50** | - | **50** | - | **50** | - | **50** | - | **50** |
| 6 | **80** |  | **80** |  | **80** |  | **80** | > | **80** | → | **80** | - | **50** | - | **50** | - | **50** | - | **50** |
| 7 | **10** | \* | **10** | \* | **40** |  | **50** | \* | **50** | > | **60** | → | **60** | - | **60** | - | **60** | - | **60** |
| 8 | **60** |  | **60** |  | **60** |  | **60** |  | **60** |  | **60** | >\* | **60** | → | **60** | - | **60** | - | **60** |
| 9 | **50** |  | **50** |  | **50** |  | **50** |  | **50** | \* | **50** |  | **80** | > | **80** | → | **80** | - | **70** |
| 10 | **70** |  | **70** |  | **70** |  | **70** |  | **70** |  | **70** |  | **70** |  | **70** | >\* | **70** | - | **80** |

**Skema Algoritma :**

KAMUS DATA

**Type** INDEKS = 1 .. N;

ELEMEN = Integer;

TDATA = Array[1..N] Of ELEMEN;

Procedure **Natural\_SelectionSort** (in/Out T : TDATA)

{I.S : T sudah berisi DATA sembarang }

{F.S : T[1]<T[2]<T[3]…<T[N] }

Kamus i,j,k : INDEKS;

Temp : ELEMEN;

Algoritma // Minimum Selection Sort

i Traversal 1..(N-1)

Temp 🡨 T[i] //data awal yang diperiksa

k 🡨 i //index data yang diseleksi

j Traversal (i+1)..N

If (T[j] < Temp) then // cari minimum

k 🡨 j // Catat indeks

Temp := T[j] // tampung minimum

Endif

Endj

{Tukar posisi yang benar}

T[k] 🡨 T[i];

T[i] 🡨 Temp;

Endi

End

### Heap\_SelectionSort

Procedure **Heap\_SelectionSort**(in/Out T : TDATA)

{I.S : T sudah berisi DATA sembarang }

{F.S : T[1]<T[2]<T[3]…<T[N] }

{Pelajari sendiri : prinsip tentang seleksi ini sebagai tugas}

## Metode Exchange

### Natural\_BubbleSort

Prinsip metode ini menggunakan ilustrasi gambar di bawah adalah :

(1). Analisis bubble (gelembung) data setiap Pass (i) dimulai pada indeks j =1 ditandai (→)

(2). Penggelembungan data ke-j dari 1 sampai dengan (10-1) setiap Pass dilakukan berikut :

- data ke-j dibandingkan dengan data ke-(j+1)

- jika data ke-j < data ke-(j+1) maka lakukan tukar data (exchange) ditandai (\*)

- jika data ke-j ≥ data ke-(j+1) maka data gelembung ganti data ke-(j+1) ditandai (→)

**Contoh ilustrasi :**

Langkah *pass* tidak efektif=tidak optimal

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  | **Pass** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Indeks i: | **Awal** |  | **(1)** |  | **(2)** |  | **(3)** |  | **(4)** |  | **(5)** |  | **(6)** |  | **(7)** |  | **(8)** |  | **(9)** |  | **(10)** |
| j:> 1 | **60** | > | **40** | → | **40** | → | **40** | → | **10** | → | **10** | → | **10** | → | **10** | → | **10** | → | **10** | → | **10** |
| \*2(60) | **40** | → | **50** | → | **50** | \* | **10** | → | **40** | → | **40** | → | **10** | → | **10** | → | **10** | → | **10** | → | **10** |
| \*3(60) | **50** | \* | **10** | \* | **10** | → | **50** | → | **50** | \* | **10** | → | **40** | → | **40** | → | **40** | → | **40** | → | **40** |
| \*4(60) | **10** | → | **50** | → | **50** | → | **50** | \* | **10** | → | **50** | → | **50** | → | **50** | → | **50** | → | **50** | → | **50** |
| \*5(60) | **50** | → | **60** | → | **60** | \* | **10** | → | **50** | → | **50** | → | **50** | → | **50** | → | **50** | → | **50** | → | **50** |
| →6(80) | **80** | \* | **10** | \* | **10** | → | **60** | → | **50** | → | **50** | → | **50** | → | **50** | → | **50** | → | **50** | → | **50** |
| \*7(80) | **10** | → | **60** | → | **50** | \* | **50** | → | **60** | → | **60** | → | **60** | → | **60** | → | **60** | → | **60** | → | **60** |
| \*8(80) | **60** | \* | **50** | \* | **60** | → | **[60]** | → | **60** | → | **60** | → | **60** | → | **60** | → | **60** | → | **60** | → | **60** |
| \*9(80) | **50** | → | **70** | → | **[70]** | → | **[70]** | → | **70** | → | **70** | → | **70** | → | **70** | → | **70** | → | **70** | → | **70** |
| 10(80) | **70** | - | **[80]** |  | **[80]** |  | **[80]** |  | **80** |  | **80** |  | **80** |  | **80** |  | **80** |  | **80** |  | **80** |

|  |
| --- |
| **Analisis bubble** |

**Skema Algoritma :**

KAMUS DATA

**Type** INDEKS = 1 .. N;

ELEMEN = Integer;

TDATA = Array[1..N] Of ELEMEN;

Procedure **Natural\_BubbleSort** (in/Out T : TDATA)

{I.S : T sudah berisi DATA sembarang }

{F.S : T[1]<T[2]<T[3]…<T[N] }

Kamus i,j : INDEKS;

Temp : ELEMEN;

Algoritma

i Traversal 1..N

j Traversal 1..(N-1)

If (T[j+1]<T[j]) then //Tukar posisi

Temp := T[j+1];

T[j+1] := T[j];

T[j] := Temp;

Endif;

Endj

Endi

End

### Optimal\_BubbleSort

Prinsip metode ini adalah adalah mengoptimalkan proses dengan cara mengurangi pemborosan iterasi **pass** yang tidak perlu lagi yaitu iterasi yang pernah terjadi tidak perlu diulang lagi.

Optimasi terlihat pada cuplikan baris berikut ini.

|  |  |
| --- | --- |
| Algoritma  i Traversal 1..N  🡪 j Traversal 1..(N-1)  :  Endj  Endi  End | Algoritma  i Traversal 2..(N-1)  🡪 j Traversal N..i  :  Endj  Endi  End |

Prinsip metode ini adalah :

(1). Proses sama dengan Natural\_BubbleSort

(2). Proses penukaran dan pembandingan (penggelembungan) dioptimalkan pengulangannya.

Procedure **Optimal\_BubbleSort** (in/Out T : TDATA)

{I.S : T sudah berisi DATA sembarang }

{F.S : T[1]<T[2]<T[3]…<T[N] }

Kamus i,j : INDEKS;

Temp : ELEMEN;

Algoritma

i Traversal 2..(N-1)

j Traversal N..i

If (T[j]<T[j-1]) then //Tukar posisi

Temp := T[j-1];

T[j-1] := T[j];

T[j] := Temp;

Endif;

Endj

Endi

End

### Flag\_BubbleSort

Prinsip metode ini adalah adalah mengoptimalkan lagi proses dari Optimal\_BubbleSort dengan cara mengurangi iterasi **pass** jika sudah diketahui tidak terjadi proses penukaran (exchange). Hal ini terlihat pada pass ke-6 sampai dengan pass ke-10 selalu tetap tanpa perubahan (penukaran).

Prinsip metode ini adalah :

(1). Proses sama dengan Optimal\_BubbleSort

(2). Proses optimalisasi lebih lanjut dari Optimal\_BubbleSort dilakukan dengan variabel boolean berupa flag sampai bernilai true yaitu apabila sudah tidak terjadi proses penukaran lagi (sudah terurut).

Procedure **Optimal\_BubbleSort** (in/Out T : TDATA)

{I.S : T sudah berisi DATA sembarang }

{F.S : T[1]<T[2]<T[3]…<T[N] }

Kamus i,j : INDEKS;

Temp : ELEMEN;

Flag : Boolean

Algoritma

Flag 🡨 True

i 🡨 2

While (i<=N) and (Flag) do

Flag 🡨 False

j := N;

While(j>=i) do

If (T[j]<T[j-1]) then //Tukar posisi

Temp := T[j-1]

T[j-1] := T[j]

T[j] := Temp

Flag 🡨 True

Endif;

J 🡨 j–1;

EndWhile //jika tidak ada penukaran maka flag = false

i 🡨 i + 1;

EndWhile

End

### Shaker\_ ExchangeSort

Prinsip metode ini merupakan Bubble Sort dua arah, yaitu :

(1). Analisis bubble-1 (gelembung-1) dari R ke L (bawah ke atas), lalu dilanjutkan analisis

bubble-2 dari L ke R (atas ke bawah). Analisis bubble diulangi dengan L + 1, dan R-1

( ruang analisis menyempit)

(2). Tukar data antara data pada indeks tanda (>) dengan data berikutnya pada indeks tanda

(\*) jika data di atasnya lebih kecil.

**Contoh ilustrasi :**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Indek(1) | **Awal** |  | **(2)** |  | **(3)** |  | **(4)** |  | **(5)** |  | **(6)** |  | **(7)** |  | **(8)** |  | **(9)** |
| \*1 | **60** |  | **10** |  | **10** |  | **10** |  | **10** |  | **10** |  | **10** |  | **10** |  | **10** |
| L >\*2 | **40** | \* | **60** |  | **40** |  | **10** |  | **10** |  | **10** |  | **10** |  | **10** |  | **10** |
| \*3 | **50** | i=L> | **40** | \* | **50** |  | **40** |  | **40** |  | **40** |  | **40** |  | **40** |  | **40** |
| >4 | **10** | \* | **50** | > | **10** | i=L> | **50** | > | **50** |  | **50** |  | **50** |  | **50** |  | **50** |
| \*5 | **50** | \* | **10** | > | **50** | **>** | **50** | > | **50** | i=L> | **50** | > | **50** |  | **50** |  | **50** |
| \*6 | **80** | \* | **50** | \* | **60** | > | **50** | > | **50** | > | **50** | > | **50** | i=L> | **50** | L=R | **50** |
| >7 | **10** | > | **80** | > | **50** | > | **60** | > | **60** | > | **60** | i=R> | **60** | > | **60** |  | **60** |
| \*8 | **60** | \* | **50** | > | **60** | > | **60** | i=R> | **60** | > | **60** |  | **60** |  | **60** |  | **60** |
| >9 | **50** | \* | **60** | i=R> | **70** | > | **70** |  | **70** |  | **70** |  | **70** |  | **70** |  | **70** |
| i=R >10 | **70** | \* | **70** |  | **80** |  | **80** |  | **80** |  | **80** |  | **80** |  | **80** |  | **80** |

Procedure **Shaker\_ExchangeSort** (in/Out T : TDATA)

{I.S : T sudah berisi DATA sembarang }

{F.S : T[1]<T[2]<T[3]…<T[N] }

kamus i,j : INDEKS;

L,R : INDEKS; {Left, Right]

Temp : ELEMEN; {temporary}

Algoritma

L 🡨 2; R 🡨 N;

j 🡨 N;

REPEAT

i traversal R..L

If (T[i-1] > T[i]) then //Tukar posisi

Temp🡨T[i-1]; T[i-1]🡨T[i]; T[i]🡨Temp;

J🡨i //indeks penukaran

Endif

Endi

L := j + 1; //Catat indeks penukaran bawah terakhir

i traversal L..R

If (T[i-1] > T[i]) then //Tukar posisi

Temp🡨T[i-1]; T[i-1]🡨T[i]; T[i]🡨Temp;

J🡨i //indeks penukaran

Endif

Endi

R := j - 1; {Catat indeks penukaran atas terakhir}

UNTIL (L>R);

End

### Quick\_ExchangeSort

Procedure **Quick\_ExchangeSort**(in/Out T : TDATA)

{I.S : T sudah berisi DATA sembarang }

{F.S : T[1]<T[2]<T[3]…<T[N] }

kamus

Const M = (N \* 2)

var i,j : INDEKS;

L,R : INDEKS; {Left, Right]

TempX, TempY : ELEMEN; {temporary}

S : 0..M

Stack : Array[1..M] Of <L,R :INDEKS>

Algoritma

S 🡨 1

Stack[1].L 🡨 1

Stack[1].R 🡨 N

REPEAT

L 🡨 Stack[S].L

R 🡨 Stack[S].R

S 🡨 S-1

REPEAT

i 🡨 L

j 🡨 R

TempX 🡨 T[(L+R) div 2]

REPEAT

While (T[i] < TempX) Do

i 🡨 i + 1

endwhile

While (T[j] > TempX) Do

j 🡨 j - 1

endwhile

If (i<=j) then

TempY 🡨 T[i]

T[i] 🡨 T[j]

T[j] 🡨 TempY

i 🡨 i + 1

j 🡨 j - 1

endif

UNTIL (i>j)

If (i<=R) then

S 🡨 S + 1

Stack[S].L 🡨 i

Stack[S].R 🡨 R

endif

R 🡨 j

UNTIL (L>=R)

UNTIL (S=0);

End

### Radix\_ExchangeSort

D=Right=1, Radix = R

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Indek(1) | **Awal** |  | D=1  R=0..9 |  | D=2  R=0 | D=2  R=1 | D=2  R=2 | D=2  R=3 | D=2  R=4 | D=2  R=5 | D=2  R=6 | D=2  R=7 | D=2  R=8 | D=2  R=9 |
| Pos>1 | **60** | Pos> | **60** | Pos> | **60** | **10** | **10** | **10** | **10** | **10** | **10** | **10** | **10** | **10** |
| 2 | **40** |  | **40** |  | **40** | **10** | **10** | **10** | **10** | **10** | **10** | **10** | **10** | **10** |
| 3 | **50** |  | **50** |  | **50** | **>50** | **>50** | **>50** | **40** | **40** | **40** | **40** | **40** | **40** |
| 4 | **10** |  | **10** |  | **10** | **60** | **60** | **60** | **>60** | **50** | **50** | **50** | **50** | **50** |
| 5 | **50** |  | **50** |  | **50** | **50** | **50** | **50** | **50** | **50** | **50** | **50** | **50** | **50** |
| 6 | **80** |  | **80** |  | **80** | **80** | **80** | **80** | **80** | **50** | **50** | **50** | **50** | **50** |
| 7 | **10** |  | **10** |  | **10** | **40** | **40** | **40** | **50** | **>60** | **60** | **60** | **60** | **60** |
| 8 | **60** |  | **60** |  | **60** | **60** | **60** | **60** | **60** | **60** | **60** | **60** | **60** | **60** |
| 9 | **50** |  | **50** |  | **50** | **50** | **50** | **50** | **50** | **80** | **>80** | **70** | **70** | **70** |
| 10 | **70** |  | **70** |  | **70** | **70** | **70** | **70** | **70** | **70** | **70** | **>80** | **80** | **80** |

Procedure **NaturalRadix\_ExchangeSort**(in/Out T : TDATA)

{I.S : T sudah berisi DATA sembarang, T[i] string }

{F.S : T[1]<T[2]<T[3]…<T[N] }

kamus

Const M = (N \* 2)

var Radix : 0..9;

D : Integer; //digit dari Item

i,Pos,Left,Right : integer;

Temp : ELEMEN; {temporary}

Algoritma

D traversal Right .. Left

Pos 🡨 1

Radix traversal 0..9

While (Pos <= N) Do

If (T[i][D]=Radix) then

Temp 🡨 T[Pos]

T[Pos]🡨 T[i]

T[i]🡨 Temp

Pos 🡨 Pos + 1

endif

endWhile

EndRadix

EndD

End

**Catatan**:

NaturalRadix Sort dapat dioptimalkan sebagaimana bubble Sort,

Baik dengan pengurangan proses penggelembungan atau dengan flag.

### Merge\_ExchangeSort

Procedure **Merge\_ExchangeSort**(in/Out T : TDATA)

{I.S : T sudah berisi DATA sembarang }

{F.S : T[1]<T[2]<T[3]…<T[N] }

{Pelajari sendiri : prinsip tentang exchange ini sebagai tugas}

DAFTAR PUSTAKA

Cormen, Thomas H., Charles E. Leiserson, Ronald L. Riverst. Introduction to Algorithms. McGraw-Hall, 1990

Jensen, Katleen and Niklaus Wirth. Pascal User Manual and Report. Sec. Ed, Springer-Verlag, New York, 1974

Korfhage, Robert R. Logic and Algorithms. John Wiley & Sons, Inc.,New York, 1996

Liem, Ingriani. Diktat Kuliah Algoritma Pemrograman, ITB

Santosa, P. Insap, Struktur Data, Andi Offset, Yogyakarta, 1992

Schneider G.M., and S.C. Bruell. Advanced Programming and Problem Solving with Pascal. John Wiley & Son Inc., Canada, 1981

Tenembaum, Aaron M and Moshe J. Augenstein. Data Structure Using Pascal. Prentice-Hall Inc., Englewood Cliffs, New Yersey, 1981

Wirth, Nikalus. Algorithms + Data Structure = Programs. Prentice-Hall of India private limited, New Delhi, 1991

.

RIWAYAT PENULIS

 Sanyata Purwidayanta, Ir. M.T setelah menamatkan pendidikan dari SMA negeri 1 Sragen, melanjutkan studinya di bidang Teknik Informatika ITB Bandung dan meraih gelar Sarjana Teknik tahun 1996. Kemudian pada tahun 1997 melanjutkan studi program pascasarjana masih di bidang Informatika ITB dengan memperoleh gelar Magister Teknik tahun 2000 pada bidang khusus Rekayasa Perangkat Lunak.

Di awal-awal tahun menjadi mahasiswa, beliau melakukan kegiatan sampingan memberikan les privat kepada siswa-siswa SMA untuk pelajaran matematika dan fisika. Karena kesukaannya pada bidang pendidikan dan pengajaran, beliau melakukan kegiatan menjadi pengajar kursus aplikasi komputer dan instruktur praktik laboratorium program setara diploma-1 di Data Computer Indonesia Bandung untuk mata kuliah pemrograman bisnis dan pemrograman akuntansi. Pada periode ini pula beliau dipercaya merangkap jabatan sebagai koordinator jurusan pemrograman dan teknik komputer.

Setelah lulus Sarjana, beliau menjadi dosen muda di Jurusan Teknik Informatika pada Sekolah Tinggi Indonesia Bandung pada mata kuliah Algoritma Pemrogram dan Basis data. Sempat menjabat Sekretaris Jurusan Teknik Informatika satu tahun, lalu dilepas karena melanjutkan studi pascasarjana strata-2.

Setelah lulus pascasarjana strata-2, beliau aktif kembali menjadi dosen pada STMIK DCI untuk mata kuliah Algoritma pemrograman, Basis data, Rekayasa Perangkat Lunak, dan Sistem informasi. Di STMIK DCI ini beliau pernah diberi beberapa ananah dengan berbagai tugas jabatan, mulai dari Wakil Ketua Bidang Akademik, Ketua Institusi, Ketua LPPM, Ketua Senat Akdemik, dan Wakil ketua Bidang Sumber Daya dan keuangan. Peran utama sekarang beliau adalah sebagai dosen.