Fakultas Informatika
School of Computing
Telkom University

CSH3E3
Implementasi dan Pengujian

Tim Dosen

KK SIDE

Pertemuan 4

Implementasi Desain Prosedural

Agenda

- Review PSPEC
- Implementasi Proses di DFD ke Bahasa Pascal/C
- Implementasi PSPEC format Strucured English ke Pascal/C
- Implementasi PSPEC format DECISION TABLE ke Pascal/C
- Implementasi Data Dicitonary
- Latihan Di Kelas
- Tugas Eksplorasi

Sub Bahasan 1

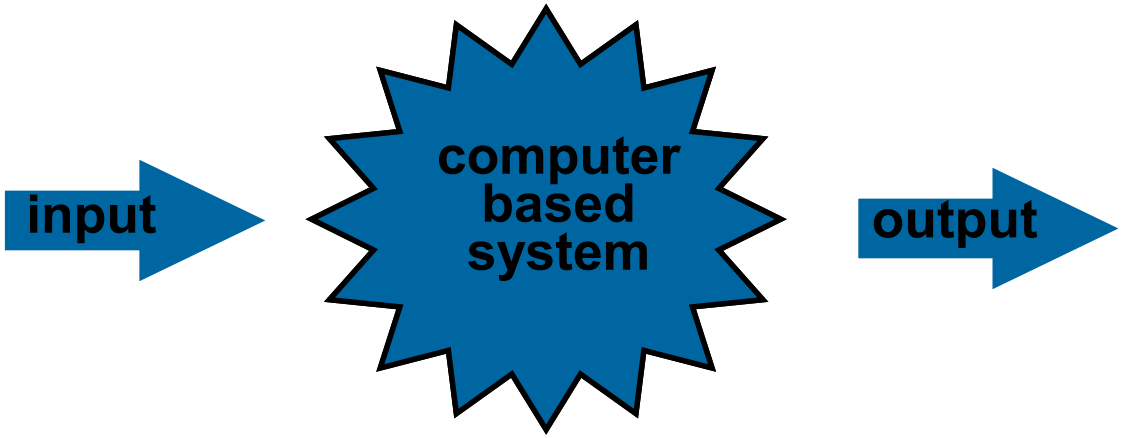
Review PSPEC

Flow-Oriented Modeling

- Represents how data objects are transformed at they move through the system
- **data flow diagram (DFD)** is the diagrammatic form that is used
- Considered by many to be an "old school" approach, but continues to provide a view of the system that is unique—it should be used to supplement other analysis model elements

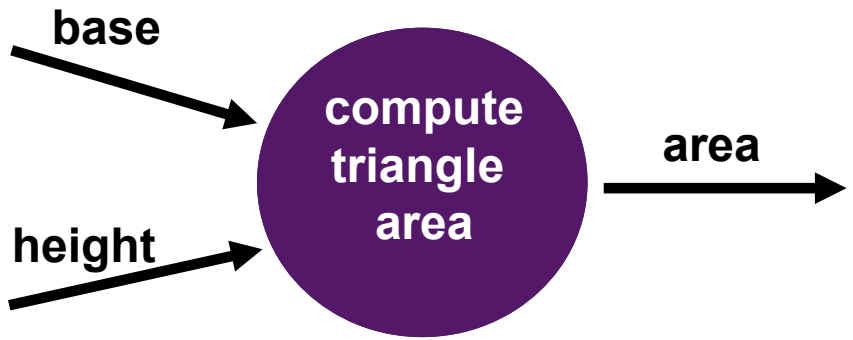
The Flow Model

Every computer-based system is an information transform ....

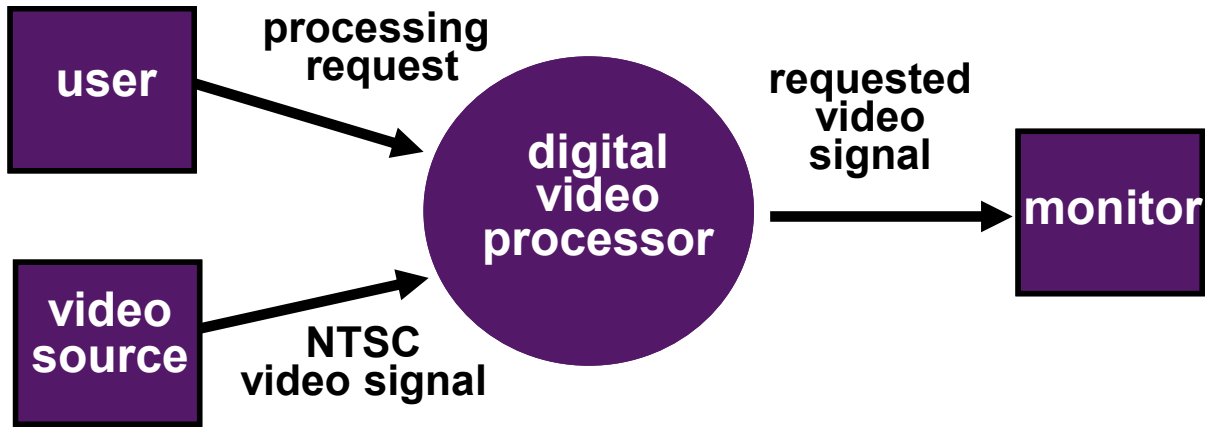


Data Flow

Data flows through a system, beginning as input and transformed into output.



Level 0 DFD Example



From Design to Implementation

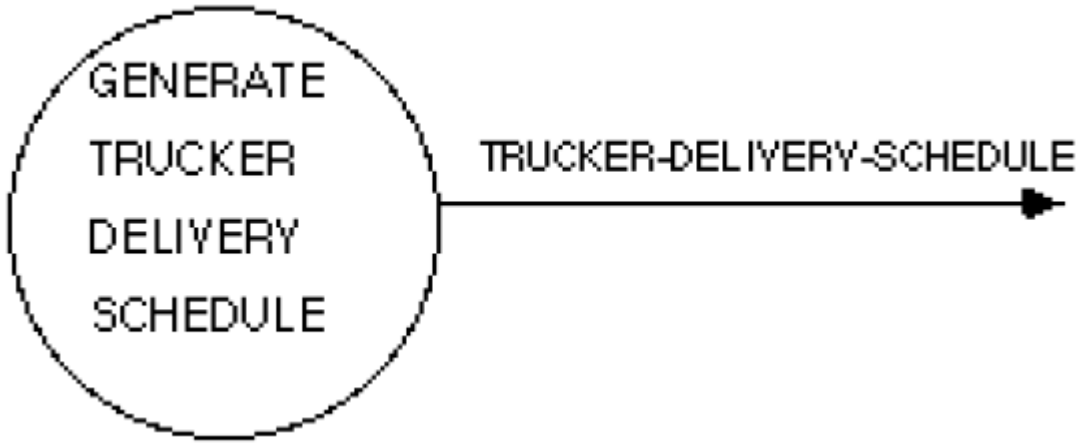
- DFD (lowest level) → code
  - Process (circle/bubble): Function or Procedure
  - Flow (directed arrow): data type for input or output in Function or Procedure
  - Data Store (two parallel lines/ellipse): variable/file/database tables
  - Process Specification (PSPEC) → code flow in Function or Procedure
- Data Dictionary
  - variable/file/database tables input validation

Note: Example codes are in Pascal/C language

Sub Bahasan 2

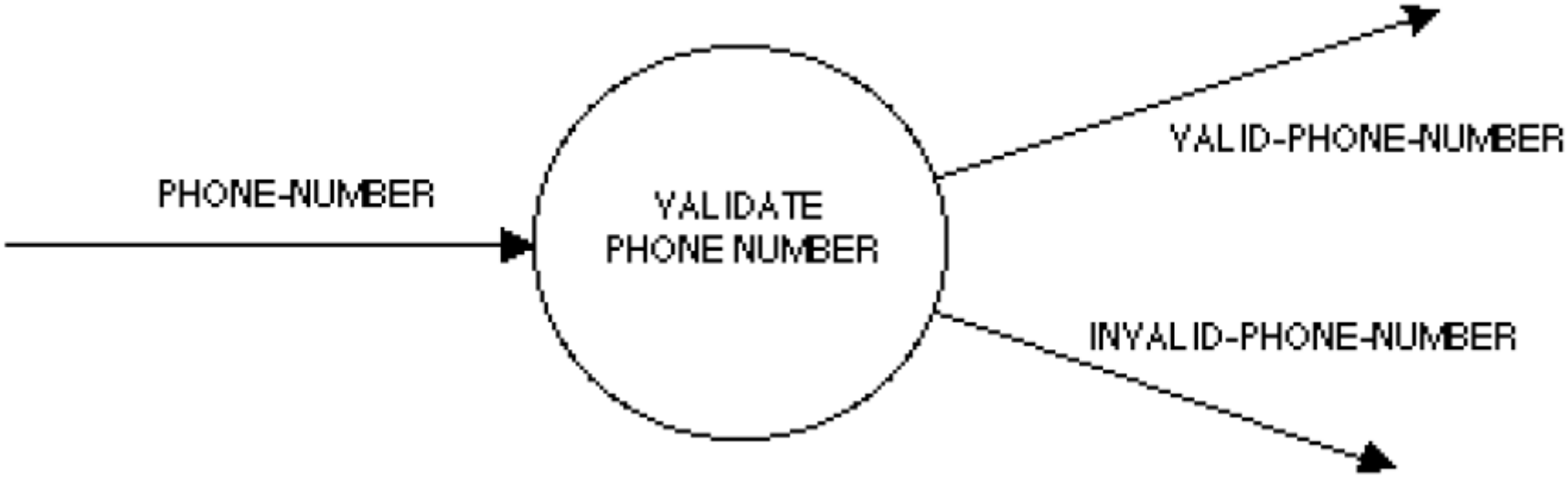
Implementasi Proses di DFD ke Bahasa Pascal/C

Process & Flow (output only)




```
TruckerDeliverySchedule GenerateTruckerDeliverySchedule(void)
{
        ...
}
```

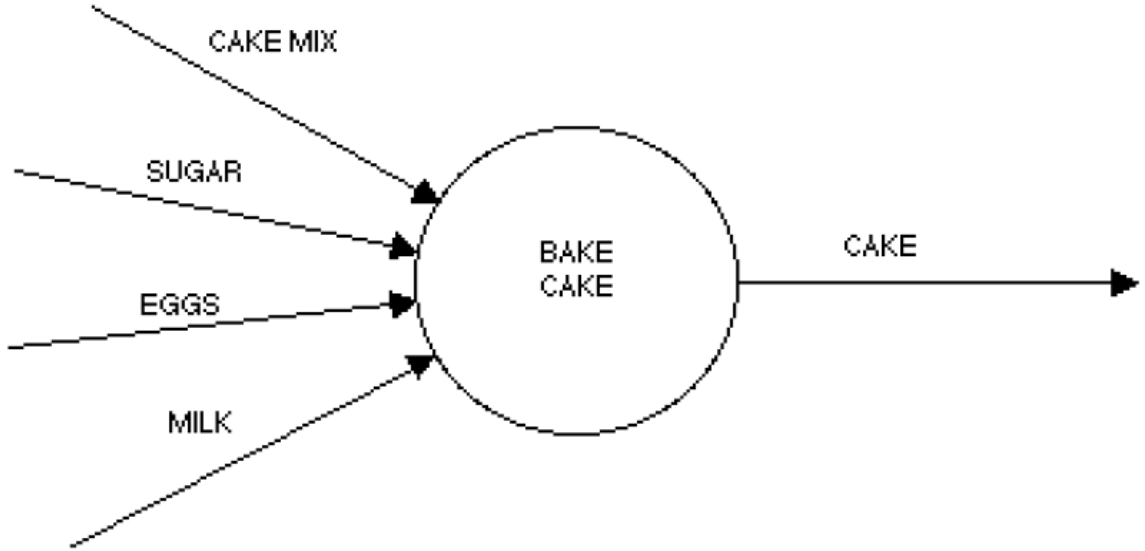
Process & Flow (input & output)




```
boolean ValidatePhoneNumber(PhoneNumber p)
{
        ...
}
```

BAGAIMANA MEMBUAT CODE UNTUK MEMERIKSA BAHWA P ITU ANGKA SEMUA (TIDAK BOLEH HURUF)?

Process & Flow (input & output)

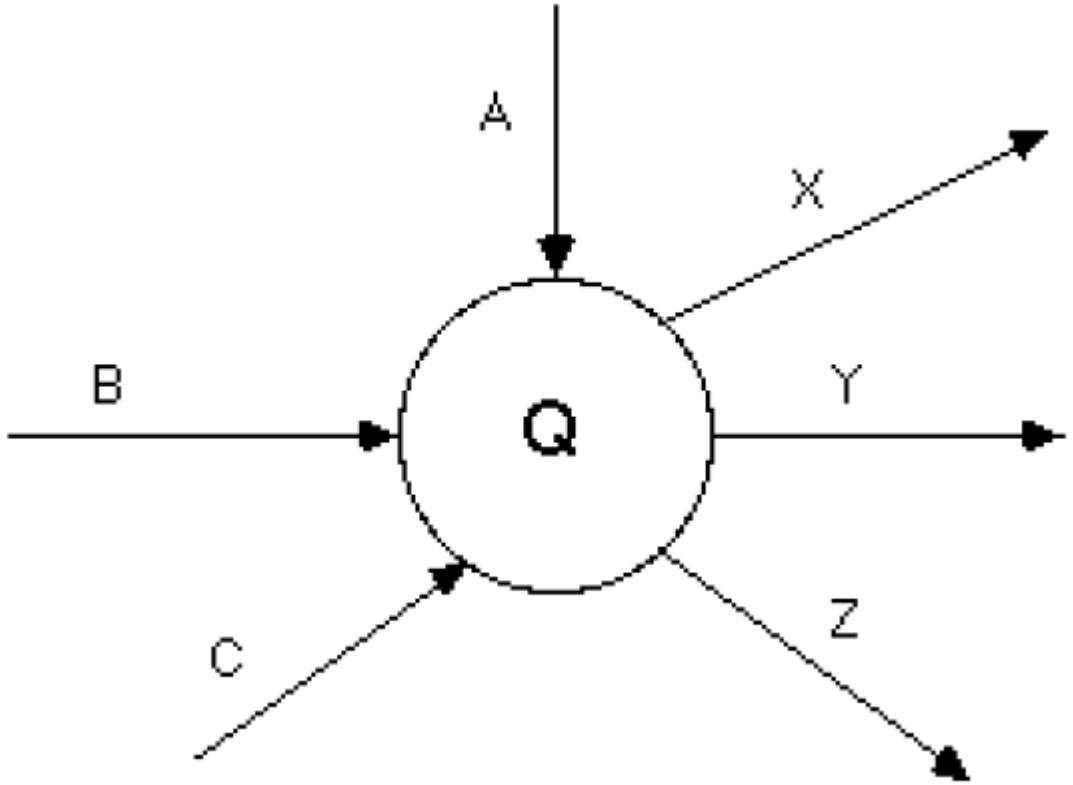



```
Cake BakeCake(CakeMix cm, Sugar s, Eggs e, Milk m)
{
        …
}
```

Process & Flow (input-output)


```
void ShipBooks(Books *B)
{
        ...
}
```

Process & Flow




```
void Q(A a, B b, C c, X *x, Y *y, Z *z)
{
        …
}
```

Data Store


```
CREATE TABLE orders (
        …
);
```

Sub Bahasan 3

Implementasi PSPEC format Strucured English ke Pascal/C

Process Specification

- a.k.a. PSPEC or minispec (abbreviation from miniature specification)
- It defines what must be done in order to transform inputs into outputs.
- Two crucial requirements:
  1. *must be expressed in a form that can be verified by the user and the systems analyst.*
     narrative English is avoided as specification tool as it is ambiguous especially when describing alternative actions (decisions) and repetitive actions (loops)
  2. *must be expressed in a form that can be effectively communicated to the various audiences involved.*

Three major process specification tools

- Structured English
- Pre/post conditions
- Decision tables

Structured English

- Structured English is “English with structure.”
- a.k.a. PDL (for program design language) and PSL (for problem statement language or problem specification language).
- Examples:
  - X = (Y*Z)/(Q+14)
  - COMPUTE X = (Y*Z)/(Q+14)
  - SET TAX-RATE TO 13
    ADD 3 TO X
    MULTIPLY UNIT-PRICE BY QUANTITY
    DIVIDE CURRENT-ASSETS BY CURRENT-LIABILITIES.

Structured English Verbs

- GET (or ACCEPT or READ)
- PUT (or DISPLAY or WRITE)
- FIND (or SEARCH or LOCATE)
- ADD
- SUBTRACT
- MULTIPLY
- DIVIDE
- COMPUTE
- DELETE
- FIND
- VALIDATE
- MOVE
- REPLACE
- SET
- SORT

Structured English Object

- Objects (the subject of the simple imperative sentences) should consist only of data elements that have been defined in the data dictionary or local terms.
- Local terms are terms (words) that are explicitly defined within an individual process specification; they are only known, relevant, and meaningful within that process specification
- Examples:
  daily-total = 0
  DO WHILE there are more **orders** in **ORDERS** with Invoice-date = today's date
  READ next **order** in **ORDERS** with **Invoice-date** = today's date
  DISPLAY to Accounting **invoice-number**, **customer-name**, **total-amount**
  daily-total = daily-total + **total-amount**
  ENDDO
  DISPLAY to Accounting daily-total

Structured English Constructs

- **IF** condition-1
  sentence-1
  ENDIF
  or
  IF condition-1
  sentence-1
  ELSE
  sentence-2
  ENDIF

- **DO CASE**
  CASE variable = value-1
  sentence-1
  CASE variable = value-n
  sentence-n
  OTHERWISE
  sentence-n+1
  ENDCASE

- **DO WHILE** condition-1
  sentence-1
  ENDDO

- **REPEAT**
  sentence-1
  UNTIL condition-1

Pre/post conditions

- It is a convenient way of describing the function that must be carried out by a process, without saying very much at all about the algorithm or procedure that will be used.
- Useful when:
  1. The user has a tendency to express the policy carried out by a bubble in terms of a particular, idiosyncratic algorithm that he or she has been using for decades.
  2. The systems analyst is reasonably sure that there are many different algorithms that could be used.
  3. The systems analyst wants to let the programmer explore several such algorithms, but does not want to get involved in such details himself, and *especially* does not want to engage in arguments with the user about the relative merits of such algorithms.

Pre/post conditions example

PROCESS SPECIFICATIONCOMPUTE SALES TAX

Precondition 1

SALE-DATA occurs with ITEM-TYPE matching an ITEM-CATEGORY in TAX-CATEGORIES

Postcondition 1

SALES-TAX is set to SALE-AMOUNT * TAX-RATE

Precondition 2

SALE-DATA occurs with ITEM-TYPE that does not match an ITEM-CATEGORY in TAX-CATEGORIES

Postcondition 2

ERROR-MESSAGE is generated

Preconditions

- *Preconditions* describe all the things (if any) that must be true before the process begins operating.
- a guarantee from the user
  - "I guarantee that when this process is activated the following things will be true."
- Describe:
  1. *What inputs must be available.*
  2. *What relationships must exist between inputs or within inputs*.
  3. *What relationships must exist between inputs and data stores*.
  4. *What relationships must exist between different stores or within a single store*.

Postconditions

- *Postconditions* describe what must be true when the process has finished doing its job.
- A guarantee:
  - "I guarantee that when the process is finished the following will be true."
- Describe:
  1. *The outputs that will be generated or produced by the process*.
  2. *The relationships that will exist between output values and the original input values*.
  3. *The relationships that will exist between output values and values in one or more stores*.
  4. *The changes that will have been made to stores: new items added, existing items modifed, or existing items deleted*.

Contoh kode dari: COMPUTE SALES TAX

```
Procedure computeSalesTax(item-type: char ; sales-amount: integer);
var
    sales-tax : real;
    tax-categories = record
        item-category : char;
        tax-rate : real;
    end;
begin
    if (item-type = tax-categories.item-category) then
        sales-tax := sales-amount * tax-categories.tax-rate;
    else
        write(error-message);
end.
```

Sub Bahasan 4

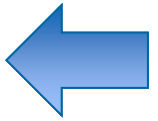
Implementasi PSPEC format Decision Table ke Pascal/C

Decision tables

- If the decisions are based on several different variables (e.g., input data elements), and if those variables can take on many different values, then the logic expressed by structured English or pre/post conditions is likely to be so complex that the user won't understand it.
- Created by listing all the relevant variables (sometimes known as *conditions* or *inputs*) and all the relevant actions on the left side of the table
- The variables and actions have been conveniently separated by a heavy horizontal line.

Decision tables example

	1	2	3	4	5	6	7	8
Age > 21	Y	Y	Y	Y	N	N	N	N
Sex	M	M	F	F	M	M	F	F
Weight > 150	Y	N	Y	N	Y	N	Y	N
Medication 1	X				X			X
Medication 2		X			X			
Medication 3			X			X		X
No medication				X			X	



BAGAIMANA MEMBUAT
IF BERTINGKAT
UNTUK 3 JENIS PENGOBATAN
UNTUK 8 KASUS YANG MUNGKIN
TSB?

Sub Bahasan 5

Implementasi Data Dictionary

Data Dictionary

- Typical data dictionary:
  - User = ID Num, Name, Address, Phone, Blood, username, password, Balance, Bank name, Bank account
  - Transaction = id Transaction, Date, Time, Total amount
  - name = courtesy-title + first-name + last-name
  - courtesy-title = [Mr. | Miss | Mrs. | Ms. | Dr. | Professor]
  - first-name = {legal-character}
  - last-name = {legal-character}
  - legal-character = [A-Z|a-z|0-9|'|-| |]

STRUKTUR DATA, FILE PROCESSING ATAU DATABASE

- Perlu pemeriksaan terhadap Data Dictionary di DFD
- Apakah cukup diimplementasikan dalam suatu Struktur Data yang akan diakses dalam suatu variable? (Misalnya : angka-angka di aplikasi Calculator)
- Atau Data Dictionary itu perlu disimpan dalam suatu file text atau file record, tanpa perlu software Basis Data? (Misalnya : output file Notepad)
- Atau kumpulan Data Dictionary itu perlu dibuatkan Basis Datanya menggunakan Software DBMS (Sudah dibahas di Pertemuan II)

Contoh Data Dictionary aplikasi Calculator

Number1 : integer

Number2 : integer

Result : integer

```
VAR  number1 : integer;
     number2 : integer;
     result : integer;
```

Contoh Data Dictionary aplikasi Notepad

- File : file of text
- VAR file : file of text (?);
- ➔ Coba eksplor kembali Pascal

Data Dictionary yang perlu Database

- ORDER : OrderID, CustID, Tgl, Total
- ORDERDetail : OrderID, ProdukID, Quantity
- PRODUK : ProdukID, ProdukName, Satuan, HrgSat, Stock)
- CUSTOMER : CustID, CustName, CustEmail, CustNoHP

- **LIHAT KEMBALI PEMBAHASAN SQL – CREATE TABLE DI PERTEMUAN KEDUA**

6. LATIHAN DI KELAS

DETERMINE TRIANGLE

Dari masukan 3 bilangan a, b, c bebas (boleh bulat, boleh pecahan), segitiga apa dapat dibangun?

- Jika ada yang negatif atau 0, tidak ada segitiga dapat dibangun.
- Jika bilangan yang terbesar lebih besar atau sama dengan penjumlahan dua bilangan lainnya yang lebih kecil, tidak ada segitiga dapat dibangun.
- Jika a=b atau b=c atau a=c (namun tidak sama dengan salah satu yang lain maka segitiga SAMA KAKI.
- Jika a=b dan b=c maka segitiga SAMA SISI.
- Jika kuadrat bilangan terbesar = penjumlahan dari kuadrat dua bilangan lainnya, maka SEGITIGA SIKU-SIKU.
- Jika bukan sama kaki, sama sisi atau siku-siku, namun bilangan terbesarnya lebih kecil daripada penjumlahan dua bilangan lainnya, maka SEGITIGA BEBAS.

SOAL

- 1. Buat untuk masukan bilangan bulat
- 2. Buat untuk bilangan pecahan dengan ketelitian 0.01 (1%) .Ketelitian 1% berarti selisih panjang sisi-sisinya tidak lebih dari 1%. Selisih lebih dari 1% dianggap beda. Selisih 1% atau kurang dianggap sama.

Soal 2

Buatkan kode program dalam bahasa pascal/C/Java untuk narasi program menghitung biaya servis komputer dalam bentuk *structured English* berikut ini:

DO CASE
CASE computer-amount = 1 **OR** 2
set **base-fee** = $50 and **additional-fee** = 0
CASE computer-amount = 3 to 10
set **base-fee** = $100 and **additional-fee** = $10 per peripheral
CASE computer-amount > 10
set **base-fee** = $500 and **additional-fee** = $10 per peripheral
IF service-time is **NOT** in business hours
base-fee is doubled
IF customer is willing to drop off and pick up
total-base-fee is reduced to one-half

Soal 3

Buatlah kode program dalam bahasa pascal/C/Java untuk narasi program menghitung biaya servis komputer dalam bentuk *pre/post conditions* berikut ini:

Precondition 1
Customer arrives with **account-number** matching an account number in **ACCOUNTS**, whose **status-code** is set to “valid.”
Postcondition 1
Invoice is produced containing **account-number** and **amount-of-sale**.
Precondition 2
Precondition 1 fails for any reason (**account-number** can’t be found on **ACCOUNTS** or **status-code** is not equal to “valid”).
Postcondition 2
Error message is produced.

7. TUGAS MINGGU INI

- Buat Contoh Implementasi PSPEC dari Kuliah APPL dulu atau dari kasus lain sebanyak jumlah member Group

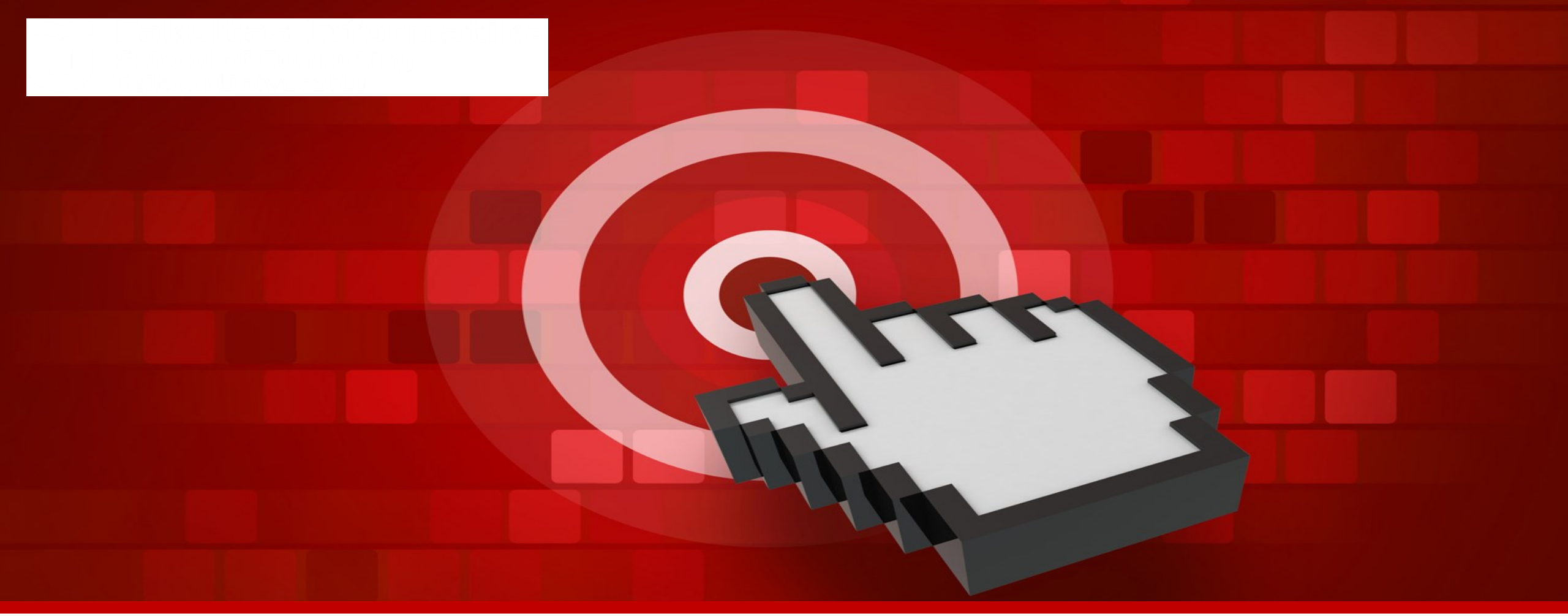
  ke Bahasa Pascal, C atau PHP

  Masing-masing member Group membuat code 1 PSPEC.

- Simpan Dokumen Di Github

References

1. Ed. Yourdon. Just Enough Structured Analysis. 2006. www.yourdon.com



THANK YOU