ITTelkom
Surabaya
Solution for The Nation

INTRODUCTION TO INFORMATION TECHNOLOGY

PROGRAM STUDI S1 TEKNOLOGI INFORMASI

FAKULTAS TEKNOLOGI INFORMASI DAN INDUSTRI

INSTITUT TEKNOLOGI TELKOM SURABAYA

HOME WORK

- Language Types & Early Languages
- Compiled vs Interpreted
- Errors
- FORTRAN and COBOL
- Structured Programming
- Object-Oriented Programming
- Byte Code and Input/Output Instruction
- Assignment Statements
- Blocks & Types of Loops
- Infinite Loops
- Subroutines
- Simple C Program with a Function

PROGRAMMING

PROGRAMING

- Dibalik ide sebuah pemrograman adalah spesifikasi algoritma atau solusi untuk menyelesaikan masalah, pemrograman biasanya :
  - Ditulis dalam Bahasa pemrograman komputer
  - Tersedia dalam bentuk yang dapat dieksekusi, seperti
    - Memerlukan kompilasi program dan mensyaratkan bahwa program tidak memiliki syntax error
    - Membutuhkan penulisan instruksi program dalam sesi dengan interpreter
- Pemrograman memiliki sebuah tantangan, antara lain:
  - Harus berpikir secara logis, dan dapat menuliskan kondisi dalam bentuk logika
  - Harus memiliki solusi, ditulis dalam langkah-langkah spesifik dan dalam urutan yang tepat
  - Harus bisa memahami dan mengerti syntax pada Bahasa pemrograman

LANGUAGE TYPES

- Machine Language
  - Satu-satunya Bahasa yang dapat dieksekusi langsung oleh komputer
  - Cryptic (0s dan 1s), simple instruction
- Assembly Language
  - Sedikit lebih mudah daripada machine language, tetapi instruksi yang dijalankan masih sederhana
    - Membutuhkan assembler untuk menerjemahkan kode dari assembly ke machine language
- High Level Language
  - Instruksi lebih bervariasi, berurusan dengan konsep seperti kondisi, seleksi, dan pengulangan
  - Dua jenis penerjemah program : compilers dan interpreters

COMPARING THE TYPES

```
1111 0000000000001          Load #1          int a = 1, b = 5, c;
0010 1010100110000          Store a          while(a < b)
1111 0000000000101          Load #5          {
0010 1010100110001          Store b                  c = c + 2;
0001 1010100110000    Top:  Load a                   b = b - 1;
0100 1010100110001          Subt b           }
1000 0000000000000          Skipcond 00      printf("%d", c);
1001 0001000101010          Jump exit
0001 1010100110011          Load c
1110 0000000000010          Add #2
0010 1010100110011          Store c
0001 1010100110001          Load b
1101 0000000000001          Subt #1
0010 1010100110001          Store b
1001 0001000100010          Jump top
0001 1010100110011    Exit: Load c
0101 0000000000000          Output
```

COMPILED VS INTERPRETED

- Compiler merupakan program sistem yang digunakan sebagai alat bantu dalam pemrograman
- Interpreted merupakan software yang mampu mengeksekusi code program lalu menterjemahkannya ke dalam Bahasa mesin, sehingga mesin melakukan instruksi yang diminta oleh programmer tersebut

ERRORS

- Syntax Error
- Run-time error
- Logical Error

FORTRAN AND COBOL

- FORmula TRANslator (FORTRAN), merupakan Bahasa high level pertama dikembangkan pada tahun 1957-1958
  - Programmer akan menentukan program sebagai rumus matematika (bersama dengan I/O statements, GO TO statements, dan basic counting looop)
  - Compiler akan menerjemakan dari sebuah kode menjadi Bahasa mesin
  - Fitur yang kurang seperti jenis karakter/string, pernyataan if-then/if-then-else
- COmmon Business – Oriented Language (COBOL)
  - Dikembangkan tak lama setelah FORTRAN oleh departemen pertahanan untuk mendukung aplikasi bisnis
- Terbagi atas dua bagian yakni definisi data dan definisi kode
- Mengalami banyak kegagalan dikarenakan memiliki sedikit kemmampuan matematika, semua variable bersifat global dan tanpa array

OTHER EARLY LANGUAGES

- LISP – LISt Processing developed for AI use
  - Menekankan pada memori, daftar, rekursi yang dialokasikan secara dinamis
  - Merupakan Interpreted Language
- ALGOL – Algorithmic Language
  - Memperkenalkan if-then-else statements dan block
  - Mengijinkan recursion
- PL/I – IBM language to "replace" FORTRAN
  - Berisi fitur-fitur dari hamper setiap Bahasa sebelumnya
  - Menambahkan kemampuan exeption handling
- SIMULA – for simulations
  - Class Definition untuk pemrograman berorientasi objek
- SNOBOL – StrinNg Oriented and SymBOlic Language
  - String machine Language

STRUCTURED PROGRAMMING

- Merupakan Bahasa pemrograman yang mendukung pembuatan program sebagai kumpulan prosedur
  - Pemrograman sering dibatasi dalam Bahasa yang harus mereka gunakan karena hardware yang tersedia, mengandalkan misalnya pada GOTO statements
- ALGOL 68, penerus ALGOL, memperkenalkan pemrograman terstruktur
  - Menggunakan structured control statements seperti if-then-else statements dan while loops
- Turunan ALGOL 68 adalah Bahasa C dan Pascal

FORTRAN IF STATEMENT

```
        READ (*,*) I, J, K
        If (I-J), 10, 20, 30
10      If(K) 30, 40, 50
20      I=J*K
        If(I-25) 40, 50, 60
30      J=K*3
        GO TO 20
40      K=K+1
        GO TO 10
50      WRITE (*,*) I, J, K
        GO TO 70
60      WRITE (*, *) J, K
70      END
```

OBJECT-ORIENTED PROGRAMMING LANGUAGES

- Dikembangkan pertama kali pada tahun 1980 di Smalltalk
- Paradigma pemrograman yang berorientasikan pada objek
- Semua data dan fungsi didalam paradigma ini dikemas dalam *class* atau *object*
- Setiap objek dapat menerima pesan, memproses data, dan mengirim pesan ke objek lainnya
- *Class* dibangun secara hirarki sehingga kelas anak dapat mewarisi kelas induk
  - Sehingga memungkinkan membangun kode diatas kode orang lain
- Menggunakan Bahasa pemrograman Java

BYTE CODE

- Dalam sebagian besar Bahasa yang dikompilasi, harus mengkompilasi program untuk setiap platform target
  - Kode yang dapat dieksekusi tidak portable untuk platform lain sehingga menempatkan beban pada programmer untuk mengkompilasi program untuk banyak platform
- Bahasa pemrograman Java menggunakan byte code sebagai Bahasa perantara
  - Byte code tidak dapat dieksekusi secara langsung
  - Interpreter kemudian dapat menjalankan program byte code hal ini tidak membuat running compiled lebih lambat
- Java byte code interpreters dibuat oleh Java Vortual Machine
- Konsep byte code dimasukkan kedalam platform Microsoft's.Net sehingga objek dari setiap Bahasa .net dapat digunakan sebagai objek oleh Bahasa .net lainnya

INPUT/OUTPUT INSTRUCTIONS

- Input – mendapatkan informasi dari user (atau file)
- Output – menghasilkan nilai yang disimpan pada variable untuk dilihat oleh user (atau disimpan dalam file)
- Instruksi Input biasanya didahului oleh pernyataan output yang menyediakan prompt pengguna

EXAMPLES

```
   OPEN(1,FILE='input.dat', ACCESS='DIRECT', STATUS='OLD')
   READ(1,50) X, Y, Z
50 FORMAT(I2, F6.2, F6.2)
```

```
FILE *file;
file = fopen("input.dat", "r");
fscanf("%d %f %f", &x , &y, &z);
```

```
var input : text;
…
Assign(input, 'input.dat');
Reset(input);
Readln(input, x, y, z);
```

```
try {
        BufferedReader input = new BufferedReader(new FileReader("input.dat"));
        data=input.readLine( );
        StringTokenzier token = new StringTokenizer(data, " ");
        x=Integer.parseInt(token.nextToken( ));
        y=Float.parseFloat(token.nextToken( ));
        z=Float.parseFloat(token.nextToken( ));
}
catch(IOException e) { }
```

ASSIGNMENT STATEMENTS

- Instruksi yang menyimpan nilai ke dalam variable (menetapkan nilai ke variabel)
- Format : Variable = Value

Language	Assignment Operator	Equality Operator	Not Equal Operator
Ada	: =	=	/=
C/C++/Java, Python, Ruby	=	= = (2 equal signs)	! =
COBOL	assign	Equals (or =)	Is not equal to (or NOT =)
FORTRAN	=	.EQ.	.NE.
Pascal	: =	=	<>
Perl	=	eq (strings), = = (numbers)	ne (strings), ! = (numbers)
PL/I	=	=	<>

CONTROL STATEMENTS

- Kode yang dijalankan dalam “straight line” atau berurutan
- Untuk mengubah perilaku ini, kita harus menggunakan instruksi pengambilan keputusan
  - Selection statements - berdasarkan pada sutau kondisi, pilih pernyataan yang akan dieksekusi
  - Iteration Statements
    - Counting loop
    - Conditional loop
    - Iterator loop
- Tanpa control statements, program akan melakukan hal yang sama setiap kali dijalankan

CONDITIONS AND SELECTION STATEMENTS

- Condition : penggabungan nilai variable dan nilai lainnya
  - Variable, expression, literal value
- Digunakan untuk membuat keputusan
- Menggunakan relational operator <,>,=,!=,<=,>=
- Complex condition mengkombinasi dua atau lebih kondisi dengan BOOLEAN operator (AND, OR, NOT)
- Pilih tindakan sesuai nilai dan kondisi
  - If-then: 1-way selection
  - If-then-else: 2-way selection
  - Nested if-then-else: n-way selection
  - Case/switch: n-way selection

EXAMPLE IF-THEN-ELSE STATEMENTS

```
If x = y or x = z
x := x + 1
Else
x := x - 1;
```

```
if ( x = = y || x = = z)
        x = x + 1;
else
        x = x - 1;
```

```
        IF ( X .EQ. Y .OR. X .EQ. Z) GO TO 10
        X = X - 1
        GO TO 20
10      X = X + 1
20 ….
```

```
If grade >= 90 then letter := 'A'
                Else if grade >= 80 then letter := 'B'
                        Else if grade >= 70 then letter := 'C'
                                Else if grade >= 60 then letter := 'D'
                                        Else letter := 'F';
```

BLOCKS

- Pada Bahasa seperti Pascal dan C, klausa then dan else kecuali hanya 1 istruksi
  - Jika memiliki beberpa instruksi, maka harus melampirkannya dalam block

```
If x > = y
        Then x: = z;
             y: = 0;
q : = x + y;
```

TYPES OF LOOPS

- Conditional
- Counting
- Iterator

EXAMPLE LOOP

```
While x > y
      begin
            …
      end;
```

```
while (x > y) {
      …
}
```

```
for I : = 1 to 10 do
      begin
            …
      end;
```

```
sum = 0;
for(i=0;i<=10;i=i+1)
      sum = sum + i;
```

```
Sum := 0;
For I := 1 to 10 do
      Sum := Sum + I;
```

```
while(x > 0)
      x = x/2;
```

INFINITE LOOPS

```
while(x > y)
        x = x + 1;
```

```
while(x > 0)
        printf("%d\n", x);
        x = x - 1;
```

```
while(x > y) {
        printf("%d\n", x);
        x = x - 1;
}
```

SUBROUTINES

- Program yang panjang membuat sulit untuk dimengerti, sulit ditulis dan sulit untuk debug
- Dua bagian dari kode untuk subroutine
- Pada banyak kasus, subroutine membutuhkan data dari fungsi panggilan

SIMPLE C PROGRAM WITH A FUNCTION

```
void foo( ) {
      printf("are we in foo? Yes!");
}

void main( ) {
      printf("Hello world. We are about to enter foo.");
      foo( );
      printf("We have now returned from foo.");
}
```

```
Hello world. We are about to enter foo.
are we in foo? Yes!
We have now returned from foo.
```

```
void determineReciprocal(float x) {
      if(x==0)
            printf("There is no reciprocal of 0");
      else printf("The reciprocal is %f", 1.0/x);
}
```

OTHER INSTRUCTIONS

- Graphic Instructions
- String Operation
- Operating system operations
- Random number generator operations
- Error Handling operations
- Variable declarations

```
Var x, y : integer;
Var z : real;

int x, y;
float z;

var x, y, z;
```

BASH : VARIABLES

- Variable tidak dideklarasikan
- Variabel mendapatkan nilai berasal dari input atau assignment statements
- Penyimpanan variable strings atau integers (bukan float)
- Variabel adalah nama yang diberikan ke beberapa lokasi penyimpanan yang menyimpan nilai
- Untuk mendapatkan nilai dari variable, menggunakan $VAR sebagai $NAME