IF1210 Algoritma dan Pemrograman 1

Pengenalan Bahasa C

Tim Pengajar IF1210

Sekolah Teknik Elektro dan Informatika

Bahasa C – Sejarah singkat

- Dikembangkan oleh Dennis Ritchie dan Brian Kernighan pada awal 1970an.
- Awalnya berkembang di lingkungan Unix
  - ±90% sistem operasi Unix ditulis dalam bahasa C
- Standar Bahasa C yang ada
  - Definisi Kernighan dan Ritchie (K&R)
  - ANSI-C → dipakai dalam kuliah ini
  - Definisi AT&T (untuk C++)
- Versi C pada sistem operasi Windows:
  - Lattice C, Microsoft C, Turbo C, dll.

Bahasa C – Penggunaan

- Bahasa C banyak digunakan untuk:
  - Membuat sistem operasi dan program-program sistem (Linux, Unix, Windows),
  - Pemrograman yang dekat dengan perangkat keras (misal: kontrol peralatan),
  - Membuat toolkit & runtimes (JRE, Cpython dll)
  - Database Systems (MySQL, Postgres, SQLite)
- Kelebihan Bahasa C:
  - Kode yang compact & efisien (performant).
- Kekurangan Bahasa C:
  - Kurang readable dibandingkan bahasa tingkat tinggi lain.
  - Tidak mencegah programmer melakukan kekeliruan dalam manajemen memori.

Bahasa C – Beberapa catatan

Blok Program

Sekumpulan kalimat (*statement*) C di antara kurung kurawal { dan }

Contoh:

```
if (a > b) {
  printf("a lebih besar dari b\n");
  b += a;
  printf("sekarang b lebih besar dari a\n");
}
```

Komentar program

Dituliskan di antara tanda /* dan */

Alternatif: // hingga *end of line* (biasanya \n)

Bahasa C adalah bahasa yang ***case-sensitive***

Berikutnya: sintaks Bahasa C

- Type, konstanta, variabel, assignment
- Input/output
- Analisis kasus

- Pada materi-materi selanjutnya, sintaks Bahasa C akan diajarkan bertahap sesuai materi

Type, konstanta, variabel, assignment

Konstanta, variabel

Notasi Algoritmik

Konstanta

constant <nama>:<type>=<harga>

Deklarasi variabel

<nama>: <type>

Inisialisasi/assignment

<nama> ← <harga>

Bahasa C

```
// Konstanta
// 1) Dengan const:
const [<type>] <nama> = <harga>;
// 2) Dengan preprocessor/macro
#define <nama> <harga>

// Deklarasi Variabel
<type> <nama>;


// Inisialisasi/assignment
<nama> = <harga>;
// Deklarasi sekaligus inisialisasi
<type> <nama> = <harga>;
```

Konstanta, variabel: contoh

Notasi Algoritmik

```
Program Test
{Tes konstanta, variabel, assignment,
inisialisasi...}

KAMUS
  {Konstanta}
  constant LIMA: real = 5.0
  constant PI: real = 3.14

  {Variabel}
  luas, r: real
  i: integer

ALGORITMA
  i ← 1          {Inisialisasi}
  r ← 10.0       {Inisialisasi}
  ...
```

Bahasa C

```c
/* File: test.c - Program Test */
/* Tes konstanta, variabel, assignment,
inisialisasi... */

/* KAMUS */
#define LIMA 5.0 // deklarasi konstanta cara-1

int main () {
  /* KAMUS */
  /* Konstanta */
  const float PI = 3.14; // dekl. konstanta cara-2

  /* Variabel */
  float luas, r;
  int i = 1; /* Deklarasi + inisialisasi */

  /* ALGORITMA */
  r = 10.0; /* Inisialisasi */
  ...
  return 0;
}
```

Catatan: C preprocessor

- Pada contoh sebelumnya, konstanta LIMA dideklarasikan menggunakan **macro**

```
#define LIMA 5.0
```

- *Compiler* terlebih dahulu akan menjalankan *preprocessor* sebelum kompilasi dilakukan.
  - Salah satu tugas *preprocessor* adalah mengganti kemunculan *macro* sesuai deklarasinya,
  - Pada contoh tadi, semua kemunculan LIMA pada *source code* akan di-*replace* dengan 5.0 sebelum dilakukan kompilasi.

Assignment

Notasi Algoritmik

Assignment

```
<nama1> ← <nama2>
<nama> ← <konstanta>
<nama> ← <ekspresi>
nama1 ← nama1 <opr> nama2
```

Contoh:

```
luas ← PI * r * r
x ← x * y
i ← i + 1

i ← i - 1
```

Bahasa C

```
// Assignment
<nama1> = <nama2>;
<nama> = <konstanta>;
<nama> = <ekspresi>;
nama1 = nama1 <opr> nama2;
// Compound Assignment:
nama1 <opr>= nama2;

// Contoh:
luas = PI * r * r;
x *= y;
i++;
++i; /* Apa bedanya? */
i--;
--i; /* Apa bedanya? */
```

Type data karakter (Bahasa C)

Contoh deklarasi:

```
char cc;
```

Contoh literal:

```
'c'       → karakter c
'0'       → karakter 0
'\013'    → karakter vertical tab
```

Jenis-jenis character:

```
[signed] char
unsigned char
```

char pada dasarnya adalah integer 1 byte (8 bits)

Type data bil. bulat (Bahasa C)

Contoh deklarasi: `int i; short int j;`

Contoh literal: `1` `2` `0` `-1`

Jenis-jenis integer:

`[signed] int`

Natural size of integers in host machine, e.g. 32 bits

No shorter than `short int`, no longer than `long int`

`[signed] short [int]`

Min. 16 bits of integer

`[signed] long [int]`

Min. 32 bits of integer

`unsigned int`, `unsigned short [int]`, `unsigned long [int]`

0 and positive integers only.

Type data bil. real (Bahasa C)

Contoh deklarasi: `float f1; double f2;`

Contoh literal: `3.14  0.0  1.0e+2  5.3e-2`

Jenis-jenis real:

- `float`
  - Single-precision floating point
  - 6 digits decimal
- `double`
  - Double-precision floating point
  - Eg. 10 digits decimal
- `long double`
  - Extended-precision floating point
  - Eg. 18 digits decimal

Type data boolean (Bahasa C)

C tidak menyediakan type boolean

Ada banyak cara untuk mendefinisikan boolean

Cara 1 – Digunakan nilai integer untuk menggantikan nilai *true* & *false*:

true = nilai bukan 0

false = 0

Cara 2 – Definisikan sebagai konstanta:

```
#define boolean unsigned char
#define TRUE 1
#define FALSE 0
```

Type data boolean (Bahasa C)

Cara 3 – Definisikan dalam file *header*, misal: `boolean.h` → digunakan sebagai standar dalam mata kuliah ini

```
/* File: boolean.h */
/* Definisi type data boolean */

#ifndef BOOLEAN_H
#define BOOLEAN_H

#define boolean unsigned char
#define TRUE 1
#define FALSE 0

#endif
```

Contoh penggunaan:

```
/* File: boolean_test.c */
#include "boolean.h"

int main () {
    /* KAMUS */
    boolean found;
    ...

    /* ALGORITMA */
    found = TRUE;
    ...

    return 0;
}
```

Type data string (Bahasa C)

Dalam C, string adalah *pointer* ke *array* dengan elemen char

Contoh konstanta string: `"Ini sebuah string"`

- Konstanta string berada di antara *double quotes* `"…"`

Namun, string ≠ array of char

- Representasi internal string selalu diakhiri character `'\0'`, sedangkan array of character tidak
- Jadi, string `"A"` sebenarnya terdiri atas dua buah character yaitu `'A'` dan `'\0'`

Type data string (Bahasa C)

Contoh deklarasi (+ inisialisasi):

```
char str1[] = "ini string"; /* deklarasi dan inisialisasi */
char str2[37]; /* deklarasi string sepanjang 37 karakter */
char *str3;
```

Contoh cara assignment nilai:

```
strcpy(str2, "pesan apa"); /* str2 diisi dgn "pesan apa" */

str3 = (char *) malloc (20 * sizeof(char)); /* alokasi memori terlebih dahulu */

strcpy(str3, ""); /* str3 diisi dengan string kosong */

/* HATI-HATI, cara di bawah ini SALAH! */
str3 = "Kamu pesan apa";
```

Type enumerasi

Notasi Algoritmik

```
KAMUS
  { Definisi type }
  type Hari: (senin, selasa, rabu,
kamis,
              jumat, sabtu, minggu)



  { Deklarasi variabel }
  h: Hari

ALGORITMA
  { Assignment }
  h ← senin
```

Bahasa C

```c
/* KAMUS */
/* Definisi type */
typedef enum {
  senin, selasa, rabu, kamis,
  jumat, sabtu, minggu
} hari; /* senin=0, selasa=1, dst. */

int main() {
  /* Deklarasi variabel */
  hari h;

  /* ALGORITMA */
  /* Assignment */
  h = senin; /* h = 0 */
}
```

Type bentukan (tuple)

Notasi Algoritmik

```
KAMUS
  { Definisi Type }
  type namatype:
     < elemen1: type1,
       elemen2: type2,
       ... >


  { Deklarasi Variabel }
  nmvar1: namatype
  nmvar2: type1 {misal}

ALGORITMA
  { Akses Elemen }
  nmvar2 ← nmvar1.elemen1
  nmvar1.elemen2 ← <ekspresi>
```

Bahasa C

```c
/* KAMUS */
/* Definisi Type */
typedef struct NamaType {
  type1 elemen1;
  type2 elemen2;
  ...
} namatype;

int main() {
  /* Deklarasi Variabel */
  namatype nmvar1;
  type1 nmvar2; /*misal*/

  /* ALGORITMA */
  /* Akses Elemen */
  nmvar2 = nmvar1.elemen1;
  nmvar1.elemen2 = <ekspresi>;
}
```

Operator

Notasi algoritmik

Ekspresi Infix:

`<opn1> <opr> <opn2>`

Contoh: `x + 1`

Operator Numerik:

```
+
-
*
/
div
mod
```

Bahasa C

Ekspresi Infix:

`<opn1> <opr> <opn2>`

Contoh: `x + 1`

Operator Numerik:

```
+
-
*
/    /*hasil float*/
/    /*hasil integer*/
%
++   /*increment*/
--   /*decrement*/
```

Operator

Notasi Algoritmik

Operator relasional:

>	≥
<	≤
=	≠

Operator logika:

and

or

not

Bahasa C

Operator relasional:

>	>=
<	<=
==	!=

Operator logika:

&&

||

!

Operator bit:

```
<<     /*shift left*/
>>     /*shift right*/
&      /*and*/
|      /*or*/
^      /*xor*/
~      /*not*/
```

Input/Output

Input

Notasi Algoritmik

```
input(<list-nama>)
```

Contoh:

```
input(x) {x integer}
input(x,y)
input(f) {f real}
input(s) {s string}
```

Bahasa C

```
scanf("<format>", <list-nama>);
// Contoh:
scanf("%d",&x); /*x integer*/
scanf("%d %d", &x, &y);
scanf("%f",&f); /*f real*/
scanf("%s",s);  /*s string*/
// format-format sederhana:
%d untuk type integer
%f untuk type real
%c untuk type character
%s untuk type string
```

Output

Notasi Algoritmik

```
output(<list-nama>)
```

Contoh:

```
output("Nomor: ",x," dapat nilai ", y)
    {x, y integer}
output(x, y)
output("Contoh output")
output("Namaku: ", nama)
output(f) {f real}
output(cc) {cc character}
```

Bahasa C

```
printf("<format>", <list-nama>);
// Contoh:
printf("Nomor: %d dapat nilai %d",x,y);
    /* x, y integer */
printf("%d %d", x, y);
printf("Contoh output");
printf("Namaku: %s", nama);
printf("%f",f); /* f real */
printf("%c",cc); /* cc character */
/* Format-format sederhana sama seperti
pada input */
```

Analisis Kasus

Analisis Kasus

Notasi Algoritmik

Satu Kasus:

```
if kondisi then
        aksi
```

Dua Kasus Komplementer:

```
if kondisi-1 then
        aksi-1
else { not kondisi-1 }
        aksi-2
```

Bahasa C

```
// Satu Kasus:
if (kondisi) {
        aksi;
}
```

```
// Dua Kasus Komplementer:
if (kondisi-1) {
        aksi-1;
} else { /* not kondisi-1 */
        aksi-2;
}
```

Analisis Kasus (> 2 kasus)

Notasi Algoritmik

```
depend on nama
        kondisi-1: aksi-1
        kondisi-2: aksi-2
        ...
        kondisi-n: aksi-n
```

```
depend on nama
        kondisi-1: aksi-1
        kondisi-2: aksi-2
        ...
        else     : aksi-else
```

Bahasa C

```
if (kondisi-1) {
        aksi-1;
} else if (kondisi-2) {
        aksi-2;
...
} else if (kondisi-n) {
        aksi-n;
}
```

```
if (kondisi-1) {
        aksi-1;
} else if (kondisi-2) {
        aksi-2;
...
} else { aksi-else; }
```

Analisis Kasus (> 2 kasus)

Jika setiap kondisi dapat dinyatakan dalam bentuk:
nama = const-exp (const-exp adalah suatu ekspresi konstan),
maka dapat digunakan *statement* **switch**.

Notasi Algoritmik

```
depend on nama
        nama=const-exp-1: aksi-1
        nama=const-exp-2: aksi-2
        ...
        else            : aksi-else
```

Bahasa C

```
switch (nama) {
        case const-exp-1:
    aksi-1;
          [break;]
        case const-exp-2:
    aksi-2;
          [break;]
        ...
        default:
    aksi-else;
          [break;]
};
```

Latihan Soal

Soal 1

- Translasikan algoritma dalam notasi algoritmik berikut ke Bahasa C.

```
Program KelilingLingkaran
{ Menghitung keliling lingkaran berdasarkan masukan jari-jari }
```

```
KAMUS
    constant PI : real = 3.14159
    R : real
    Kel : real
```

```
ALGORITMA
    input(R)
    Kel ← 2 * PI * R
    output(Kel)
```

Soal 2

- Dalam Fisika, jarak (s) dapat dihitung berdasarkan kecepatan (v) dan waktu tempuh (t), yaitu: s = v * t
- Buatlah program dalam Bahasa C untuk menghitung jarak (dalam m) berdasarkan masukan kecepatan (dalam m/s) dan waktu (dalam s).

Soal 3

- Sebuah toko menjual kelereng. Berikut adalah tabel harga kelereng berdasarkan warnanya:

Warna kelereng	Harga 1 butir (dalam ratusan rupiah)
Merah	10
Hijau	15
Kuning	20

- Seorang anak membeli kelereng sejumlah m kelereng merah, h kelereng hijau, dan k kelereng kuning. Asumsikan m >= 0, h >= 0, k >= 0.
- Buatlah program dalam Bahasa C yang menerima masukan m, h, dan k dan menghitung serta menampilkan berapa yang harus dibayarkan anak itu.

Soal 4

- Sebuah titik di atas bidang kartesian terdiri atas sebuah absis (x, bertipe real) dan ordinat (y, bertipe real)
- Buat program dalam Bahasa C yang membaca sebuah data bertipe titik (misalnya P) dan menampilkannya ke layar dalam format: (x,y).

Soal 5

- Buatlah program dalam notasi algoritmik dan Bahasa C yang menerima masukan 2 buah bilangan, misalnya X dan Y, dan menuliskan ke layar bilangan terbesar di antara X dan Y.