

Bahasa C

IF2110/IF2111 – Algoritma dan Struktur Data
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung

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 – Kegunaan

Bahasa C banyak digunakan untuk:

- Membuat sistem operasi dan program-program sistem,
- Pemrograman yang dekat dengan perangkat keras (misal: kontrol peralatan),
- Membuat *toolkit*,
- Menulis program aplikasi (misalnya dBase, Wordstar, Lotus123).

Kelebihan Bahasa C:

- Kode yang *compact* & efisien.

Kekurangan Bahasa C:

- Kurang *readable* dibandingkan bahasa tingkat tinggi lain.

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***

Translasi: notasi algoritmik ke bahasa C

```
{ Modul ADT Time }  
type Time: < hours: integer[0..23],    { 0 ≤ hours ≤ 23 }  
             minutes: integer[0..59],    { 0 ≤ minutes ≤ 59 }  
             seconds: integer[0..59] > { 0 ≤ seconds ≤ 59 }  
  
{ Konstruktor: membentuk Time dari komponen-komponennya: h sebagai hours, m sebagai  
  minutes, dan s sebagai seconds. }  
procedure CreateTime(output t: Time, input h: integer[0..23],  
                    input m: integer[0..59], input s: integer[0..59])  
{ Mendapatkan komponen hours dari T }  
function getHours(T: Time) → integer[0..23]  
{ Mendapatkan komponen minutes dari T }  
function getMinutes(T: Time) → integer[0..59]  
{ Mendapatkan komponen seconds dari T }  
function getSeconds(T: Time) → integer[0..59]  
{ Selisih antara dua Time, dalam satuan detik }  
function difference(start: Time, end: Time) → integer
```

Translasi: notasi algoritmik ke bahasa C

File: src/time.h

```
/* Modul ADT Time */
#ifndef TIME_H
#define TIME_H

typedef struct Time { int hours;
                     int minutes;
                     int seconds; } time;

/* (Komentar-komentar tidak dituliskan untuk mempersingkat.) */
void CreateTime(time *t, int h, int m, int s);
int getHours(time t);
int getMinutes(time t);
int getSeconds(time t);
int difference(time start, time end);

#endif /* TIME_H */
```

Translasi: notasi algoritmik ke bahasa C

```
function difference(start: Time, end: Time) → integer  
{ Menghasilkan selisih antara dua Time start dan end, dengan syarat  $\text{start} \leq \text{end}$ . }
```

```
{ EXPECT
```

```
    CreateTime(start, 1,2,3); CreateTime(end, 2,3,4)
```

```
    ⇒ difference(start,end) = 3661
```

```
    CreateTime(start, 2,3,4); CreateTime(end, 2,3,4)
```

```
    ⇒ difference(start,end) = 0
```

```
    CreateTime(start, 2,3,4); CreateTime(end, 1,2,3)
```

```
    ⇒ difference(start,end) = tak terdefinisi }
```

KAMUS LOKAL

startSec, endSec: integer

ALGORITMA

```
startSec ← getHours(start)*60*60 + getMinutes(start)*60 + getSeconds(start)
```

```
endSec   ← getHours(end)*60*60   + getMinutes(end)*60   + getSeconds(end)
```

```
→ endSec-startSec
```

Translasi: notasi algoritmik ke bahasa C

File: src/time.c

```
#include "time.h"

int difference(time start, time end) {
    /* Menghasilkan selisih antara dua Time start dan end, dengan syarat start ≤ end. */
    /* KAMUS LOKAL */
    int startSec, endSec;
    /* ALGORITMA */
    startSec = getHours(start)*60*60 + getMinutes(start)*60 + getSeconds(start);
    endSec   = getHours(end)*60*60   + getMinutes(end)*60   + getSeconds(end);
    return endSec - startSec;
}
```


Translasi: notasi algoritmik ke bahasa C

File: tests/check_time.c

```
#include <check.h>
#include "../src/time.h"

time t1, t2;

void setup() { CreateTime(&t1, 2,3,4); CreateTime(&t2, 1,2,3); }

START_TEST(time_difference) {
    ck_assert_int_eq(difference(t1,t2), 3661);
    ck_assert_int_eq(difference(t1,t1), 0);
} END_TEST
```

Berikutnya: sintaks Bahasa C

Type, konstanta, variabel, assignment

Input/output

Analisis kasus

Pengulangan

Subprogram

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

```
/* 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 \neq 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

```
/* 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

```
/* 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>;  
}
```


Type bentukan (contoh)

Notasi Algoritmik

KAMUS

```
{ Definisi Type }  
type point:  
  < x: integer,  
    y: integer >
```

```
{ Deklarasi Variabel }  
p: point  
bil: integer {misal}
```

ALGORITMA

```
{ Akses Elemen }  
bil ← p.x  
p.y ← 20
```

Bahasa C

```
/* KAMUS */  
/* Definisi Type */  
typedef struct Point {  
    int x;  
    int y;  
} point;  
  
int main() {  
    /* Deklarasi Variabel */  
    point p;  
    int bil; /* misal */  
  
    /* ALGORITMA */  
    /* Akses Elemen */  
    bil = p.x;  
    p.y = 20;  
}
```

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*/  
/*Lihat contoh di diktat*/
```

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;]  
};
```

Contoh

Diktat “Contoh Program Kecil Bahasa C”

Instruksi Kondisional

Pengulangan

Pengulangan

Notasi Algoritmik

Pengulangan berdasarkan kondisi berhenti:

```
repeat
    Aksi
until kondisi-stop
```

Pengulangan berdasarkan kondisi ulang:

```
while (kondisi-ulang) do
    Aksi
{not kondisi-ulang}
```

Bahasa C

```
do {
    Aksi;
} while (!kondisi-stop);
```

```
while (kondisi-ulang) {
    Aksi;
} /*not kondisi-ulang */
```

Pengulangan

Notasi Algoritmik

Pengulangan berdasarkan pencacah:

i traversal [Awal..Akhir]
 Aksi

Bahasa C

```
/* Jika Awal <= Akhir */  
for(i=Awal;i<=Akhir;i++) {  
    Aksi;  
}  
/* Jika Awal >= Akhir */  
for(i=Awal;i>=Akhir;i--) {  
    Aksi;  
}
```

Catatan:

```
for(exp1;exp2;exp3) {  
    Aksi;  
}  
ekivalen dengan:  
exp1;  
while (exp2) {  
    Aksi;  
    exp3;  
} /* !exp2 */
```

Pengulangan

Notasi Algoritmik

Pengulangan berdasarkan dua aksi:

iterate

Aksi-1

stop kondisi-stop

Aksi-2

Bahasa C

```
for(;;) {  
    Aksi-1;  
    if (kondisi-stop)  
        break;  
    else  
        Aksi-2;  
}
```

Contoh

Diktat “Contoh Program Kecil Bahasa C”

Pengulangan

Subprogram

Fungsi (Notasi algoritmik)

```
function NAMA F (param1: type1, param2: type2, ...) → type-hasil  
  { Spesifikasi fungsi }
```

KAMUS LOKAL

```
{ Semua nama yang dipakai dalam algoritma dari fungsi }
```

ALGORITMA

```
{ Deretan fungsi algoritmik:  
  pemberian harga, input, output, analisis kasus, pengulangan }  
  
{ Pengiriman harga di akhir fungsi, harus sesuai dengan type-hasil }  
→ hasil
```

Fungsi (Bahasa C)

```
type-hasil NAMA (type1 param1, type2 param2, ...) {  
    /* Spesifikasi fungsi */  
  
    /* KAMUS LOKAL */  
    /* Semua nama yang dipakai dalam algoritma dari  
    fungsi */  
  
    /* ALGORITMA */  
    /* Deretan fungsi algoritmik:  
    pemberian harga, input, output, analisis kasus,  
    pengulangan */  
  
    /* Pengiriman harga di akhir fungsi, harus sesuai  
    dengan type-hasil */  
    return hasil;  
}
```

Pemanggilan fungsi (Notasi algo.)

Program POKOKPERSOALAN

{ Spesifikasi: Input, Proses, Output }

KAMUS

{ semua nama yang dipakai dalam algoritma }

{ Prototype fungsi }

function NMAF ([list nama:type input]) → **type-hasil**

{ Spesifikasi fungsi }

ALGORITMA

{ Deretan instruksi pemberian harga, input, output, analisa kasus, pengulangan yang memakai fungsi }

{ Harga yang dihasilkan fungsi juga dapat dipakai dalam ekspresi }

nama ← NMAF ([list parameter aktual])

output (NMAF ([list parameter aktual]))

{ Body/Realisasi Fungsi diletakkan setelah program utama }

Pemanggilan fungsi (Bahasa C)

```
/* Program POKOKPERSOALAN */
/* Spesifikasi: Input, Proses, Output */
/* Prototype Fungsi */
type-hasil NMAF ([list <type nama> input]);
/* Spesifikasi Fungsi */
int main () {
    /* KAMUS */
    /* Semua nama yang dipakai dalam algoritma */
    /* ALGORITMA */
    /* Deretan instruksi pemberian harga, input, output, analisis kasus, pengulangan yang
    memakai fungsi */
    /* Harga yang dihasilkan fungsi juga dapat dipakai dalam ekspresi */
    nama = NMAF ([list parameter aktual]);
    printf ("[format]", NMAF ([list parameter aktual]));
    /* Harga yang dihasilkan fungsi juga dapat dipakai dalam ekspresi */

    return 0;
}
/* Body/Realisasi Fungsi diletakkan setelah program utama */
```

Prosedur (Notasi algoritmik)

```
procedure NAMAP (input nama1: type1,  
                  input/output nama2: type2,  
                  output nama3: type3)  
{ Spesifikasi, Initial State, Final State }
```

KAMUS LOKAL

```
{ Semua nama yang dipakai dalam BADAN PROSEDUR }
```

ALGORITMA

```
{ BADAN PROSEDUR }  
{ Deretan instruksi pemberian harga, input, output,  
  analisis kasus, pengulangan atau prosedur }
```

Prosedur (Bahasa C)

```
void NAMAP (type1 nama1, type2 *nama2, type3 *nama3)
/* Spesifikasi, Initial State, Final State */
{
    /* KAMUS LOKAL */
    /* Semua nama yang dipakai dalam BADAN PROSEDUR */

    /* ALGORITMA */
    /* Deretan instruksi pemberian harga, input, output,
       analisis kasus, pengulangan atau prosedur */

}
```

Pemanggilan Prosedur (Notasi alg.)

Program POKOKPERSOALAN

{ Spesifikasi: Input, Proses, Output }

KAMUS

{ semua nama yang dipakai dalam algoritma }

{ Prototype prosedur }

procedure NAMAP (input nama1: type1,
 input/output nama2: type2,
 output nama3: type3)
{ Spesifikasi prosedur, initial state, final state }

ALGORITMA

{ Deretan instruksi pemberian harga, input, output, analisis kasus, pengulangan }

NAMAP(paramaktual1, paramaktual2, paramaktual3)

{ Body/Realisasi Prosedur diletakkan setelah program utama }

Pemanggilan Prosedur (Bahasa C)

```
/* Program POKOKPERSOALAN */
/* Spesifikasi: Input, Proses, Output */

/* Prototype prosedur */
void NAMAP (type1 nama1, type2 *nama2, type3 *nama3);
/* Spesifikasi prosedur, initial state, final state */

int main () {
    /* KAMUS */
    /* semua nama yang dipakai dalam algoritma */

    /* ALGORITMA */
    /* Deretan instruksi pemberian harga, input, output, analisis kasus, pengulangan */
    NAMAP(paramaktual1, &paramaktual2, &paramaktual3);

    return 0;
}

/* Body/Realisasi Prosedur diletakkan setelah program utama */
```