Pengantar Pemrograman

Bahasa C

Baharuddin Aziz

Semester I 2019/2020

Tujuan Perkuliahan

- Mengenal bahasa C
- Memahami dasar-dasar bahasa C

Sejarah Singkat dan Versi (1/2)

- Bahasa C dikembangkan pada awal tahun 1970 oleh
  - Dennis M. Ritchie
  - Brian W. Kernighan
- Bahasa C berkembang di lingkungan UNIX
  - ±90% sistem operasi UNIX ditulis dalam bahasa C
- Standar yang ada:
  - Definisi Kernighan & Ritchie (K&R)
  - ANSI-C
  - Definisi AT&T (untuk superset C, C++)

Sejarah Singkat dan Versi (2/2)

- Versi pada PC, misalnya:
  - Lattice C
  - Microsoft C/Microsoft QuickC
  - Turbo C/Borland C++
- Pada tahun 1986, dikembangkan superset C oleh Bjarne Stroustrup
  - dilengkapi dengan kemampuan pemrograman berorientasi objek
  - disebut bahasa **C++ (C with Class)**

Aplikasi dalam Bahasa C (1/2)

Bahasa C banyak dipakai untuk:

1. membuat sistem operasi dan program-program sistem
2. pemrograman yang "dekat" ke perangkat keras
    - misalnya untuk kontrol peralatan atau sistem tertanam (*embedded system*)
3. membuat *tool kit*
4. menulis program aplikasi
    - GUI (*graphical user interface*), misal: Adobe Photoshop, Premiere, Illustrator
    - Google, yaitu file sistemnya dan browser Google chromium
    - Mozilla Firefox
    - MySQL
    - *gaming* dan animasi

Aplikasi dalam Bahasa C (2/2)

Kelebihan bahasa C, sehingga terpilih untuk aplikasi-aplikasi tersebut:

- kemampuannya untuk membuat kode yang ***compact***, **efisien**
- tidak mengorbankan *readability*
  - beda dengan bahasa *assembly* yang efisien namun susah dibaca, atau
  - bahasa tingkat tinggi lain yang enak dibaca namun tidak efisien

Walaupun tak dapat diingkari bahwa program dalam bahasa C lebih sulit dibaca (karena *compact*)

- dibandingkan dengan bahasa tingkat tinggi yang lain

Istilah dalam Bahasa C (1/2)

- **Blok**
  - sekumpulan kalimat C yang ditulis di antara **{** dan **}**
- **Definisi**
  - memberitahukan sifat (*property*) objek dan sekaligus mengalokasikan memori untuk objek
- **Deklarasi**
  - memberitahukan sifat (*property*) objek (terutama berkaitan dengan tipe)
- **Inisialisasi**
  - memberikan nilai objek
- **Deklarasi Global**
  - deklarasi yang berlaku dalam satu unit translasi (file)
- **Deklarasi Lokal**
  - deklarasi yang hanya berlaku dalam blok tempat deklarasi

Istilah dalam Bahasa C (2/2)

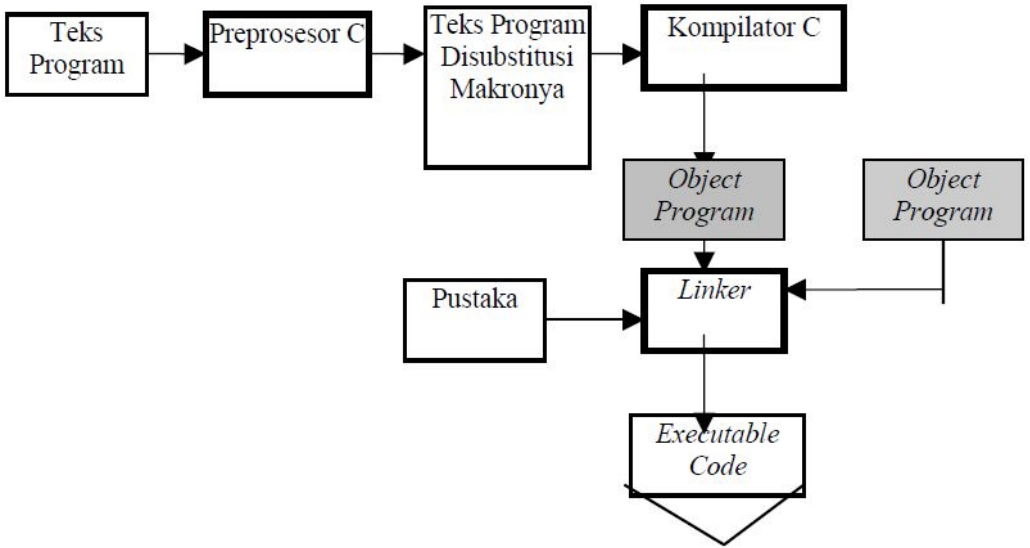
- **Objek**
  - daerah memori bernama (sama dengan variabel)
- **Lvalue**
  - ekspresi yang mereferensi suatu objek
  - *lvalue*: nilai di sebelah kiri ekspresi *assignment*
- **Prototipe deklarasi fungsi**
  - menyatakan nama, tipe *return value*, nama dan tipe parameter formal (*argument*)
- **Body**
  - realisasi dari fungsi
- **Scope**
  - daerah program tempat suatu nama dikenal

Struktur Program dalam Bahasa C

```
/* Nama File : …… */
/* identitas perancang/penulis */
/* Deskripsi ringkas dari program */

<tipe> main([int argc, char** argv[, char** envp]]) {

/* Keterangan program */

/* KAMUS */

/* Algoritma atau deretan instruksi yang executable */

return(<retval>);

}
```

Pemrosesan Kode Sumber dalam Bahasa C

Teks kode sumber diproses sebagai berikut.



Karakter yang Dipakai

- **Alphabet, besar dan kecil**
  - A-Z dan a-z
- **Angka**
  - 0 - 9
- **Karakter khusus**
  - + - * / = < > ( ) [ ] . , ; : { } ? # ! ~ & | % \
- **Karakter "blank"**
  - spasi, tabulasi horizontal/vertikal, CR, LF, FF

Jenis *Statement* dalam Bahasa C (1/6)

- Kalimat dalam bahasa C
  - selalu **diakhiri** dengan tanda **titik koma (';')**
- Kalimat dapat digolongkan menjadi dua
  - kalimat yang **tidak dieksekusi** (komentar, *assignment*)
  - kalimat yang **dieksekusi** (instruksi)

Jenis *Statement* dalam Bahasa C (2/6)

Kalimat *non-executable*

- kalimat yang **bukan dieksekusi**
  - melainkan sekedar komentar, atau
- kalimat untuk **melakukan deklarasi nama**
  - mungkin sekaligus melakukan inisialisasi nilai

Jenis *Statement* dalam Bahasa C (3/6)

Komentar

- Dituliskan **di antara** tanda **/*** dan ***/**
  - Disarankan agar setiap komentar dituliskan dalam satu baris walaupun dalam bahasa C dimungkinkan untuk membuat komentar yang terdiri dari lebih dari satu baris
- Pada beberapa kompilator
  - **di antara** tanda **//** dan **<eol> (*end of line*)**

Jenis *Statement* dalam Bahasa C (4/6)

Deklarasi

Mewakili "kamus" semua **nama** yang **didefinisikan** dan **akan dipakai**.

Nama yang harus dideklarasikan sebelum dipakai dalam lingkup yang sesuai:

- Deklarasi **nama konstanta dan nilainya**
- Deklarasi **struktur dan union**
- Deklarasi **nama type** yang didefinisikan
- Deklarasi **nama variabel dan type** yang sudah didefinisikan
  - baik oleh bahasa C atau didefinisikan sebelumnya
- Deklarasi **tipe turunan**
- Deklarasi **fungsi** (*prototype*)

Jenis *Statement* dalam Bahasa C (5/6)

Kalimat *executable*

Instruksi yang akan **dikerjakan** oleh **komputer**, meliputi **pemberian harga**, **kondisional**, **pengulangan** atau kalimat **percabangan**:

- *Assignment* (dengan operator **=**)
- Kondisional

```
if (<kondisi>) { };
if () { } else { };
switch
```

Jenis *Statement* dalam Bahasa C (6/6)

- Pengulangan
  `while`
  `do while`
  `for`
- Pencabangan
  `goto`
  `continue`
  `break`
  `return`

Nama dalam Bahasa C (1/8)

Nama (*identifier*)

dipakai untuk **mengenali** suatu **objek** dalam sebuah program

Macam-macam nama

- nama fungsi
- nama tipe data, struktur, union, enumerasi
- nama konstanta
- nama objek/variabel
- nama label

Nama dalam Bahasa C (2/8)

Struktur Blok dan Nama

- Sebuah "**Blok**" dalam bahasa C dituliskan **di antara** tanda kurung kurawal buka "**{**" dan kurung kurawal tutup "**}**".
  - Sebuah blok dapat mengandung **deklarasi data** (kamus) dan **instruksi**
  - Bahasa C tidak mengenal deklarasi blok bertingkat (*nested*) seperti Pascal atau Ada
- Deklarasi nama (fungsi, variabel, tipe, konstan) yang dilakukan di luar fungsi disebut **deklarasi eksternal**.
  - Deklarasi di dalam fungsi disebut deklarasi internal.
  - Variabel dengan **deklarasi internal** berlaku lokal terhadap blok tempat ia dideklarasi.
  - Nama variabel dengan **deklarasi eksternal** berlaku global dalam file tempat ia dideklarasi.

Nama dalam Bahasa C (3/8)

Mengacu suatu Nama

- Dengan menyebutkan (mengacu) suatu **nama**,
  - berarti kita **mengacu kepada nilainya**
- Nama yang diacu harus pernah dideklarasikan sebelumnya (tidak berlaku untuk nama fungsi eksternal).
  - Fungsi eksternal yang belum dideklarasikan dianggap mempunyai *return value* dan parameter bertipe **int** atau **double** (tergantung pada tipe parameter aktual).
  - Jika deklarasi implisit ini tidak sesuai akan timbul kesalahan pada saat kompilasi.
  - Sebaiknya, setiap fungsi eksternal yang dipakai dideklarasikan dengan prototipe.

Nama dalam Bahasa C (4/8)

Aturan nama

- terdiri dari **huruf**, **angka**, dan **garis bawah "_"** (*underscore*)
- huruf besar dan huruf kecil dibedakan
- dimulai dengan huruf

Nama dalam Bahasa C (5/8)

- **tidak boleh** *reserved word*, untuk C standar (ANSI C):

`auto`	`double`	`int`	`struct`
`break`	`else`	`long`	`switch`
`case`	`enum`	`register`	`typedef`
`char`	`extern`	`return`	`union`
`const`	`float`	`short`	`unsigned`
`continue`	`for`	`signed`	`void`
`default`	`goto`	`sizeof`	`volatile`
`do`	`if`	`static`	`while`

Nama dalam Bahasa C (6/8)

Aturan akses nama

- Berdasarkan deklarasinya, dibedakan:
  - nama **global** (deklarasi global)
  - nama **lokal** (deklarasi lokal)
- Nama **global** dapat diakses oleh semua fungsi dalam file yang sama
  - agar nama ini dapat diakses fungsi di file lain, nama ini harus dideklarasikan lagi di file tsb.
- Nama yang dideklarasi pada suatu fungsi
  - hanya dapat diakses **dalam fungsi tersebut**
- Jika ada **nama yang sama**
  - yang diacu adalah nama lokal

Nama dalam Bahasa C (7/8)

Name space dalam C

- *Name space* (ruang nama)
  - kategori nama yang dapat dipunyai oleh suatu nama yang dideklarasikan
- Sebuah nama yang sama dapat dipakai untuk keperluan yang berbeda, asalkan *name space*-nya berbeda
  - pada kuliah ini tidak dianjurkan memakai nama yang sama untuk keperluan yang berbeda

Nama dalam Bahasa C (8/8)

Ada lima *name space* dalam bahasa C

- nama **makro *preprocessor***
  - ini dipakai pada saat dilakukan preproses program/kode sumber;
  - setelah preproses selesai, nama ini tidak dikenal lagi;
- nama **label tujuan** perintah `goto`;
- nama **tag** struktur/union
  - nama yang mengikuti kata kunci *struct* atau *union*);
- nama **anggota** struktur/union;
  - masing-masing struktur/union mempunyai *name space* sendiri,
  - nama yang sama dapat muncul sebagai anggota struktur/ union yang berbeda;
- nama yang tidak termasuk salah satu di atas,
  - termasuk dalam *name space* untuk variabel, fungsi, tipe, dan enumerasi.

Pustaka Bahasa C yang Standar

Nama file	Isi
stdio.h	Berisi fungsi, type dan makro yang terdefinisi untuk operasi Input/Output, yaitu: operasi file format output format input fungsi input dan output untuk karakter fungsi untuk "direct input and output" Menentukan posisi pada file fungsi-fungsi sehubungan dengan error
ctype.h	Character Class type, yaitu fungsi-fungsi untuk melakukan test terhadap karakter
math.h	fungsi matematik
stdlib.h	berisi deklarasi fungsi untuk konversi, alokasi memori dan sejenisnya
string.h	Fungsi-fungsi untuk manipulasi string, yang terdiri dari dua kelompok: kelompok yang dimulai dengan str, untuk manipulasi string (misalnya copy, membandingkan, dsb) kelompok yang dimulai dengan mem, untuk manipulasi objek bertype caharacter array
assert.h	untuk diagnostik program
stdarg.h	fasilitas untuk memanfaatkan argumen fungsi
setjmp.h	untuk menghindari pemanggilan/pengembalian nilai fungsi yang normal. Berguna misalnya untuk keluar langsung dari nested function call yang rumit
signal.h	untuk menangani exception yang terjadi selama eksekusi, misalnya interrupt dari piranti eksternal atau error pada saat eksekusi
time.h	berisi type dan fungsi untuk manipulasi waktu (date, time)
limit.h	Tergantung implementasi kompilator. Berisi nilai konstanta untuk type integral (misalnya nilai integer minimum, maksimum dsb)

Terjemah dari Notasi Algoritmik ke C (1/6)

ALGORITMA	C
1. ASSIGNMENT : <nama> <- harga	 <nama> = harga;
2. KONDISIONAL : If <kondisi> then Aksi-A	 if (kondisi) Aksi-A;

Terjemah dari Notasi Algoritmik ke C (2/6)

```
if   <kondisi>
then
     Aksi-A
else
     Aksi-B
```

```
if  (kondisi)
          Aksi-A;
else
          Aksi-B;
```

Terjemah dari Notasi Algoritmik ke C (3/6)

depend on <nama> <kondisi-1> : Aksi-1 <kondisi-2> : Aksi-2 <kondisi-3> : Aksi-3 . . . . <kondisi-n> : Aksi-n	if (kondisi-1) Aksi-1; else if (kondisi-2) Aksi-2; else if (kondisi-3) Aksi-3; else if (kondisi-4) . . . else if (kondisi-n) Aksi-n;

Terjemah dari Notasi Algoritmik ke C (4/6)

ALGORITMA	C
3. PENGULANGAN :	
while <kondisi-ULANG> do Aksi	while (kondisi-ULANG) Aksi;
repeat Aksi until<kondisi-STOP>	do Aksi while !(kondisi-STOP);

Terjemah dari Notasi Algoritmik ke C (5/6)

iterasi
 	
Aksi-A
 	
stop <kondisi-STOP>
 	
Aksi-B
	for(;;) {
	
	Aksi-A;
	
	if (kondisi-STOP)
	exit;
	else
	AKSI-B;
	}

i traversal[Awal..Akhir]
 	
Aksi
	/* Jika Awal <= Akhir */
	for(i=Awal, i<=Akhir, i++)
	
	Aksi;
	
	/* Jika Awal >= Akhir */
	for(i=Awal, i>=Akhir, i--)
	
	Aksi;

Terjemah dari Notasi Algoritmik ke C (6/6)

4. INPUT/OUTPUT	
read <nama>	fscanf(stream, format, &nama) ; scanf(format,nama);
write <nama>	fprintf(stream, format, nama); printf (format, nama);

Kompilasi Program Bahasa C

- Proses kompilasi menggunakan GCC pada *Command Prompt* di Windows.
- Melakukan seluruh proses kompilasi
  - *preprocessing*,
  - *compiling*,
  - *assembly*, dan
  - *linking*

```
gcc -o Program.exe Program.c
```

Contoh Program: `hello.c`

```
/* File : hello.c */

#include<stdio.h>

void main() {
    printf("hello\n");
}
```

Contoh Program: `hello1.c`

```
/* File   : hello1.c */
/* Author : Baharuddin Aziz */
/* menuliskan hello ke layar */
/* pola ini merupakan standar yang dipakai di kelas */

#include<stdio.h>

int main () {
     /* KAMUS */
     int kosong;

     /* ALGORITMA */
     printf ("hello\n");
     scanf("%d",&kosong);

     return 0;
}
```

Pertanyaan?