Translasi Notasi Algoritmik ke Bahasa C

Tim Pengajar IF2030

C Programming Language
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
- Pada 1986 dikembangkan superset C yaitu C++ yang dilengkapi kemampuan pemrograman berorientasi objek

C Programming Language
Aplikasi

- 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, masih readable
- Kekurangan Bahasa C:
  - Masih kurang readable dibanding bahasa tingkat tinggi lain

C Programming Language
Beberapa Catatan

- Blok Program
  - Sekumpulan kalimat 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 diantara tanda /* dan */
  - Alternatif pada beberapa kompilator: // dan <eol>
- Bahasa C adalah bahasa yang *case-sensitive*

Konsep Pemrograman Prosedural

- Struktur umum program prosedural
- Type, konstanta, deklarasi, assignment
- Input/output
- Analisis kasus
- Pengulangan
- Subprogram (fungsi, prosedur)
- Type data koleksi: array

Struktur Umum Program

Struktur Umum Program (1/2)

- Notasi Algoritmik:

<u>Program</u> <nama-program> { Spesifikasi teks algoritmik secara umum }
KAMUS { Definisi konstanta, type, deklarasi variabel, spesifikasi prosedur, fungsi }
ALGORITMA { Teks algoritma - tidak berada di antara tanda kurung kurawal }

Struktur Umum Program (2/2)

- Bahasa C

```
/* File : <nama-file>.c */
/* Program <nama-program> */
/* Spesifikasi teks algoritmik secara umum  */
int main ()
{
    /* KAMUS */
    /* deklarasi nama variable */

    /* ALGORITMA */
    /* teks algoritma */


    return 0;
}
```

Type, Konstanta, Variable, Assignment

Konstanta, Variable

Notasi Algoritmik	Bahasa C
Konstanta constant <nama>:<type>=<harga>	**Konstanta** **1) Dengan const:** const [<type>] <nama> = <harga>; **2) Dengan C preprocessor:** #define <nama> <harga>
Deklarasi Variable <nama> : <type> **Inisialisasi/Assignment** <nama> ← <harga>	**Deklarasi Variable** <type> <nama>; **Deklarasi sekaligus inisialisasi** <type> <nama> = <harga>;

Notasi Algoritmik

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

KAMUS
  {Konstanta}
  constant lima : real = 5.0



  constant PI : real = 3.14

  {Deklarasi Variabel}
  L, r : real
  I    : integer

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

Bahasa C

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

/* KAMUS */
#define lima 5.0

int main () {
  /* KAMUS */
  /* Konstanta */
  const float PI = 3.14;

  /* Deklarasi variabel */
  float L, r;
  int I = 1; /* Inisialisasi */

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

Assignment

Notasi Algoritmik	Bahasa C

Assignment
<nama1> ← <nama2>
<nama> ← <konstanta>
<nama> ← <ekspresi>
nama1 ← nama1 <opr> nama2
 	
Contoh:
L ← PI * r * r
x ← x * y
i ← i + 1
 	
i ← i - 1
	Assignment
	<nama1> = <nama2>;
	<nama> = <konstanta>;
	<nama> = <ekspresi>;
	nama1 = nama1 <opr> nama2;
	Compound Assignment:
	nama1 <opr>= nama2;
	Contoh:
	L = PI * r * r;
	x *= y;
	i++;
	++i; /* Apa bedanya? */
	i--;
	--i; /* Apa bedanya? */

Type Data Character (Bahasa C)

- Contoh deklarasi: `char CC;`
- Contoh konstanta:

  `'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 Integer (Bahasa C)

- Contoh deklarasi: `int i; short int j;`
- Contoh konstanta: `1  2  0  -1`
- Jenis-jenis integer
  - [signed] **int**
    - Natural size of integers in host machine, eg. 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]
    - Positive integers only and 0

Type Data Real (Bahasa C)

- Contoh deklarasi: `float f1; double f2;`
- Contoh konstanta: `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 sbg. 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 kuliah**

```
/* 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 : cobabool.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 character
- Contoh konstanta string: `"Ini sebuah string"`
  - Konstanta string berada di antara double quotes " "
- String ≠ array of char
  - Representasi internal string selalu diakhiri character '\0', sedang 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 msg1[] = “ini string”; /* deklarasi dan inisialisasi */
char msg2[37]; /* deklarasi string sepanjang 37 karakter */
char *msg3;
```

- Contoh cara assignment nilai:

```
strcpy(msg2, “pesan apa”);  /* msg2 diisi dgn “pesan apa” */
msg3 = (char *) malloc (20 * sizeof(char)); /* alokasi
memori terlebih dahulu */
strcpy(msg3, “”);  /* msg3 diisi dengan string kosong */
/* HATI-HATI, cara di bawah ini SALAH! */
msg3 = “Kamu pesan apa”;
```

Type Enumerasi

Notasi Algoritmik	Bahasa C
KAMUS **{ Definisi type }** **type** hari : (senin, selasa, rabu, kamis, jumat, sabtu) **{ Deklarasi variable }** H : Hari	**/* KAMUS */** **/* Definisi type */** typedef enum { senin, selasa, rabu, kamis, jumat, sabtu } hari; /* senin=1, selasa=2, rabu=3, dst. */ **/* Deklarasi variable */** Hari H;
ALGORITMA **{ Assignment }** H ← senin	**/* ALGORITMA */** **/* Assignment */** H = senin; /* H = 1 */

Type Bentukan

Notasi Algoritmik	Bahasa C

KAMUS
{ Definisi Type }
type namatype :
  < elemen1 : type1,
  elemen2 : type2,
  ... >
 	
{ Deklarasi Variable }
nmvar1 : namatype
nmvar2 : type1 {misal}
	/* KAMUS */
	/* Definisi Type */
	typedef struct {
	  type1 elemen1;
	  type2 elemen2;
	  ...
	} namatype;
	/* Deklarasi Variabel */
	namatype nmvar1;
	type1 nmvar2; /*misal*/

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

Type Bentukan (Contoh)

Notasi Algoritmik	Bahasa C

KAMUS
{ Definisi Type }
type Point :
  < X : integer,
  Y : integer >
 	
{ Deklarasi Variable }
P : Point
Bil : integer {misal}
	/* KAMUS */
	/* Definisi Type */
	typedef struct {
	  int X;
	  int Y;
	} Point;
	/* Deklarasi Variabel */
	Point P;
	int Bil; /* misal */

ALGORITMA **{ Akses Elemen }** Bil ← P.X P.Y ← 20	**/* ALGORITMA */** **/* Akses Elemen */** Bil = P.X; P.Y = 20;

Operator

Notasi Algoritmik	Bahasa C
Ekspresi Infix: <opn1> <opr> <opn2> Contoh: X + 1	**Ekspresi Infix:** <opn1> <opr> <opn2> Contoh: X + 1
Operator Numerik: + - * / div mod	**Operator Numerik:** + - * / /*hasil float*/ / /*hasil integer*/ % ++ /*increment*/ -- /*decrement*/

Operator

Notasi Algoritmik	Bahasa C

Operator Relasional:
`>` `<`
`≥` `≤`
`=` `≠`
	Operator Relasional:
	`>` `<`
	`>=` `<=`
	`==` `!=`

Operator Logika:
AND
OR
NOT
	Operator Logika:
	`&&`
	`||`
	`!`

	Operator Bit:
	`<<  /*shift left*/`
	`>>  /*shift right*/`
	`&  /*and*/`
	`|  /*or*/`
	`^  /*xor*/`
	`~  /*not*/`
	`/*Lihat contoh di diktat*/`

Input/Output

Input

Notasi Algoritmik	Bahasa C
input(<list-nama>)	scanf("<format>", <list-nama>);
Contoh: input(X) {x integer} input(X, Y) input(F) {F real} input(s) {s string}	**Contoh:** scanf("%d",&X); /*x integer*/ scanf("%d %d", &X, &Y); scanf("%f",&F); /*F real*/ scanf("%s",s); /*s string*/
	Contoh format sederhana: %d untuk type integer %f untuk type real %c untuk type character %s untuk type string

Output

Notasi Algoritmik	Bahasa C
output(<list-nama>)	printf("<format>", <list-nama>);

Contoh:
output(X) {x integer}
output(X, Y)
output("Contoh output")
output("Namaku: ", nama)
output(F) {F real}
output(CC) {c character}
	Contoh:
	printf("%d",X); /*x integer*/
	printf("%d %d", X, Y);
	printf("Contoh output");
	printf("Namaku: %s", nama);
	printf("%f",F); /*F real*/
	printf("%c",CC); /*CC char*/

	Contoh format sederhana sama seperti pada input

Analisis Kasus

Analisis Kasus

Notasi Algoritmik	Bahasa C

Satu Kasus:
`if kondisi then`
`  aksi`
	Satu Kasus:
	`if (kondisi)`
	`  aksi;`

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

Analisis Kasus (> 2 kasus)

Notasi Algoritmik	Bahasa C

depend on nama
  kondisi-1 : aksi-1
  kondisi-2 : aksi-2
  ...
  kondisi-n : aksi-n
	if (kondisi-1)
	  aksi-1;
	else if (kondisi-2)
	  aksi-2;
	...
	else if (kondisi-n)
	  aksi-n;

depend on nama
  kondisi-1 : aksi-1
  kondisi-2 : aksi-2
  ...
  else : aksi-else
	if (kondisi-1)
	  aksi-1;
	else if (kondisi-2)
	  aksi-2;
	...
	else aksi-else;

Analisis Kasus (> 2 kasus)

Jika kondisi-1, kondisi-2… dapat dinyatakan dalam bentuk:
nama = const-exp (const-exp adalah suatu ekspresi konstan),
maka dapat digunakan switch.

Notasi Algoritmik	**Bahasa C**

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

Pengulangan berdasarkan kondisi berhenti:
repeat
  Aksi
until kondisi-stop
	
	do
	  Aksi;
	while (!kondisi-stop);

Pengulangan berdasarkan kondisi ulang:
while (kondisi-ulang) do
  Aksi
{not kondisi-ulang}
	
	while (kondisi-ulang)
	  Aksi;
	/*not kondisi-ulang */

Pengulangan

Notasi Algoritmik	Bahasa C

Pengulangan berdasarkan pencacah:
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;

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

Pengulangan

Notasi Algoritmik	Bahasa C

Pengulangan berdasarkan dua aksi:
iterate
  Aksi-1
stop kondisi-stop
  Aksi-2
	for(;;) {
	  Aksi-1;
	  if (kondisi-stop)
	  exit;
	  else
	  Aksi-2;
	}

Contoh

- Diktat “Contoh Program Kecil Bahasa C”
  - Pengulangan

Subprogram

Fungsi

Notasi Algoritmik:

```
function NAMAF (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 NAMAF (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 Algoritmik:

```
Program POKOKPERSOALAN
{ Spesifikasi: Input, Proses, Output }
```

```
KAMUS
{semua nama yang dipakai dalam algoritma }
   {Prototype fungsi}
   function NAMAF ([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 ← NAMAF ([list parameter aktual])
   output (NAMAF ([list parameter aktual]))

{Body/Realisasi Fungsi diletakkan setelah program utama}
```

Pemanggilan Fungsi

Bahasa C:

```
/* Program POKOKPERSOALAN */
/* Spesifikasi: Input, Proses, Output */
/* Prototype Fungsi */
type-hasil NAMAF ([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 = NAMAF ([list parameter aktual]);
    printf ("[format]", NAMAF ([list parameter aktual]));
    /* Harga yang dihasilkan fungsi juga dapat dipakai dalam
       ekspresi */

    return 0;
}
/* Body/Realisasi Fungsi diletakkan setalah 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 Algoritmik:

```
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
*/
```

Contoh

- Diktat “Contoh Program Kecil Bahasa C”
  - Prosedur & Fungsi

Type Data Koleksi : Array

Array Statik

Notasi Algoritmik	Bahasa C
{ Deklarasi Variabel } nm_array : array [0..nmax-1] of type-array	**/* Deklarasi Variabel */** type-array nm_array[nmax]; **Catatan: indeks array selalu dimulai dari 0**
{ Cara Mengacu Elemen } nm_array_{indeks}	**{ Cara Mengacu Elemen }** nm_array[indeks]
Contoh: TabInt : array [0..99] of integer $TabInt_i \leftarrow 1$ $X \leftarrow TabInt_{10}$	**Contoh:** int TabInt[100]; TabInt[i] = 1; X = TabInt[10];

Array Dinamik (Bahasa C)

- Array dinamik: alokasi memori untuk array (ukuran array) dapat diatur pada saat runtime
- Deklarasi: `type-array *nm_array;`
- Tahapan:
  1. Deklarasi (definisikan nama)
  2. Alokasi (tentukan ukuran/alokasi memori)
  3. Inisialisasi nilai
  4. Dealokasi (pengembalian memori)

Array Dinamik (Bahasa C) Contoh

1. Deklarasi (definisikan nama)

```
int *TabInt;
```

2. Alokasi (tentukan ukuran/alokasi memori)

```
TabInt = (int *) malloc (100 * sizeof(int));
/* TabInt dialokasi sebesar 100 */
```

3. Inisialisasi nilai

```
*(TabInt + i) = 9; /* pny. efek sama dgn: */
TabInt[i] = 9;
```

4. Dealokasi (pengembalian memori)

```
free(TabInt);
```

Array 2 Dimensi (Statik)

Notasi Algoritmik	Bahasa C
{Deklarasi Array} nm_array : array [0..nmax1-1] of array [0..nmax2-1] of type-array	**/*Deklarasi Array*/** type-array nm_array [nmax1][nmax2];
{Cara mengacu elemen} $nm\_array_{idx1,idx2}$	**/*Cara mengacu elemen*/** nm_array[idx1][idx2]
Contoh: Tab2D : array [0..2] of array [0..3] of integer $Tab2D_{i,j} \leftarrow 9$ $X \leftarrow TabInt_{2,3}$	**Contoh:** int Tab2D[3][4]; Tab2D[i][j] = 9; X = Tab2D[2][3];

Contoh

- Diktat “Contoh Program Kecil Bahasa C”
  - Array

Tambahan
Bahasan dalam Bahasa C

File Inclusion (1)

- Notasi umum:

  ```
  #include “nama-file”
  #include <nama-file>
  ```

  – Baris ini akan disubtitusi dengan isi dari file yang bernama nama-file (perhatikan letak direktori)
  – `#include “nama-file”` : pencarian dilakukan terhadap file berupa source code yang didefinisikan oleh pemrogram, `#include <nama-file>` : pencarian dilakukan terhadap file dalam standard library
  – Umumnya dituliskan di awal program di mana file inclusion dilakukan
- File yang di-include hanya file .h
- File .h tidak boleh mengandung variabel

File Inclusion (2)

- Beberapa contoh library

stdio.h	Input/output
ctype.h	Tes character
string.h	Fungsi-fungsi terkait string
math.h	Fungsi-fungsi matematika
stdlib.h	Fungsi-fungsi utility
time.h	Fungsi-fungsi date dan time

Scope & Lifetime Variabel (1)

- Mendeklarasikan variabel pada dasarnya adalah memesan sebuah lokasi di memori, interpretasinya tergantung pada 2 atribut utama: **type** dan **kelas penyimpanan**
- Kelas Penyimpanan
  - Menentukan *lifetime* dari suatu variable, yaitu berlakunya harga variabel tsb.
  - Ada 2 jenis:
    - **Otomatis**: objek aktif dalam masa hidup suatu blok program, lingkupnya lokal terhadap suatu blok, dan “dibuang” ketika eksekusi blok selesai → default
    - **Statik**: masa hidup “seumur hidup program utama”, lingkupnya lokal terhadap blok

Scope & Lifetime Variabel (2)

- Lingkup (*scope*) suatu variabel
  - Daerah program tempat variabel dapat dikenal
  - Lingkup dapat berupa suatu **blok program**, atau suatu **unit translasi (file)**.
- Kaitan (*linkage*)
  - Menentukan apakah suatu nama yang sama di lingkup lain menunjuk ke objek yang sama
  - Terdiri atas:
    - Objek dengan kaitan internal, hanya dikenal lokal dalam suatu file
    - Objek dengan kaitan eksternal, global terhadap seluruh program (termasuk file lain)

Scope & Lifetime Variabel (3)

- Saat deklarasi nama variabel dapat disertai kata kunci: auto, static, extern, register
- Jika S: scope/lingkup, K:kelas, dan L:kaitan/linkage:

Deklarasi	**auto**	**register**	**static**	**extern**	**Tanpa kata kunci**
Lokal	S:Blok K: otomatis L:-	S:Blok K: otomatis L:-	S:Blok K:statik L:-	S:Blok K:statik L:-	S:Blok K: otomatis L:-
Global	Tidak boleh auto	Tidak boleh register	S:File K:statik L: internal	S:File K:statik L: eksternal	S:File K:statik L: eksternal

Scope & Lifetime Variabel (4)

- Diktat “Contoh Program Kecil Bahasa C”
  - Scope & Lifetime

Scope & Lifetime Variabel (5)


```
/* deklarasi Global/Eksternal */
static i;   /* i lokal terhadap file */
extern j;   /* j harus didefinisikan di file lain */
float r;    /* r dapat diakses dari file lain */

/* deklarasi lokal dalam suatu blok */
/* awal suatu blok*/ {
    /* Kamus */
    auto int i; /* sama dengan int i */
    register loat f;
    static double d;
    ...
    /* Algoritma */
    ...
}
```

Struktur Komposisi (1)

- Dalam C, merepresentasikan struktur komposisi (*record*) dapat dengan beberapa cara (masing-masing berbeda pada prinsipnya)
- Cara-1: **variabel** berstruktur komposisi

```
struct {
   char nama[20];
   int nim;
   int nilai;
} Mhs; /*Mhs adalah variabel berupa struct*/
```

Struktur Komposisi (2)

- Cara-2: **tag** berstruktur komposisi
  - Catatan: tidak terbentuk type baru

```
struct Mhs {
   char nama[20];
   int nim;
   int nilai;
}; /*Mhs adalah tag berupa struct*/
struct Mhs M1, M2; /*variabel berstruktur Mhs*/
```

- Cara-3: **type** berstruktur komposisi
  - Lihat kembali type bentukan

Struktur Komposisi (3)

- Cara-4: **Union**
  - Memungkinkan struktur komposisi mempunyai alternatif type elemen
  - Seluruh elemen union memakai lokasi memori yang sama

```
/* KAMUS */
union IsiSel {
   char *form;
   int nili;
   float nilf;
}; /*IsiSel dapat bertype string, int, float*/
/*variabel berstruktur union*/
union IsiSel sel1, sel2;
/* ALGORITMA */
strcpy(sel1.form,“isinya apa ya”);
printf(“Isi sel1: %s”,sel1);
```

Type Data Pointer (1)

- Type data Pointer berisi alamat mesin
  - Menunjuk kepada nama yang diacu sehingga informasi pada nama dapat diakses
  - Memungkinkan alokasi dinamik → memori baru **dialokasi** berdasarkan kontrol pemrogram, jika sudah tidak dibutuhkan, dapat **didealokasi** → harus hati-hati
  - Dalam bahasa C, nilai variabel bertype pointer dapat dimanipulasi sebagaimana halnya nilai numerik

Type Data Pointer (2)

- Pointer sederhana:

Format Deklarasi:

```
<type> *<nama>;
```

Contoh:

```
int *i;       /*pointer ke integer*/
float *f;     /*pointer ke real*/
char *cc;     /*pointer ke character*/
FILE *FT;     /*handle suatu file*/
int *(T)[10]; /*pointer ke array dg 10 elemen integer*/
int *T[10];   /*pointer ke array dg 10 elemen bertype pointer
                ke integer*/
void *p;      /*pointer ke objek yg tdk diketahui typenya*/
```

Type Data Pointer (3)

- Pointer sederhana:

```
/*Kamus*/
int *iptr;

/*Algoritma*/

/*Alokasi memori untuk 1 buah integer*/
iptr = (int *) malloc (sizeof(int));
/*Inisialisasi nilai yang ditunjuk iptr*/
*iptr = 10;
printf("Nilai yang ditunjuk iptr : %d",*iptr);
free(iptr); /*Dealokasi iptr*/

/*Alokasi kembali dengan ukuran 4 integer*/
/*Menjadi array of integer yang dialokasi dinamik*/
iptr = (int *) malloc (4 * sizeof(int));
/*Mengisi elemen ke-0 dari iptr*/
*(iptr+0) = 999;
printf("Nilai yang ditunjuk iptr : %d", *(iptr+0));
free(iptr); /*Dealokasi iptr*/
```