Procedural Programming in C Language

Pelatnas I TOKI di ITB

Oleh : Inggriani Liem

Oktober 2012

Tujuan

- Siswa memahami konsep pemrograman prosedural dan implementasinya dalam bahasa C, berdasarkan pengalaman pemrograman dalam bahasa Pascal yang sudah dikuasainya.
- Siswa memahami hal-hal penting yang harus dikuasai, sebelum melakukan problem solving dan lomba yang sesungguhnya

Program Prosedural

- Program dalam bahasa C termasuk dalam program prosedural : Algoritma + Struktur Data
- Pemrograman prosedural (imperatif) :
  - Dihasilkan berdasarkan dekomposisi “aksional”, menjadi Aksi yang akan dijalankan secara beruturan.
  - Aksi :
    - Jelas Initial state, Final state dan harus dalam waktu terbatas
    - Dapat didekomposisi menjadi Sub Aksi
  - Aksi diterjemahkan menjadi sederetan instruksi (aksi primitif) yang akan dijalankan oleh mesin

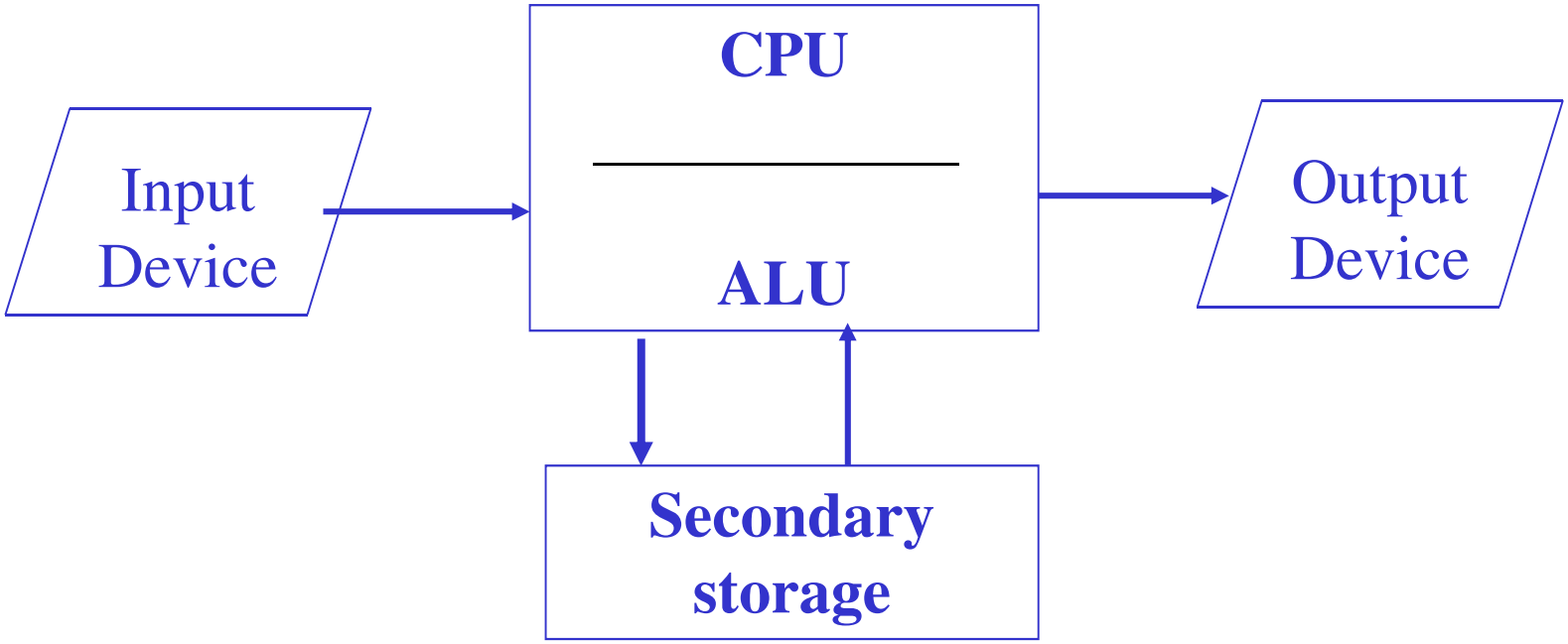
Program Prosedural

- Elemen sangat dasar:
  - Pengertian Type (nama, domain, literal,operator)
  - Ekspresi, variabel, konstanta
  - Input/output
- Elemen pembangun program
  - Sekuens [aksi sekuensial]
  - Analisa kasus, kondisional
  - Loop, pengulangan
- Modularisasi : prosedur dan fungsi

Bahasa C dan C++

- Bahasa C adalah bahasa Prosedural
- Bahasa C++ (C with Class) adalah bahasa campuran, yaitu bahasa prosedural yang dilengkapi fitur OO.
- Program dalam bahasa C dapat dieksekusi oleh C++, kecuali jika mengandung keyword C++

Model mesin Pengeksekusi Mesin Von Newmann

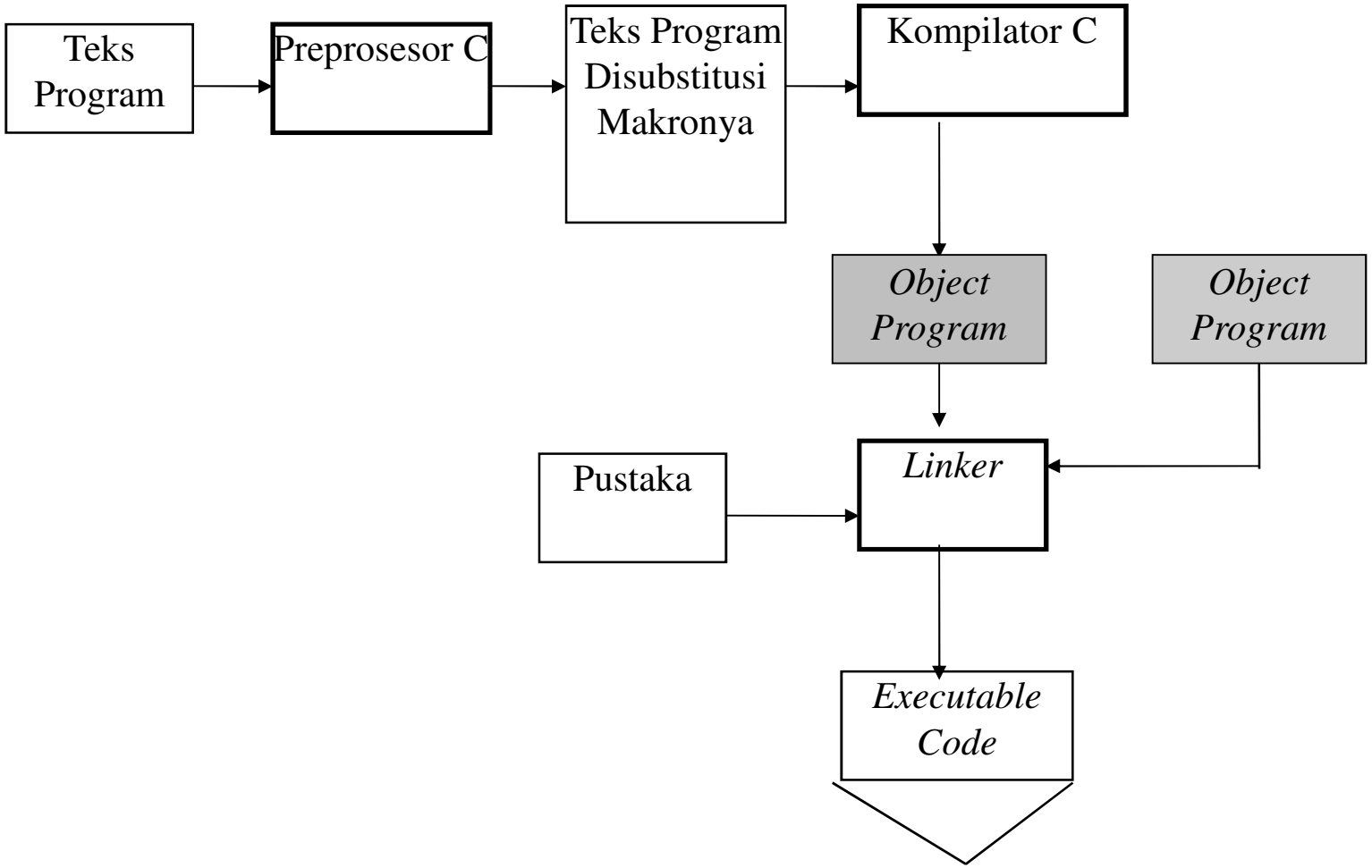


Model mesin ini menjadi dasar berpikir dalam membuat program. Oleh karena mesin sangat primitif, maka perlu ada abstraksi

Pengenalan Bahasa C

- Berasal dari Bahasa B, dibuat oleh Kernighan dan Ritchie
- Ciri : program yang ditulis salam Bahasa C biasanya singkat, ringkas dan padat
- Bahasa C hanya terdiri dari 32 keywords
- Sangat berbeda dengan Pascal: Case sensitive (Misalnya nama variabel `Bila` berbeda dengan `bila`)

Pemrosesan Program



Program Utama

```
/* File hello.c */
void main()
{
  printf("hello\n");
}
```

"Prosedur" namanya harus

main

```
(* File hello.pas *)
program Hello;
begin
  writeln('hello');
end.
```

Program Utama

```
/* File hello.c */
int main()
{
  printf("hello\n");
   return 0;
}
```

```
(* File hello.pas *)
program Hello;
begin
  writeln('hello');
end.
```

Contoh Deklarasi

```
(* Pascal *)
I: integer;
f: real;
CC: character;
s: string;
Found: boolean;
```

```
/* Bahasa C*/
int I;
float f;
char CC;
char* s;
/* tidak ada*/
/* false = 0
  */
```

Deklarasi

- 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 nama variabel dapat diikuti dengan inisialisasi nilainya atau tidak. **Semua variabel harus diinisialisasi (saat deklarasi atau dengan instruksi)**
- Deklarasi tipe turunan
- Deklarasi fungsi (prototype)

Lihat Contoh Program Kecil

Record

```
(* Pascal *)
TYPE Point= record
                x: integer;
                y: integer
             end;
VAR P:Point; (* akses: P.x; P.y *)
```

```
/* Bahasa C : bermacam cara*/
struct { int x;
         int y; } P; /*P:variabel*/
/*************************************/
typedef struct { int x;
                 int y; } Point; /*type*/
Point P; /* variabel: P.x; P.y */
```

Enumerasi – Bahasa C

Enumerasi dalam Bahasa C merupakan implementasi "set" dalam bahasa Pascal

```
enum hari  /* "type" */
 { senin, selasa, rabu, kamis, jumat,
  sabtu } hariku;
/* hariku : variabel yang langsung
  dideklarasi*/
enum
 {  satu, dua, tiga
    } angka;  /* variabel */
```

Enumerasi – bahasa C

```
/* "konstanta" bernama ", mengelompokkan */
enum
    {
      KEYWORD = 01, EXTERNAL = 03, STATIC = 04
    }

/* definisi enumerasi */
  typedef enum
    {
      merah, putih, kuning
    }
  warna; /* nama type */
  warna w = kuning;
```

Lihat Contoh Program Kecil

Contant
Tidak boleh berubah nilainya

```
CONST
 PI = 3.1415;
```

```
#define PI 3.1415
constant PI = 3.1415;
```

Catatan :

```
#define PI 3.1415
```

Lihat Contoh Program Kecil

Akan diproses oleh macro processor, di mana semua kemunculan “PI” dalam source code akan diganti menjadi “3.1415”

Apa yang terjadi dengan baris sbb

```
#define PI 3.1415  /* adalah nilai 22/7 */
```

Type Primitif

Pascal:

- integer
- real
- character
- boolean
- string

Bahasa C

- int
- float
- char
- (* 0=false; else true *)
- char *

Catatan : hati-hati dengan "string" dalam Bhs C. Akan dibahas secara khusus

Batasan Nilai

Variable Type	Keyword	Bytes Required	Range
Character	char	1	-128 to 127
Integer	int	2	-32768 to 32767
Short integer	short	2	-32768 to 32767
Long integer	long	4	-2,147,483,648 to 2,147,438,647
Unsigned character	unsigned char	1	0 to 255
Unsigned integer	unsigned int	2	0 to 65535
Unsigned short integer	unsigned short	2	0 to 65535
Unsigned long integer	unsigned long	4	0 to 4,294,967,295
Single-precision	float	4	1.2E-38 to
floating-point			3.4E38
Double-precision	double	8	2.2E-308 to
floating-point			1.8E308[2]

Operator Dasar

Pascal

- Numerik :
  `[+, -, *, /, div, mod]`
- Relational:
  `[<, >, =, <>, <=, >=]`
- Boolean:
  `[and, or, not]`

Bahasa C

- Numerik :
  `[+, -, *, /, /, %]`
- Relational:
  `[<, >, ==, !=, <=, >=]`
- Boolean:
  `[&&, ||, !]`
  `[&, |, !]`
  `/* bit */`

Ekspresi

- Terdiri dari Operan dan Operator
- Penulisan : prefix, infix, postfix
- Rumus, untuk menghitung, mengoperasikan suatu nilai sesuai dengan type
- Hasil ekspresi dapat disimpan, ditampilkan
- Ekspresi :
  - ketat terhadap type
  - "loose" terhadap type
- Usahakan menulis ekspresi ketat type

Lihat Contoh Program Kecil

Baca Tulis – `scanf/printf`

- Hanya type dasar. Untuk type lain: harus dibuat prosedur baca/tulis
- Membuat primitif Baca/Tulis untuk setiap type yang anda ciptakan, merupakan kewajiban.
- Membuat baca/tulis data dari textfile, harus dapat anda kode dalam waktu singkat untuk lomba IOI.

Lihat Contoh Program Kecil

Kalimat Executable

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

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

- **Pengulangan**

```
for, while() {. .}, do .. While
```

- **Pencabangan**

```
goto, continue, break, return
```

Assignment (=)

- Ruas kiri = Ruas Kanan;
- Ruas kiri harus variable
- Ruas kanan harus <ekspresi>
- Ekspresi :
  - operan (nama variabel, konstanta, aplikasi fungsi) dan operator harus kompatibel, ketat type.
  - ekspresi bukan hanya ekspresi aritmetika, ada ekspresi boolean, ekpresi relasional

Assignment, Blok, I/O

Pascal

Assignment :=

Blok : begin.. end;

Baca/Tulis:

```
readln (a,x);
write (a);
writeln (a);
```

Bahasa C

=

{ }

Baca/tulis :

```
scanf("%d,%d",&a,&b);
printf ("%d", a);
printf ("%d\n", a);
```

Lihat Contoh Program Kecil

Sekuensial [1]

- Dikerjakan secara terurut, mulai dari paling "awal" sampai paling akhir.
- Ada yang bisa dibolak-balik, ada yang tidak boleh.
- Jika program hanya mengandung sekuensial, selalu mengeksekusi secara sama, tidak terlalu banyak gunanya.

Sekuensial [2]

- Beberapa instruksi dasar atau konstruktor yang dieksekusi mesin secara sekuensial, berurutan, satu per satu per blok:
  - input
  - output
  - kondisional/analisa kasus
  - loop
- Dalam konteks program modular, program merupakan deretan aksi/sub aksi

Kondisional [1]

```
if (kondisi)
begin
(* aksi *)
end;

if (kondisi)
begin
(* aksi *)
end else
begin
end;
```

```
if (kondisi) {


  }

if (kondisi) {

} else {

}
```

Kondisional [2]

```
if (kondisi)
begin
(* aksi *)
end else if (kondisi)
begin . . .
end else if(kondisi)
begin
. . .
end else if (kondisi)
begin . . .
end else
begin . . . end;
```

```
if (kondisi) {

} else if (kondisi){
. . .
} else if (kondisi){
. . .
} else if (kondisi){
. . .
} else {
. . .
}
```

Kondisional [3]

```
case (var) of
val1 : begin
(* aksi *)
       end;
Val2 : begin
       end;
val3 : begin

       end;
else begin ... end;
```

```
switch (var) {
case val1:{  break;}
case val2:{  break;}
case val3:{  break;}
case val4:{  break;}
default: { }

}
```

Rancangan Kondisional

- Semantik harus benar (pilih instruksi “if” yang cocok). Akan diberikan contoh.
- Syarat Kondisi:
  - harus mencakup semua kasus yang mungkin terjadi, atau domain nilai. Kalau ada yang lupa, akan ‘bocor’
  - tidak boleh overlap (harus *mutual exclusive*).
- Pakailah “else” hanya untuk menangani kesalahan.

Lihat Contoh Program Kecil

Loop [1]

```
for var := Awal to Akhir do
begin . . . end;
for var := Awal downto Akhir do
begin . . . end;
```

```
for (var=awal;var<=akhir;var++)
{ . . . }
for (var=awal;var<=akhir;var--)
{ . . . }
```

Lihat Contoh Program Kecil

Catatan mengenai Loop `for`

- Loop `for` dalam C dapat dipakai secara lebih umum (bukan hanya traversal naik/turun dari Awal s/d akhir)

```
for (inisialisasi; kondisi ulang; next-elmt)
{ … }
```

```
for (inisialisasi; kondisi ulang; deretan-
   instruksi dipisahkan koma)
{ … }
```

Loop while

```
while  (kondisi ulang) do
begin . . . end;
```

```
while (kondisi ulang) {

}
```

Lihat Contoh Program Kecil

Loop Repeat

```
repeat

until (stop kond);
```

```
do {

} while (kond ulang);
```

Lihat Contoh Program Kecil

Memilih Loop

- `For` : Jika diketahui batasannya awal dan akhir, berapa kali body loop harus diulang.
- `REPEAT` : Jika kondisi berhenti oleh suatu ekspresi boolean, dan body loop minimal harus dieksekusi satu kali tanpa perlu ditest.
- `WHILE` : jika kondisi berhenti suatu ekspresi boolean, dan body loop mungkin tidak pernah dieksekusi, perlu ditest sebelum masuk ke body

Rancangan Loop

- Pakailah sesuai semantiknya.
- Mungkin perlu sub-aksi inisialisasi/terminasi
- Harus diyakinkan berhenti dari dibaca, bukan dari eksekusi
- Aksi dalam sebuah loop: sekuensial, kondisional, sebuah loop
- Loop dalam Loop : dapat mempengaruhi performansi dengan menentukan mana outer/inner loop

String

- String di bahasa Pascal: array of char (pointer ke karakter)
- Dalam bahasa C string adalah type khusus, dengan terminator `'\0'`
- Jangan menggunakan "assignment" untuk type string, pakailah fungsi `strcpy` yang diberikan
- Pelajari padanan fungsi string Pascal yang diberikan.

Alokasi/Dealokasi

```
(* Pascal *)
TYPE
  Elmt= record
          I:integer;
        next:^Elmt;
        end;


new(P);
dispose (P);
```

```
/* Bahasa C */
typedef struct
  telmt * adres;
typedef struct
  telmt {int I;
  adres next;}elmt;


P=(adres)malloc
  (sizeof (elmt));
free(P);
```

Array

```
T : array [1..10] of integer;
M : array [1..10][1..10] of integer;
```

```
int * T; /*belum alokasi */
int T[]; /* belum dialokasi */
int T[10]; /* indeks dari 0 s/d 9 */
int M[][]; /* belum dialokasi */
```

Lihat Contoh Program Kecil

Alokasi Array (Bahasa C)

```
T= (int*)malloc (10*sizeof(int));

M= (int*)malloc (10*sizeof(int));
M[i]:= NULL;
for (i=1,i<10;i++){
  M[i]= (int*)malloc(10*sizeof(int));
}
```

Lihat Contoh Program Kecil

Tahapan memakai Array

- Pertama :
  - Memakai array dalam bahasa C seperti memrogram dalam bahasa Pascal jika deklarasi indeks 1 s/d N, dengan menggeser indeks mulai dari 0 s/d N-1
  - Deklarasi T : array [1..10] of integer
  - Deklarasi T dalam bahasa C : int T[10]
- Kedua :
  - Memakai array secara dinamik (boleh berubah-ubah ukurannya)
- Ketiga
  - Memakai STL C++ vector

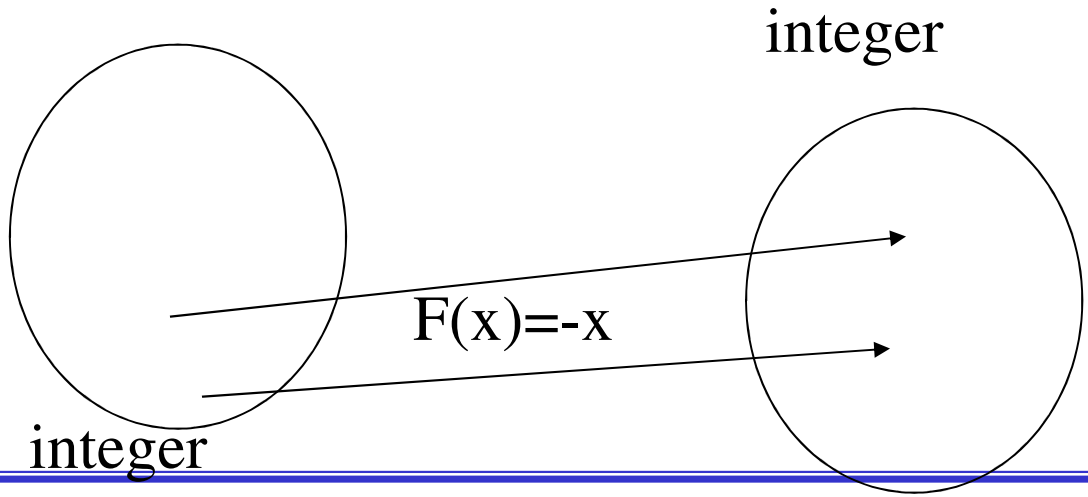
Deklarasi & Inisialisasi Variabel

```
int I=10;
float f=0.5;
int *T={1,2,3};
int T[5]= {0,1,2,3,4};
char C='a';
```

- Inisialisasi bukan instruksi.

Fungsi

- Pemetaan dari suatu domain ke range.
  Dalam konteks prosedural : parameter input (jika ada) menjadi type hasil.
- Fungsi mengirim nilai
- Nama fungsi, parameter input, type hasil



Prosedur

- Menyatakan perubahan state: dari Initial State menjadi Final State, dengan efek neto yang didefinisikan
- Aksi terdefinisi: nama, parameter in/out/in-out
- State biasanya dinyatakan oleh variable
- Nama prosedur : “aksional” (dibandingkan nama fungsi : “komputational).

Signature

- Nama subprogram dan parameternya
- Parameter formal: nama dan type
  - Untuk fungsi : input
  - Untuk prosedur : input, output, input output
- Spesifikasi fungsi/prosedur, interface
- Ada bahasa yang memungkinkan untuk menentukan mutability (const atau var) dan initial/default value terhadap parameter
- Ada bahasa yang memungkinkan “overloading”

Subprogram

- Dalam bahasa C hanya ada fungsi (tidak ada prosedur)
- Prosedur adalah fungsi yang tidak mengirim nalai, atau dengan return value `void`
- Disarankan untuk membuat prototype (lihat program kecil)
- Pemanggilan prosedur tanpa parameter: `geser;` versus **`geser();`**

Kerangka program dengan sub program

```
/* nama File : ….C*/

/* prototype prosedur/fungsi */

/* variabel global */

void main () {
}
/* body prosedur/fungsi */
```

Lihat Contoh Program Kecil

Templates Program

```
/* File : main.c             */
/* Deskripsi : program utama &semua nama lokal thd persoalan*/

#include "xxx.h"
/* include file lain yang perlu */

/* Bagian I : berisi kamus GLOBAL dan Prototypr              */
/* deklarasi semua nama dan prosedur/fungsi global*/

/* Bagian II : PROGRAM UTAMA                                 */
int main () {
/* Kamus lokal terhadap main */

/* Algoritma */


   return 0;

}

/* Bagian III : berisi realisasi kode program yang merupakan  */
/* BODY dari semua prototype yang didefinisikan pada file ini */
/* yaitu pada bagian I, dengan urut-urutan yang sama          */
/* Copy prototype, kemudian edit!! */
```

Lihat Contoh Program Kecil

Contoh

```
[<type qualifier>] [<kelas penyimpanan>][<tipe>]<NamaVar1>,<NamaVar2>, ...;
```

Contoh deklarasi sederhana:

```
/* deklarasi global/eksternal */
static i; /* i lokal terhadap file */
extern j; /* sama dengan extern int j, */
          /* j didefinisikan di file lain */
float r;  /* r dapat diakses dari file lain */

/* deklarasi lokal  */
/* awal sebuah blok */
     { /* Kamus */
       auto int i; /* sama dengan int i */
       register float f;
       static double d;
       static char* WalrusSaid[] = {"The", "time",
            "has", "come,", "to", "speak", "of", "many",
        "things"};
      /* Algoritma : … */

      }
```

Passing Parameter

- Passing parameter output atau parameter input/output selalu dengan reference (pointer)
- Perhatikan cara akses nilai. Lihat Contoh program kecil prosedur maksimum, fungsi scanf.
- Passing parameter array: perhatikan bahwa array sudah merupakan pointer.

Kerangka Prototype Fungsi Dalam Bahasa C

```
<type kembalian> NamaFungsi(param);
int Plus (int X; int Y);
int Max (int a; int b);

void NamaProsedur (parameter);
void Kali (int X; int Y; int*hsl);
void Swap (int *a; int *b);
```

Contoh Body Fungsi

```
int fact(int N) {
  if (N==0) /* basis */
   { return 1;}
 else /* recc */
   {
     return N*fact(N-1);
     }
}
/*call: printf("%d",fact(5));*/
```

Contoh Body Prosedur

```
void tukar(int* a, int* b) {
    /* menukar isi *a dan *b */
        int tmp=*a;
        *a=*b;
        *b=tmp;
    }
/* call: tukar (&a, &b) */
```

Array Sebagai Parameter [1]

- Perhatikan baik-baik jika array dipakai sebagai parameter prosedur/fungsi, karena deklarasi array of integer adalah

```
int* T; /* array of integer*/
```

- Prototype memprint array sebagai berikut tidak boleh mengubah isi array

```
void print (int* T);
/*print  array of integer*/
```

- Prototype increment isi setiap elemen array sebagai berikut harus mengubah isi array

```
void Inkremen (int* T, int N);
/*setiap elemen  array nilainya ditambah N*/
```

Array sebagai parameter [2]

- Sebagai latihan, tuliskan body dari prosedur dan fungsi serta contoh pemanggilan dalam main

```
int main (){
  int * MyTab; int N;
/* baca isi array MyTab[1] s/d MyTab[N] langsung di
  sini*/
/* print isi array MyTab [1..N] dg prosedur print */
/* incremen MyTab [1..N]  setiap nilai dengan 7 */
}
void print(int* T);
void Inkremen(int* T, int N);
```

Array sebagai parameter [3] bandingkan dg diktat bacatab1.c

```
typedef struct { int * T; int N } TabInt;
int main (){
  TabInt MyTab;
/* baca isi array MyTab.T[1] s/d
  MyTab.T[MyTab.N] langsung di sini*/
/* print isi array MyTab.T [1.. MyTab.N] dg
  prosedur print */
/* incremen MyTab.T [1.. MyTab.N]  setiap
  nilai dengan 7 */
}
void print(Tabint  T);
void Inkremen(Tabint* T, int N);
```

Variable Global dan Lokal

- Pada Bahasa Pascal, variabel yang dideklarasi pada program utama disebut global, sedangkan pada subprogram disebut lokal
- Pada Bahasa C, variabel yang dideklarasi pada program utama ("`main`") tetap merupakan lokal terhadap `main`
- Untuk mensimulasi variabel global, jika semua program dalam satu file, maka deklarasi ditaruh sebelum `main`
- Lihat contoh pada program kecil, untuk "Scope dan Lifetime".

Lihat Contoh Program Kecil

Deklarasi Static

- Dengan tambahan static, maka scope variabel menjadi lokal, namun nilainya selalu ada.
- Variabel static sekali diinisialisasi, akan berlanjut sampai akhir program
- Lihat contoh di program kecil, berjudul "Scope dan Life time"

Lihat Contoh Program Kecil

Pointer

- Bahasa C sangat konsisten terhadap aritmetika, termasuk aritmetika address (pointer)
- Pointer bisa diprint :)
- Jika p adalah pointer ke suatu nilai elemen array, anda dapat melakukan `p++` atau `p--` (indeks elemen berikutnya/sebelumnya)
- Berikut ini hanya dibahas pointer dalam bahasa C

Pointer to Type

Contoh pendefinisian nama bertype pointer ke ….

```
int *i; /* pointer ke integer */
float *x; /* pointer ke float */
char *cc; /* pointer ke karakter */
FILE *FileKu; /* Handle sebuah file */
int *PointerKeInteger;
char **argv; /* pointer ke pointer ke karakter */
int (*tabel)[13]; /* pointer ke array dengan 13
                          elemen integer */
int *tabel[13]; /* pointer ke array dengan 13 elemen
                       yang bertipe pointer ke integer */
void* p; /* pointer ke objek yang tidak diketahui
              tipenya */
```

Contoh pendefinisian, kemudian alokasi dinamik, dan penentuan nilai yang ditunjuk

```
int *iptr; /* deklarasi dalam kamus */
/* Algoritma */
iptr=(int*) malloc(sizeof(int)); /* alokasi tempat
                                    untuk satu integer */
*iptr=999;  /* nilai yg ditunjuk iptr diisi */
            /* dapat dilakukan setelah alokasi ! */
```

Pointer to function

Contoh deklarasi pointer ke fungsi:

```
int (*comp)(int, int);
 /* comp: pointer ke fungsi dengan dua argumen
          bertipe integer dan mempunyai return value
          integer */
char (*(*x[3])())
       /* x  array 3  elemen, masing-masing elemen
          adalah  pointer ke  fungsi  dengan  return
          value char* */

char (*(*y[3])())[5]
       /* y array 3 elemen, masing-masing elemen
           adalah  pointer  ke fungsi dengan return
          value pointer ke array karakter 5 elemen*/
```

Contoh pemanggilan:

```
if ((*comp)(a, b)) aksi();
printf("%s\n", x[0]);
```

Peringatan

- Pointer :
  - merupakan fitur dalam bahasa C yang kalau dimanfaatkan optimal akan menghasilkan program yang optimal, efisien, namun juga sulit dibaca dan “berbahaya” kalau tidak cermat.
  - Suatu implementasi yang bisa dilakukan tanpa pointer, bisa dilakukan dengan pointer.

Macro

```
#define pi 3.1415
#define max(a,b)((a)>(b) ? (a):(b))
#define Info(P) (P)->Info
```

- Macro akan disubstitusi oleh Macro processor

Hati-hati...[1]
Pesan Untuk Pascal Programmer

- Pascal tidak Case sensitive. C : case sensitive
- Pascal: selalu diakhiri `';'`
  C : setelah blok tidak harus diakhiri
  `';'(menjadi null statement)`
- Indeks array : dalam bahasa Pascal dapat ditentukan; dalam bahasa C selalu mulai dari NOL (lakukan offset)
- Bentuk `repeat...` yang diganti dengan `do..while` (kondisi adalah kondisi ulang)

Hati-hati [2]
Pesan Untuk Pascal Programmer

- String dalam C tidak boleh di-assign, pakailah `strcpy`. Lihat contoh
- Tidak disarankan memakai assignment berantai: `x=y=z;`
- Operator post/pre increment/decrement harus dipakai dengan hati-hati : `x++; ++x; y--; --y;`
- Type boolean dalam bahasa C: integer (0 adalah false; bukan 0 adalah true). Emulasi type boolean: lihat catatan singkat bahasa C.

Emulasi Boolean

Emulasi data boolean dapat dilakukan dengan beberapa cara: mendefinisikan nilai true dan false lewat `#define`,

```
#define true 1
#define false 0
#define boolean unsigned char
```

menggunakan enumerasi,

```
enum boolean {false, true}; /* perhatikan urutan,
      false harus pertama, karena false = 0 */
```

atau

```
enum boolean {true = 1, false = 0};
```

Hati-hati [3]
Pesan Untuk Pascal Programmer

- Deklarasi dalam blok instruksi tidak disarankan
- Komparasi: `and` dalam bahasa C (operasi relasional `&&` atau operasi bit `&`) bedanya besar (akan dijelaskan dengan contoh
- Kesalahan ketik sbb fatal (assignment "=") versus pemeriksaan apakah sama "==") :

```
if (I=0) { printf ("%d", I) }
```

jika yang dimaksud:

```
if (I==0) {printf ("%d", I) }
```

- Konversi type : **`integer (x)`** versus **`(int) x`**

Hati-hati
Pesan Untuk Pascal Programmer

- Type `char` langsung dapat dimanipulasi menjadi integer (kode ascii). Contoh:

```
{ char CC='a';
  printf ("%d", CC); }
```

- Bagi yang terbiasa memakai Pascal, berpindah secara '*smooth*", tidak apa-apa jika program mirip Pascal, tidak memakai fitur advanced bahasa C.

Referensi

- Dalam mengulas isi bahan ini, program kecil yang ada di diktat diacu. Oleh karena itu, “Diktat program kecil dalam bahasa C” dan “Ringkasan Bahasa C” merupakan dua diktat yang tidak terpisahkan dari bahan ini
- Padanan fungsi Turbo Pascal dengan C juga diberikan sebagai bagian tidak terpisahkan dari bahan ini

Latihan

- Pelajarilah kedua versi diktat sbb (bisa didownload di http://toki.if.itb.ac.id
  - Inggriani Liem : “Diktat program kecil dalam bahasa Pascal”
  - Inggriani Liem : “Diktat program kecil dalam bahasa C”
- Kenalilah dan bandingkanlah program-program kecil yang “sama”.

Latihan

- Terjemahkan dengan cepat, beberapa program bahasa pascal yang pernah anda buat, dengan mengacu ke padanan yang diberikan dalam slides ini dan spreadsheet padanan pascal bahasa C
- Terjemahkan beberapa instruksi dalam bahasa Algoritmik yang diberikan