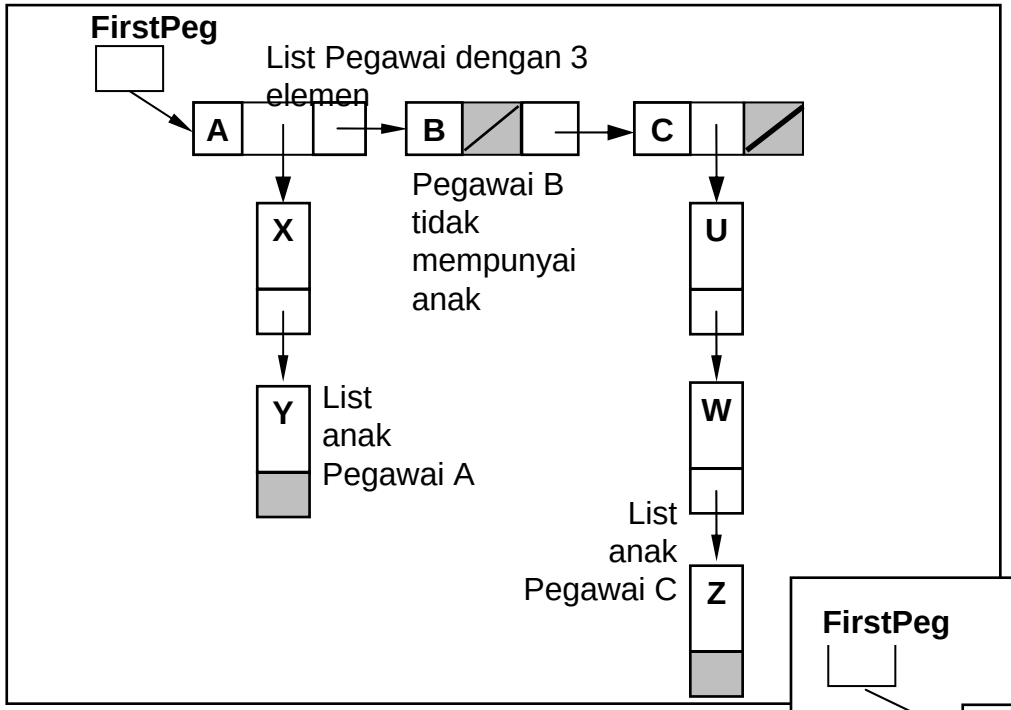
Studi Kasus: Multi-List

Tim Pengajar IF2030

Deskripsi Persoalan

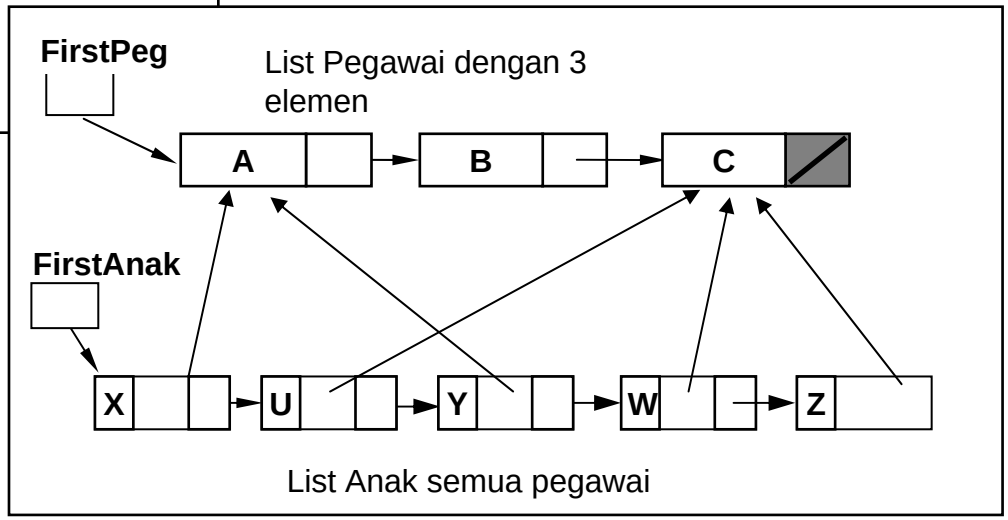
- Kita harus mengelola sekumpulan pegawai, dan untuk setiap pegawai selain informasi mengenai dirinya kita juga harus menyimpan informasi tentang anak-anaknya (jika ada).
- Jika informasi tersebut harus direpresentasikan dalam struktur data internal, maka kita mempunyai list dari pegawai, dan juga list dari anak-anak pegawai.
- Informasi pegawai: nopeg, nama, jabatan, gaji
  Informasi anak: nama, tanggal lahir

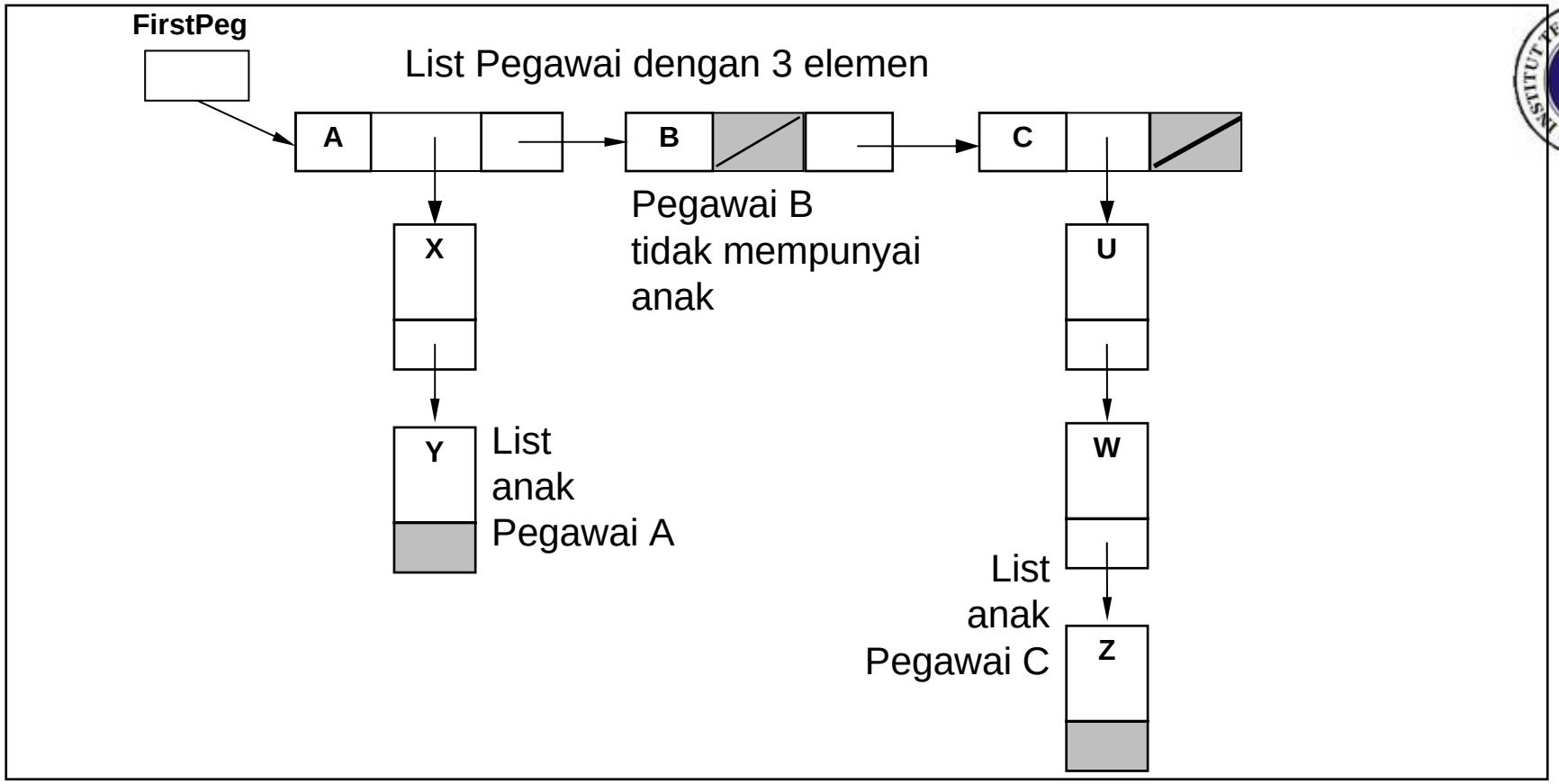
Alternatif Struktur Data



Mana yg lebih baik ?

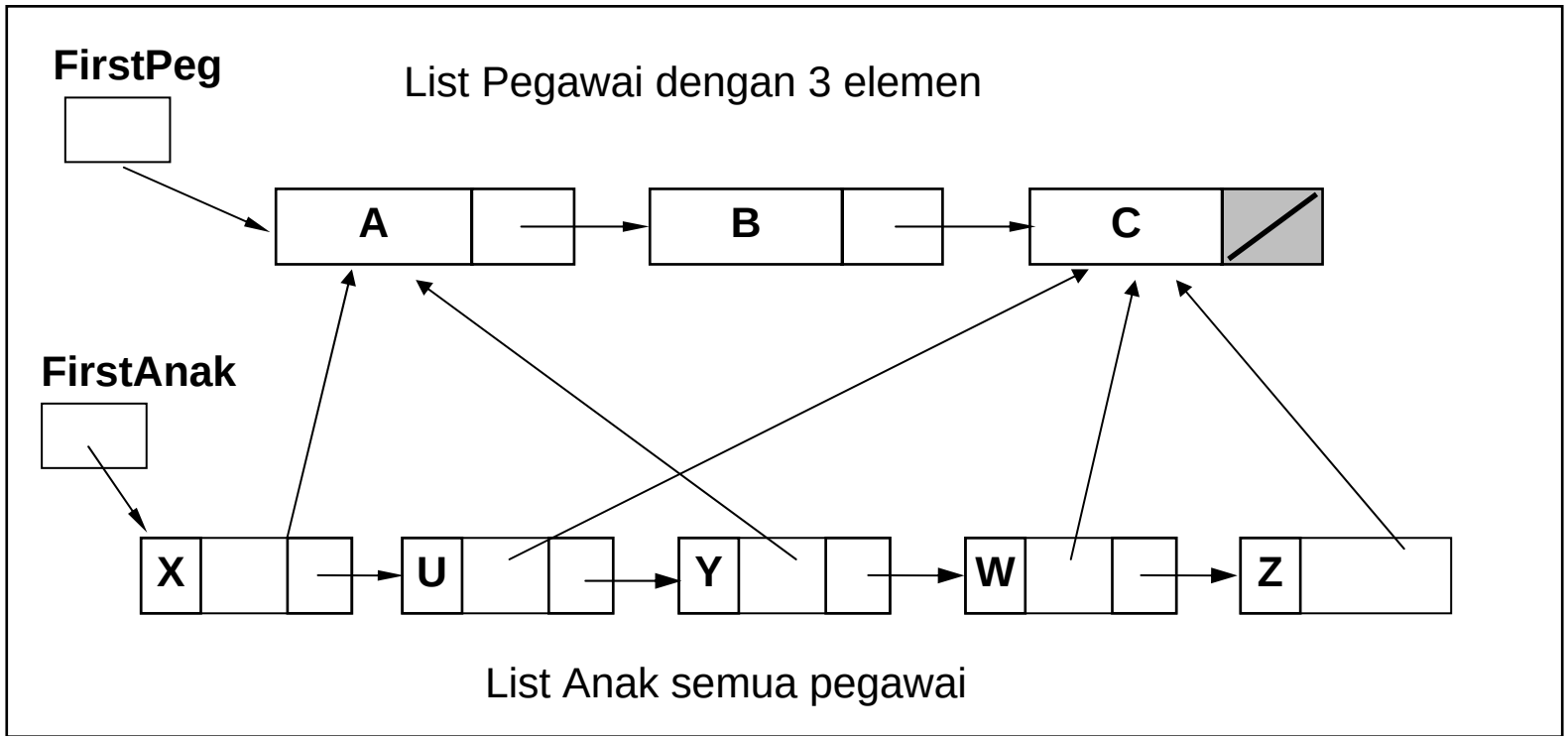
Pegawai: A, B, C, …
Anak:
A: X,Y
B: -
C: U,W,Z





KAMUS

```
type AdrPeg : { type terdefinisi, alamat sebuah elemen list pegawai }
type AdrAnak : { type terdefinisi, alamat sebuah elemen list anak }
type Pegawai : < NIP : integer, Nama : string, Jabatan : string,
                 GajiPokok:real, FirstAnak:AdrAnak, NextPeg:AdrPeg >
type Anak : < Nama : string, TglLahir : integer, NextAnak : AdrAnak >
type ListPeg : AdrPeg
FirstPeg : ListPeg
```



KAMUS

```
type AdrPeg : {type terdefinisi, alamat sebuah elemen list pegawai }
type AdrAnak : {type terdefinisi, alamat sebuah elemen list anak }
type Pegawai : < NIP : integer, Nama : string, Jabatan : string,
                 GajiPokok : real, NextPeg : AdrPeg >
type Anak : < Nama : string, TglLahir : integer, NextAnak : AdrAnak,
              Father : AdrPeg>
type ListPeg : AdrPeg
type ListAnak : AdrAnak
FirstPeg : ListPeg
FirstAnak : ListAnak
```

Fitur Program

1. Daftar pegawai, dan untuk setiap pegawai harus dibuat juga nama anak-anaknya (jika ada).
2. Daftar anak-anak yang umurnya kurang dari 18 tahun (untuk keperluan tunjangan).
3. Daftar pegawai yang anaknya lebih dari 3 (keperluan KB).
4. Diketahui nama seorang anak, harus dituliskan nama bapaknya.
5. Mendaftarkan seorang anak yang baru lahir ke dalam list anak, jika diberikan tanggal lahir dan nama anaknya, dan NIP Bapaknya.

Alternatif 1: hal 183-187

Alternatif 2: hal 188-191

Daftar Nama Pegawai & Anaknya

Alternatif-1:

```
Loop list pegawai
  output nama-pegawai
  Loop list anak dari pegawai
     output nama-anak
```

Alternatif-2:

```
Loop list pegawai
  output nama-pegawai
  Loop list anak
     if father(anak)=pegawai then
          output nama-anak
```

Kedua alternatif:

Perlu penanganan kasus kosong: "List kosong, tidak ada pegawai"
Perlu penanganan kasus kosong: "Pegawai ybs. tidak mempunyai anak“

Alternatif 1 (hal. 184)

```
procedure ListPegLengkap (input FirstPeg : ListPeg)
KAMUS LOKAL
PtrPeg : AdrPeg {address untuk traversal, @ sebuah elemen list pegawai }
PtrAnak : AdrAnak {address untuk traversal, @ sebuah elemen list anak }
Algoritma
{Traversal pegawai:skema pemrosesan sekuensial dg penanganan kasus kosong
Untuk setiap pegawai, traversal list anak untuk dituliskan namanya }
    PtrPeg ← FirstPeg { First Pegawai }
    if (PtrPeg = Nil) then
        output ("List kosong, tidak ada pegawai")
    else { Minimal 1 Pegawai }
        repeat
            output (Nama(PtrPeg))
            { Traversal Anak }
            PtrAnak ← FirstAnak(PtrPeg) { First Anak }
            if (PtrAnak = Nil) then
                output ("Pegawai ybs. tidak mempunyai anak")
            else
                repeat
                        output (Nama(PtrAnak)) { Proses anak }
                        PtrAnak ← NextAnak(PtrAnak) { Next Anak }
                until (PtrAnak = Nil)
            PtrPeg ← NextPeg(PtrPeg) { Next Pegawai }
        until (PtrPeg = Nil)
```

Alternatif 2 (hal. 189)

```
procedure ListPegLengkap ( input FirstPeg : ListPeg,
                           input FirstAnak : ListAnak )
KAMUS LOKAL
PtrPeg : AdrPeg {address untuk traversal, @ sebuah elemen list pegawai}
PtrAnak : AdrAnak {address untuk traversal, @ sebuah elemen list anak}
ALGORITMA
    {Traversal pegawai}
    PtrPeg ← FirstPeg { First Pegawai }
    if (PtrPeg = Nil) then
        output ("List kosong, tidak ada pegawai")
    else
        repeat
            output (Nama(PtrPeg))
            { Traversal Anak }
            PtrAnak ← FirstAnak { First Anak }
            while (PtrAnak ≠ Nil) do
                if (Father(PtrAnak) = PtrPeg) then { Proses }
                    output (Nama(PtrAnak))
                PtrAnak ← NextAnak(PtrAnak) { Next Anak }
            { PtrAnak = Nil }
            PtrPeg ← NextPeg(PtrPeg) { Next Pegawai }
        until (PtrPeg = Nil)
```

Daftar Anak < 18 tahun (Tunj.)

Alternatif-1:

```
Loop list pegawai
  output nama-pegawai
  Loop list anak dari pegawai
     if umur(anak)<18 then
          output nama-anak
```

Alternatif-2:

```
Loop list anak
  if umur(anak)<18 then
     output nama-anak, nama-father(anak)
Catatan: tidak perlu traversal list pegawai
```

Kedua alternatif:

Perlu penanganan kasus kosong

Alternatif 1 (hal. 185)

```
procedure ListTunjAnak (input FirstPeg : ListPeg)
{ Spesifikasi : mengacu pada diktat hlm. 185}
KAMUS LOKAL
    PtrPeg : AdrPeg; PtrAnak : AdrAnak
    UmurAnak : integer { umur anak pegawai }
    function Umur (TglLahir : integer) → integer
    { Fungsi yg mengirim umur... (lihat diktat) }
ALGORITMA
{ Trav.list Pegawai, skema sekuensial dg penanganan kasus kosong.
  Untuk setiap pegawai, traversal anaknya }
    PtrPeg ← FirstPeg { First Pegawai }
    if (PtrPeg = Nil) then
       output ("List kosong, tidak ada pegawai")
    else { Minimal ada satu pegawai }
        repeat
            output (Nama(PtrPeg))
            { Traversal Anak }
            PtrAnak ← FirstAnak(PtrPeg) { First anak }
            if (PtrAnak = Nil) then
                output ("Pegawai ybs tidak mempunyai anak")
            else { Minimal ada 1 anak }
                repeat
                  UmurAnak ← Umur(TglLahir(PtrAnak)) {Proses}
                  depend on UmurAnak
                    UmurAnak < 18 : output(Nama(PtrAnak),UmurAnak)
                    UmurAnak ≥ 18 : -
                  PtrAnak ← NextAnak(PtrAnak) { Next Anak }
                until (PtrAnak=Nil)
            PtrPeg ← NextPeg (PtrPeg) { Next Pegawai }
        until (PtrPeg=Nil)
```

Alternatif 2 (hal. 189)

```
procedure ListTunjAnak (input FirstAnak : ListAnak)
KAMUS LOKAL
    PtrAnak: AdrAnak {address utk traversal,@sbh elemen list anak}
    UmurAnak: integer {umur anak pegawai}
    function Umur (TglLahir : integer) → integer
    { Fungsi yg mengirim umur dgn rumus: tgl hari ini dr sistem
      dikurangi TglLahir }
ALGORITMA
{ Trav.list anak,skema proses sekuensial dg penanganan kasus kosong }
{ Untuk setiap anak periksa umurnya }
    PtrAnak ← FirstAnak { First Anak }
    if (PtrAnak = Nil) then
        output ("List Anak kosong, tidak ada anak")
    else
        repeat
            UmurAnak ← Umur(TglLahir(PtrAnak)) { Proses }
            depend on UmurAnak
                UmurAnak < 18 : output (Nama(Father(PtrAnak)))
                                output (Nama(PtrAnak), UmurAnak)
                UmurAnak ≥ 18 : -
            PtrAnak ← NextAnak(PtrAnak) { Next Anak }
        until (PtrAnak = Nil)
```

Daftar pegawai dg anak>3 (KB):

Alternatif-1:

```
Loop list pegawai
  Hitung anak pegawai dgn loop list anaknya
  if jumlah-anak>3 then
     output nama-pegawai, status anak>3
```

Alternatif-2 (pola ListPegLengkap):

```
Loop list pegawai
  Hitung anak pegawai dgn loop list anak
  if jumlah-anak>3 then
  output nama-pegawai, status-anak>3
```

Kedua alternatif:

Perlu penanganan kasus kosong

Alternatif 1 (hal. 186)

```
procedure ListPegNonKB (input FirstPeg : ListPeg)
{ I.S. List FirstPeg terdefinisi, mungkin kosong }
{ F.S. Semua pegawai yg anaknya > 3 orang ditulis informasinya }
KAMUS LOKAL
  PtrPeg: AdrPeg   {address utk traversal, @ elemen list pegawai }
  PtrAnak: AdrAnak {address utk traversal, @ elemen list anak }
  JumlahAnak: integer { banyaknya anak pegawai }
ALGORITMA
  { Traversal pegawai }
  PtrPeg ← FirstPeg                { First-Pegawai }
  if (PtrPeg = Nil) then
     output ("List kosong, tidak ada pegawai")
  else  { minimal ada satu pegawai }
     repeat
         { Traversal Anak }
         JumlahAnak ← 0            { Inisialisasi }
         PtrAnak ← FirstAnak(PtrPeg)    { First Anak}
         while (PtrAnak ≠ Nil) do
             JumlahAnak ← JumlahAnak + 1    { Proses }
             PtrAnak ← NextAnak(PtrAnak)    { Next Anak }
         { PtrAnak = Nil }
         if (JumlahAnak > 3 ) then
             output (Nama(PtrPeg), " mempunyai anak > 3")
         PtrPeg ← NextPeg(PtrPeg)         { Next Pegawai }
     until (PtrPeg = Nil)
```

Alternatif 2 (hal. 190)

```
procedure ListPegNonKB (input FirstPeg : ListPeg, input FirstAnak : ListAnak)
{ I.S. List First Peg terdefinisi, mungkin kosong }
{ F.S. Semua pegawai yang anaknya > 3 orang ditulis informasinya }
KAMUS LOKAL
  PtrPeg: AdrPeg   { address utk traversal, @ elemen list pegawai }
  PtrAnak: AdrAnak { address utk traversal, @ elemen list anak }
  JumlahAnak: integer { banyaknya anak pegawai }
ALGORITMA
{ Traversal pegawai }
  PtrPeg ← FirstPeg       { First Pegawai }
  if (PtrPeg = Nil) then
      output ("List pegawai kosong")
  else
      repeat  { Proses }
           JumlahAnak ← 0
           { Traversal Anak }
           PtrAnak ← FirstAnak    { First Anak }
           while (PtrAnak ≠ Nil) do
                if (Father(PtrAnak) = PtrPeg) then { Proses Anak }
                    JumlahAnak ← JumlahAnak + 1
                PtrAnak  ← NextAnak(PtrAnak)       { Next Anak }
           { PtrAnak = Nil) }
           if (JumlahAnak > 3) then
                output (Nama(PtrPeg), " mempunyai anak lebih dari 3")
           PtrPeg ← NextPeg(PtrPeg)                { Next Pegawai }
       until (PtrPeg = Nil)
  { semua elemen list pegawai selesai diproses }
```

Search Nama Bapak dari Anak

Alternatif-1:

```
Loop list pegawai
  Search NamaAnak dg loop list anak dr pegawai
     if NamaAnak ketemu then
        output nama-pegawai
        keluar dr loop anak dan loop pegawai
```

Alternatif-2:

```
Search NamaAnak dgn loop list anak
  if NamaAnak ketemu then
     output nama-pegawai
  keluar dr loop list anak
```

Alternatif 1 (hal. 186)

```
procedure OrTuAnak (input FirstPeg : ListPeg, input NamaAnak : string)
{ I.S. List Pegawai terdefinisi }
{ F.S. Jika ada anak yg bernama sesuai dg NamaAnak, nama Pegawai ditulis.
       Jika tidak ada NamaAnak, tidak menuliskan apa-apa }
KAMUS LOKAL
  PtrPeg: AdrPeg   { address utk traversal, @ elemen list pegawai }
  PtrAnak: AdrAnak { address utk traversal, @ elemen list anak }
  Found: boolean   { hasil pencarian orangtua anak }
ALGORITMA
  { Search }
  Found ← false
  PtrPeg ← FirstPeg
  while (PtrPeg ≠ Nil) and (not Found) do
     { Search anak dengan NamaAnak yang diberikan pada list anak }
     PtrAnak ← FirstAnak(PtrPeg)
  while (PtrAnak ≠ Nil) and (not Found) do
     if (Nama(PtrAnak) = NamaAnak) then
          Found ← true
     else
          PtrAnak ← NextAnak(PtrAnak)
  { PtrAnak = Nil or Found }
  if (not Found) then { eksplore pegawai yg berikutnya }
     PtrPeg ← NextPeg(PtrPeg)
  { PtrPeg = Nil or Found }
  if (Found) then
     output (Nama(PtrPeg))
```

Alternatif 2 (hal. 190)

```
procedure OrTuAnak (input FirstAnak : ListAnak, input NamaAnak : string)
{ Alternatif Kedua }
{ I.S. List Pegawai terdefinisi }
{ F.S. Jika ada anak yg bernama NamaAnak, nama Pegawai ditulis.
  Jika tidak ada NamaAnak, tidak menuliskan apa-apa.}
KAMUS LOKAL
PtrPeg : AdrPeg   { address utk traversal, @ elemen list pegawai }
PtrAnak : AdrAnak { address utk traversal, @ elemen list anak }
Found : boolean   { hasil pencarian orangtua anak }
ALGORITMA
{Search pada list Anak berdasarkan nama. Jika ketemu, akses Bapaknya }
PtrAnak ← FirstAnak; Found ← false
while (PtrAnak ≠ Nil) and (not Found) do
    if (Nama(PtrAnak) = NamaAnak) then
        Found ← true
    else
        PtrAnak ← NextAnak(PtrAnak)
{ PtrAnak = Nil or Found }
if (Found) then
    output (Nama(Father(PtrAnak)))
```

Mendaftarkan anak yang baru lahir

Alternatif-1:

```
Search elemen dgn NIPPeg pd list pegawai
If ketemu then
   Insert first data-anak pd list anaknya
```

Alternatif-2:

```
Search elemen dgn NIPPeg pd list pegawai
If ketemu then
   Insert first data-anak pada list anak
```

Alternatif 1 (hal. 187)

```
procedure AddAnak (input/output FirstPeg : ListPeg,
                   input NIPPeg : string, input NamaAnak : string,
                   input TglLahirAnak : integer)
{ Mendaftar seorang anak yang baru lahir, insert selalu pada awal list }
{ I.S. List Pegawai terdefinisi }
{ F.S. Jika pegawai dgn NIP=NIPPeg ada, alokasi anak.
Jika berhasil insert seorang anak sebagai elemen pertama list anak.
Jika alokasi gagal atau NIPPeg tidak ada, hanya menulis pesan }
KAMUS LOKAL
   PtrPeg : AdrPeg    { address utk traversal, @ elemen list pegawai }
   PtrAnak : AdrAnak  { address utk traversal, @ elemen list anak }
   FoundNIP : boolean { hasil pencarian NIP pegawai sebelum insert anak }
ALGORITMA
   { Search Pegawai }
   FoundNIP ← false
   PtrPeg ← FirstPeg
   while (PtrPeg ≠ Nil) and (not FoundNIP) do
       { search Pegawai dengan NIP yang diberikan }
       if (NIP(PtrPeg) = NIPPeg) then
           FoundNIP ← true
       else
           PtrPeg ←NextPeg(PtrPeg)

{ dilanjutkan ke slide 21 }
```

Alternatif 1 (hal. 187): lanjutan

```
{ lanjutan slide 20 }
{ Akhir search pegawai : PtrPeg=Nil or FoundNIP }
if (FoundNIP) then   { add anak }
   Alokasi(PtrAnak)
   if (PtrAnak ≠ Nil) then
       TglLahir(PtrAnak) ← TglLahirAnak
       Nama(PtrAnak) ← NamaAnak
       NextAnak(PtrAnak) ← Nil
       if (FirstAnak(PtrPeg)=Nil) then {Insert anak pertama }
           FirstAnak(PtrPeg) ← PtrAnak
       else  { Insert anak sebagai FirstAnak yang baru }
           NextAnak(PtrAnak) ← FirstAnak(PtrPeg)
           FirstAnak(PtrPeg) ← PtrAnak
   else { Alokasi gagal, tidak insert, hanya pesan }
        output ("Alokasi gagal")
else  { NIPPeg tidak ada, error }
   output ("Pegawai tidak ada dalam list")
```

Alternatif 2 (hal. 191)

```
procedure AddAnak (input/output FirstPeg: ListPeg,
                   input NIPPeg:integer,
                   input/output FirstAnak:ListAnak,
                   input NamaAnak:string, input TglLahirAnak:integer)
{ Mendaftar seorang anak yang baru lahir,
  Cari Pegawai, insert anak pada list anak selalu pada awal list }
{ I.S. List Pegawai terdefinisi }
{ F.S. List FirstPeg terdefinisi.
       Jika pegawai dengan NIP=NIPPeg ada, alokasi anak.
       Jika alokasi berhasil, insert seorg anak sbg elemen pertama
       list anak, tentukan Bapak.
       Jika alokasi gagal atau NIPPeg tidak ada, hanya menulis pesan.
}
KAMUS LOKAL
  PtrPeg : AdrPeg    {address utk traversal, @ elemen list pegawai }
  PtrAnak : AdrAnak  {address utk traversal, @ elemen list anak }
  FoundNIP : boolean {hasil pencarian NIP pegawai sblm insert anak }

{ dilanjutkan ke slide 23 }
```

```
{ lanjutan slide 22 }
ALGORITMA
{Search Pegawai dgn NIP yg diberikan: skema search dgn boolean}
FoundNIP ← false
PtrPeg ← FirstPeg
while (PtrPeg ≠ Nil) and (not FoundNIP) do
        if (NIP(PtrPeg) = NIPPeg) then
                FoundNIP ← true
        else
                PtrPeg ← NextPeg(PtrPeg)
{ PtrPeg = Nil or FoundNIP }
{ Akhir search pegawai : PtrPeg=Nil or FoundNIP }
if (FoundNIP) then      { Insert anak }
        Alokasi(PtrAnak)
        if (PtrAnak ≠ Nil) then
                TglLahir(PtrAnak) ← TglLahir
                Nama(PtrAnak) ← NamaAnak
                Father(PtrAnak) ← Ptrpeg { Tentukan Bapaknya }
                NextAnak(PtrAnak) ← Nil
                { Insert Anak }
                if (FirstAnak ≠ Nil) then
                     NextAnak(PtrAnak) ← FirstAnak
                     FirstAnak ← PtrAnak
       else {Alokasi gagal:tidak melakukan apa-apa, hanya pesan }
                output ("Alokasi gagal")
else  { NIPPeg tidak ada, error }
        output ("Pegawai tidak ada dalam list")
```

PR

- Modul Pra-Praktikum Bagian Pemrograman Prosedural:
  - P-19. STUDI KASUS 4 : MULTILIST
    - Bagian 1. Alternatif I
    - Bagian 2. Alternatif II