IF1210 Algoritma dan Pemrograman 1

Studi Kasus

Tim Pengajar IF1210

Sekolah Teknik Elektro dan Informatika

Studi Kasus: Polinom

Sumber: Diktat Struktur Data, hlm. 129-152

(Diktat tersedia di Edunex)

Deskripsi Persoalan

- Sebuah polinom berderajat n didefinisikan sebagai fungsi $P(x)$ berikut:

  $P(x) = a_n x^n + a_{n-1} x^{n-1} + a_{n-2} x^{n-2} + \ldots + a_2 x^2 + a_1 x^1 + a_0$

- Perhatikan bahwa untuk mempermudah pemrosesan, definisi $P(x)$ tersebut berbeda dengan definisi polinom pada matematika, yang biasanya diberikan sebagai berikut:

  $P(x) = a_0 x^n + a_1 x^{n-1} + a_2 x^{n-2} + \ldots + a_{n-2} x^2 + a_{n-1} x^1 + a_n$

- Contoh Polinom:

  $P1(x) = 4x^5 + 2x^4 + 7x^2 + 10$

  $P2(x) = 23x^{100} + 9x^9 + 2x^7 + 4x^5 + 9x^9 + 2x^4 + 3x^2 + 1$

  $P3(x) = 10$

  $P4(x) = 3x^2 + 2x + 8$

  $P5(x) = x^{1000}$

Proses Dasar Polinom (1)

1. Membentuk sebuah polinom *P* dari pasangan harga yang dibaca dari masukan, data yang dibaca adalah pasangan harga:
   (*) ⟨ degree: integer, coefficient: integer ⟩
   (1) ⟨ −999, 0 ⟩
2. Menuliskan sebuah polinom *P* terurut mulai dari suku terbesar sampai terkecil
3. Menjumlahkan dua buah polinom *P1* dan *P2* dan menyimpan hasilnya pada *P3*,
   P3 ≠ *P1* dan *P3* ≠ *P2*; pada akhir proses *P3* = *P1* + *P2*.

Proses Dasar Polinom (2)

4. Mengurangi dua buah polinom *P1* dan *P2* dan menyimpan hasilnya pada *P3*,
   P3 ≠ *P1* dan *P3* ≠ *P2*, pada akhir proses *P3* = *P1* – *P2*.
5. Membuat turunan dari sebuah polinom *P* dan menyimpan hasilnya pada *P'*,
   P' ≠ *P1*, pada akhir proses *P'* adalah turunan *P1*.

Pemilihan Struktur Data

- Secara lojik sebuah suku polinom direpresentasi oleh sepasang harga integer:

  `< degree: integer, coefficient: integer >`

- *P*(*x*) adalah kumpulan pasangan harga tersebut yang diidentifikasi oleh namanya dan merupakan sebuah list linier dengan elemen bertipe **Suku**, yaitu pasangan harga `< degree: integer, coefficient: integer >`, dengan harga **degree** yang maksimum sebagai derajat polinom.
- *P*(*x*) dapat direpresentasi secara **KONTIGU** atau **BERKAIT**.

Polinom: Representasi Kontigu

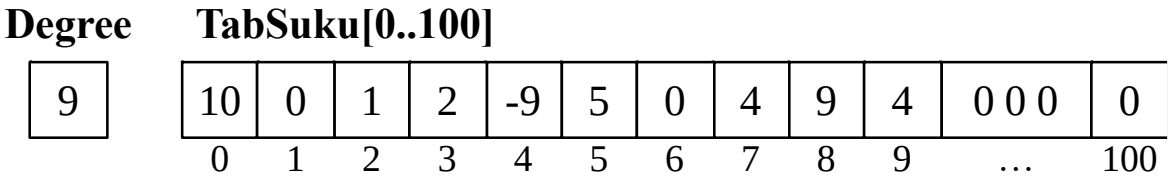
Representasi Kontigu

- Polinom direpresentasi dalam sebuah tabel P
  - setiap elemennya berisi koefisien dari polinom
  - **Derajat polinom secara implisit adalah indeks dari tabel**

```
KAMUS
{ Definisi sebuah polinom p adalah }
constant nMax: integer = 100
type  Polinom: < degree: integer≥0,
                 arrSuku: array [0..nMax] of integer >
p1, p2, p3: Polinom
{ p.arrSuku[i] adalah koefisien dari suku ke-i dari sebuah polinom }
```

Membentuk sebuah polinom

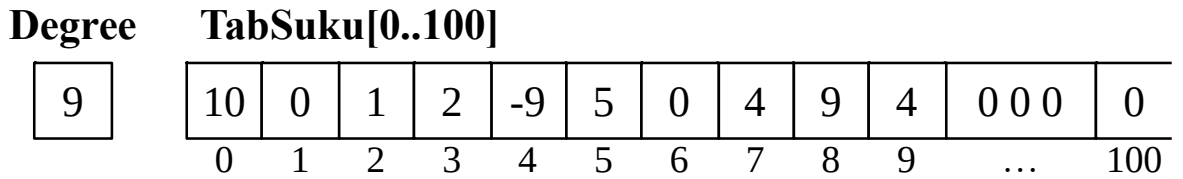
- Membentuk sebuah polinom dari pasangan harga yang dibaca dari alat masukan:
  - Setiap kali memasukkan data, yang dimasukkan adalah pasangan
    (*) ⟨ degree: integer, coefficient: integer ⟩
  - Akhir pemasukan adalah pasangan harga yang diketikkan bernilai ⟨ −999, 0 ⟩
- Prosesnya adalah **proses sekuensial dengan mark** untuk membaca dari masukan dan menyimpannya dalam tabel polinom.
- Contoh: Jika dibaca P1: ⟨ 0,10 ⟩, ⟨ 4,−9 ⟩, ⟨ 5,5 ⟩, ⟨ 8,9 ⟩, ⟨ 2,1 ⟩, ⟨ 3,2 ⟩, ⟨ 9,4 ⟩, ⟨ 7,4 ⟩, ⟨ −999,0 ⟩ maka tabel polinom yang dibentuk adalah polinom berderajat 9.



- Polinom “kosong”, yaitu polinom dengan Degree = −999

Menuliskan sebuah polinom

- Prosesnya adalah **proses sekuensial tanpa mark**, traversal untuk *i* [Degree..0].
- Harga suku dituliskan hanya jika koefisiennya tidak nol.
- Contoh:



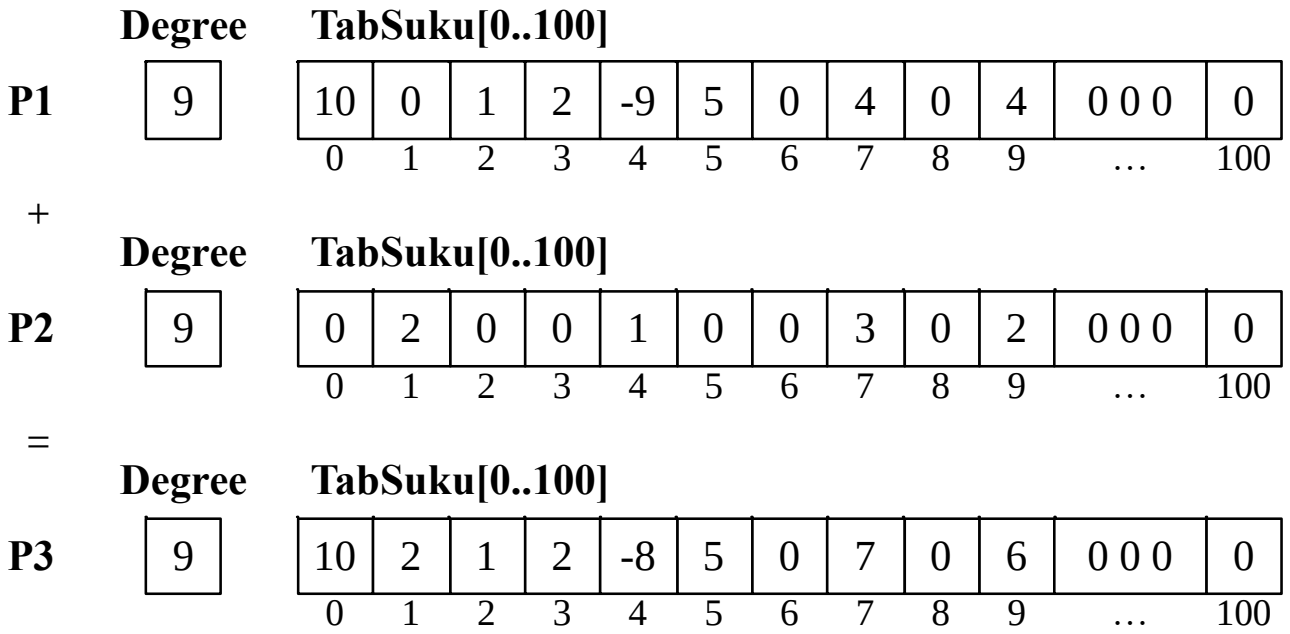
I	P(I)
9	4
8	9
7	4
5	5
4	-9
3	2
2	1
0	10

Menjumlahkan dua buah polinom

- Menjumlahkan dua buah polinom P1 dan P2 dan menyimpan hasilnya pada P3, P3 ≠ P1 dan P3 ≠ P2:
  - menjumlahkan suku P1 dan P2 yang berderajat sama, menjadi suku berderajat tersebut pada P3
  - prosesnya adalah mencari derajat tertinggi dari P1 dan P2, kemudian pemrosesan sekuensial untuk setiap pasangan suku P1 dan P2
  - Penjumlahan memungkinkan hasil berupa polinom kosong atau polinom yang derajatnya “turun”

Menjumlahkan dua buah polinom (2)

- Contoh:
  - P1: ⟨ 9,4 ⟩, ⟨ 7,4 ⟩, ⟨ 5,5 ⟩, ⟨ 4,−9 ⟩, ⟨ 3,2 ⟩, ⟨ 2,1 ⟩, ⟨ 0,10 ⟩
  - P2: ⟨ 9,2 ⟩, ⟨ 7,3 ⟩, ⟨ 4,1 ⟩, ⟨ 1,2 ⟩
- maka P3 = P1+P2 akan mempunyai harga:
  - P3: ⟨ 9,6 ⟩, ⟨ 7,7 ⟩, ⟨ 5,5 ⟩, ⟨ 4,−8 ⟩, ⟨ 3,2 ⟩, ⟨ 2,1 ⟩, ⟨ 1,2 ⟩, ⟨ 0,10 ⟩

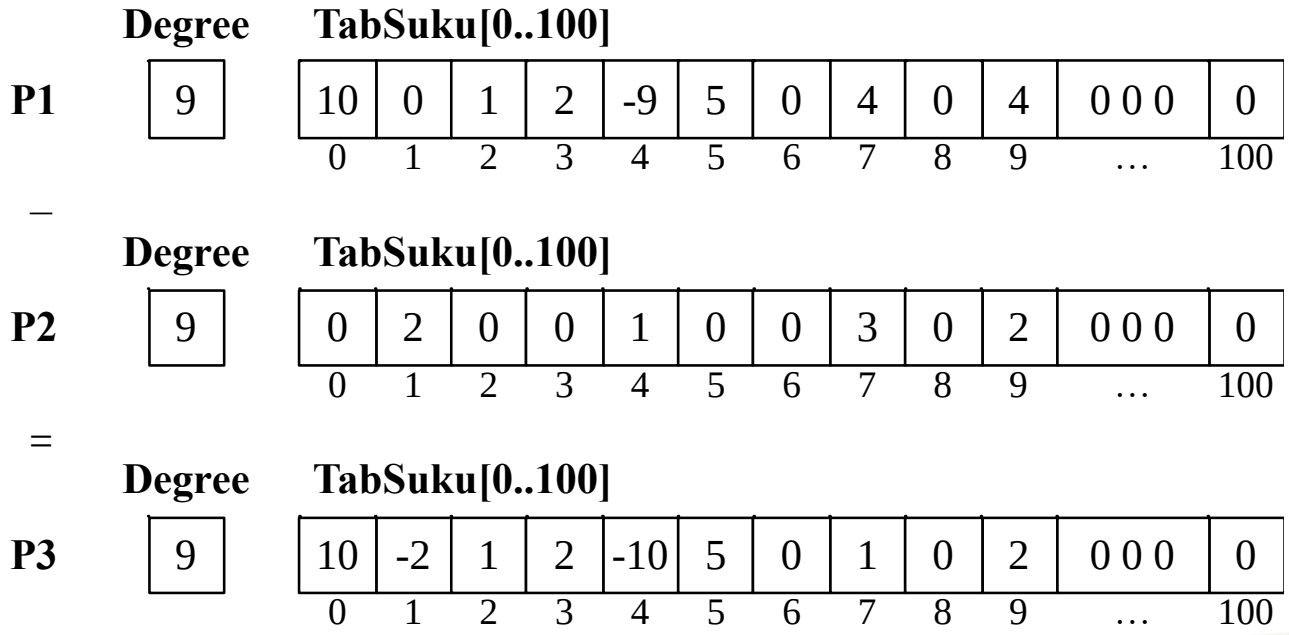


Mengurangi dua buah polinom

- Mengurangi dua buah polinom P1 dan P2 dan menyimpan hasilnya pada P3, P3 $\neq$ P1 dan P3 $\neq$ P2:
  - Prosesnya sama dengan menjumlahkan, hanya operasi penjumlahan diganti operasi pengurangan.
  - Pengurangan juga memungkinkan hasil berupa polinom kosong atau polinom yang derajatnya “turun”.

Mengurangi dua buah polinom (2)

- Contoh:
  - P1: ⟨ 9,4 ⟩, ⟨ 7,4 ⟩, ⟨ 5,5 ⟩, ⟨ 4,−9 ⟩, ⟨ 3,2 ⟩, ⟨ 2,1 ⟩, ⟨ 0,10 ⟩
  - P2: ⟨ 9,2 ⟩, ⟨ 7,3 ⟩, ⟨ 4,1 ⟩, ⟨ 1,2 ⟩
- maka P3 = P1−P2 akan mempunyai harga:
  - P3: ⟨ 9,2 ⟩, ⟨ 7,1 ⟩, ⟨ 5,5 ⟩, ⟨ 4,−10 ⟩, ⟨ 3,2 ⟩, ⟨ 2,1 ⟩, ⟨ 1,−2 ⟩, ⟨ 0,10 ⟩



Membuat turunan polinom

- Membuat turunan P1 dari sebuah polinom P, P1 ≠ P:
  - Prosesnya adalah proses sekuensial, untuk setiap suku ke-i, $i>0$, yaitu $a_i x^i$ pada polinom P, dihitung $i * a_i$ dan disimpan pada indeks ke-$[i-1]$ pada tabel untuk polinom P1

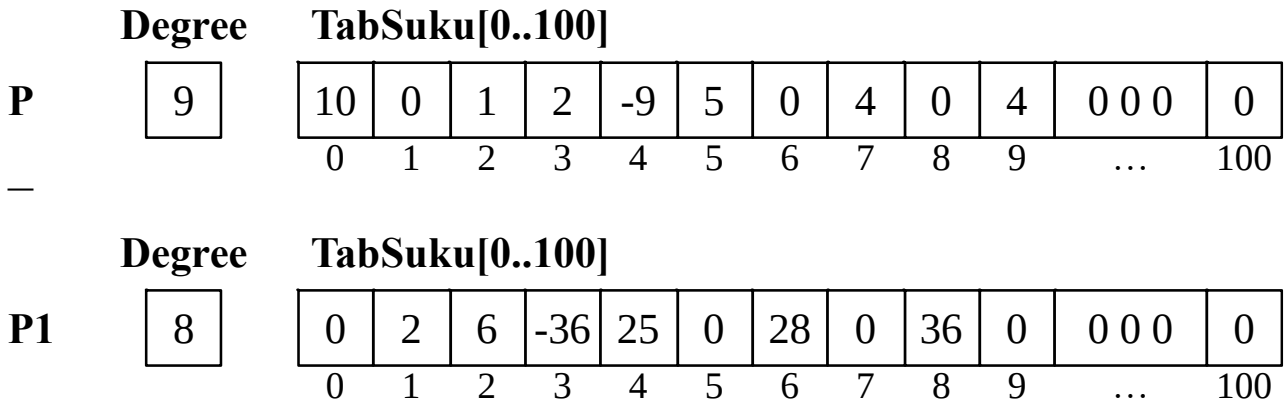
Membuat turunan polinom (2)

- Contoh jika diberikan:

  P: ⟨ 9,4 ⟩, ⟨ 7,4 ⟩, ⟨ 5,5 ⟩, ⟨ 4,−9 ⟩, ⟨ 3,2 ⟩, ⟨ 2,1 ⟩, ⟨ 0,10 ⟩

- maka P1 = P' akan mempunyai harga:

  P1: ⟨ 8,36 ⟩, ⟨ 6,28 ⟩, ⟨ 4,25 ⟩, ⟨ 3,−36 ⟩, ⟨ 2,6 ⟩, ⟨ 1,2 ⟩



```
Program POLINOMIAL                                    { Representasi KONTIGU }
KAMUS
{ Struktur data untuk representasi polinom secara kontigu}
    constant nMax: integer = 100 { Derajat tertinggi polinom yang diproses }
    type Polinom: ⟨  degree: integer,
                    arrSuku: array [0..nMax] of integer  ⟩
    p1,p2: Polinom {Operan}
    p3: Polinom { Hasil }
{ Struktur data untuk interaksi }
    finish: boolean { Mengakhiri proses }
    pilihan: integer [0..5] { Nomor tawaran }
{ Primitif operasi terhadap polinom yang dibutuhkan untuk proses }
    procedure createPolinom (output p: Polinom)
    { Membuat polinom p yang kosong }
    procedure adjustDegree (input/output p: polinom)
    { Melakukan adjustment terhadap Degree. Diaktifkan jika akibat suatu
      operasi, derajat polinom hasil berubah.
{ Primitif operasi terhadap polinom yang disediakan untuk pemakai }
    procedure populatePol (output p: polinom) { Mengisi polinom p }
    procedure displayPol (input p: polinom)   { Menulis polinom p }
    procedure addPol (input p1, p2: polinom, output p3: polinom)
    { Menjumlahkan p1 + p2 dan menyimpan hasilnya di p3, p3 ≠ p1 dan p3 ≠ p2 }
    procedure subPol (input p1, p2: polinom, output p3: polinom)
    { Mengurangkan p1 - p2 dan menyimpan hasilnya di p3, p3 ≠ p1 dan p3≠ p2 }
    procedure derivPol (input p: polinom, output p1: polinom)
    { Membuat turunan p dan menyimpan hasilnya di p1, p1 ≠ p }
```

```
ALGORITMA
    finish ← false
    repeat
        iterate
            output ("Ketik nomor di bawah [0..5] untuk memilih operasi")
            output ("1. Membentuk dua buah polinom P1 dan P2")
            output ("2. Menuliskan polinom P1,P2 dan P3")
            output ("3. Menjumlahkan polinom P1 dan P2 menjadi P3")
            output ("4. Mengurangkan P1 dan P2 menjadi P3")
            output ("5. Membentuk turunan polinom P1 yaitu P3")
            output ("0. Akhir proses")
            input (pilihan)
        stop pilihan ∈ [0..5]
            output ("Ulangi, pilihan di luar harga yang ditawarkan")
        { Pilihan sesuai, maka lakukan proses sesuai dengan Pilihan }
        depend on pilihan
            pilihan = 1: populatePol(p1); populatePol(p2)
            pilihan = 2: displayPol(p1); displayPol(p2); displayPol(p3)
            pilihan = 3: addPol(p1,p2,p3)
            pilihan = 4: subPol(p1,p2,p3)
            pilihan = 5: derivPol(p1,p3)
            pilihan = 0: finish ← true
    until finish
```

Polinom: Representasi Berkait

Representasi Berkait (1)

- Hanya suku yang muncul saja yang disimpan datanya.
- Degree setiap suku harus disimpan secara eksplisit.
- Operasi akan lebih efisien jika suku yang muncul diurut mulai dari derajat tertinggi sampai derajat terendah
- Maka:
  - terjadi penghematan memori kalau suku-suku [0..N] banyak yang tidak muncul. Kalau banyak yang muncul?
  - polinom kosong menjadi sangat “natural”

Representasi Berkait (2)

```
KAMUS
{ Definisi sebuah polinom P adalah }
type Address: ... { terdefinisi alamat sebuah suku }
type Polinom: Address { alamat elemen pertama list }
type Suku: < degree: integer,
             coef: integer,
             next: Address  >
{ Polinom adalah List Suku, dengan elemen terurut menurun   menurut
Degree }
p1, p2, p3: Polinom
```

Membentuk sebuah polinom

- Membentuk sebuah polinom dari pasangan harga yang dibaca dari alat masukan:
  - Setiap kali memasukkan data, yang dimasukkan adalah pasangan

    (*) ⟨ Degree: integer, Coefficient: integer ⟩
  - Akhir pemasukan adalah pasangan harga yang diketikkan bernilai ⟨ −999, 0 ⟩
- Proses pembentukan polinom adalah proses sekuensial untuk membaca dari masukan dan melakukan penyisipan dalam list Suku polinom yang selalu terurut menurun menurut Degree.

Membentuk sebuah polinom

- Contoh: Jika dibaca P1:
  ⟨ 1,4 ⟩, ⟨ 2,5 ⟩, ⟨ 5,7 ⟩, ⟨ 8,9 ⟩, ⟨ 3,4 ⟩, ⟨ –999,0 ⟩
- maka list polinom yang dibentuk adalah polinom berderajat 8 dengan urutan penyisipan:
  ⟨ 1,4 ⟩
  ⟨ 2,5 ⟩, ⟨ 1,4 ⟩
  ⟨ 5,7 ⟩, ⟨ 2,5 ⟩, ⟨ 1,4 ⟩
  ⟨ 8,9 ⟩, ⟨ 5,7 ⟩, ⟨ 2,5 ⟩, ⟨ 1,4 ⟩
  ⟨ 8,9 ⟩, ⟨ 5,7 ⟩, **⟨ 3,4 ⟩**, ⟨ 2,5 ⟩, ⟨ 1,4 ⟩

- Polinom “kosong”, yaitu polinom (P) = nil

Menuliskan sebuah polinom

- Prosesnya menggunakan skema traversal dasar terhadap sebuah list polinom.
- Harga suku yang dituliskan otomatis hanya yang koefisiennya tidak nol.
- Contoh:

```
I    P(I)
8    9
5    7
3    4
2    5
1    4
```

Menjumlahkan dua buah polinom (2)

- Contoh, jika diberikan:
  P1: ⟨ 9,4 ⟩, ⟨ 7,4 ⟩, ⟨ 5,5 ⟩, ⟨ 4,−9 ⟩, ⟨ 3,2 ⟩, ⟨ 2,1 ⟩, ⟨ 0,10 ⟩
  P2: ⟨ 9,2 ⟩, ⟨ 7,3 ⟩, ⟨ 4,1 ⟩, ⟨ 1,2 ⟩
- Maka urutan pembentukan P3 adalah:
  ⟨ 9,6 ⟩ InsertLast (P3,P1+P2), Next(P1), Next(P2)
  ⟨ 7,7 ⟩ InsertLast (P3,P1+P2), Next(P1), Next(P2)
  ⟨ 5,5 ⟩ InsertLast (P3,P1), Next(P1)
  ⟨ 4,−8 ⟩ InsertLast (P3,P1+P2), Next(P1), Next(P2)
  ⟨ 3,2 ⟩ InsertLast (P3,P1), Next(P1)
  ⟨ 2,1 ⟩ InsertLast (P3,P1), Next(P1)
  ⟨ 1,2 ⟩ InsertLast (P3,P2), Next(P2)
  ⟨ 0,10 ⟩ InsertLast (P3,P1), Next(P1)
- Keadaan akhir P3: ⟨ 9,6 ⟩, ⟨ 7,7 ⟩, ⟨ 5,5 ⟩, ⟨ 4,−8 ⟩, ⟨ 3,2 ⟩, ⟨ 2,1 ⟩, ⟨ 1,2 ⟩, ⟨ 0,10 ⟩

Menjumlahkan dua buah polinom (1)

- Menjumlahkan dua buah polinom P1 dan P2 dan menyimpan hasilnya pada P3
- Prosesnya adalah "merging" dua buah list linier yang terurut dan setiap suku P1 dan P2
- Analisis kasus:
  - derajat P1 sama dengan derajat P2, jumlahkan dan sisipkan pada P3 (jika hasil penjumlahan tidak 0), maju ke suku P1 dan P2 yang berikutnya
  - derajat P1 > derajat P2: sisipkan suku P1 pada P3, maju ke suku P1 yang berikutnya
  - derajat P1 < derajat P2: sisipkan suku P2 pada P3, maju ke suku P2 yang berikutnya

Mengurangi dua buah polinom

- Sama dengan menjumlahkan, hanya operasi penjumlahan diganti operasi pengurangan. Contoh, jika diberikan:
  P1: ⟨ 9,4 ⟩, ⟨ 7,4 ⟩, ⟨ 5,5 ⟩, ⟨ 4,−9 ⟩
  P2: ⟨ 9,2 ⟩, ⟨ 7,3 ⟩, ⟨ 4,1 ⟩, ⟨ 3,2 ⟩, ⟨ 2,1 ⟩, ⟨ 1,2 ⟩, ⟨ 0,10 ⟩
- maka urutan pembentukan P3 adalah sebagai berikut:

⟨ 9,2 ⟩	InsertLast(P3,P1−P2), Next(P1), Next(P2)
⟨ 7,1 ⟩	InsertLast(P3,P1−P2), Next(P1), Next(P2)
⟨ 5,5 ⟩	InsertLast(P3,P1), Next(P1)
⟨ 4,−10 ⟩	InsertLast(P3,P1−P2), Next(P1), Next(P2)
⟨ 3,−2 ⟩	InsertLast(P3,P2), Next(P2)
⟨ 2,−1 ⟩	InsertLast(P3,P2), Next(P2)
⟨ 1,−2 ⟩	InsertLast(P3,P2), Next(P2)
⟨ 0,−10 ⟩	InsertLast(P3,P2), Next(P2)

- Dan keadaan akhir P3 adalah: ⟨ 9,2 ⟩, ⟨ 7,1 ⟩, ⟨ 5,5 ⟩, ⟨ 4,−10 ⟩, ⟨ 3,−2 ⟩, ⟨ 2,−1 ⟩, ⟨ 1,−2 ⟩, ⟨ 0,−10 ⟩

Membuat turunan polinom

- Prosesnya adalah proses sekuensial, traversal list P, untuk setiap suku ke-i, $i>0$: yaitu $a_i x^i$ pada polinom P, dihitung $i * a_i$ dan disisipkan P1 sebagai suku ke-$(i-1)$.
- Seperti pada proses penjumlahan, list P1 dibentuk dengan penyisipan elemen terakhir, dan P1 diinisialisasi sebagai list kosong.
  - Karena penyisipan selalu pada akhir list, maka alamat elemen terakhir list P1 selama proses berlangsung layak untuk disimpan.
- Contoh: Jika diberikan:

  P: ⟨ 9,4 ⟩, ⟨ 7,4 ⟩, ⟨ 5,5 ⟩, ⟨ 4,−9 ⟩, ⟨ 3,2 ⟩, ⟨ 2,1 ⟩, ⟨ 0,10 ⟩
- maka P1 akan mempunyai harga:

  ⟨ 8,36 ⟩, ⟨ 6,28 ⟩, ⟨ 4,25 ⟩, ⟨ 3,−36 ⟩, ⟨ 2,6 ⟩, ⟨ 1,2 ⟩

Representasi lojik berkait	Representasi fisik berkait dengan Pointer	Representasi fisik berkait dengan Tabel
KAMUS UMUM **type** Address:... **type** Suku: < degree: **integer**, coef: **integer**, next: Address > **type** Polinom: Address p : Polinom pt: Address	KAMUS UMUM **type** Address: **pointer to** Suku **type** Suku: < degree: **integer**, coef: **integer** next: Address > **type** Polinom: Address p : Polinom pt: Address	KAMUS UMUM **constant** NMAX: **integer**=100 **type** Address: **integer**[0..NMAX] **type** Suku: < degree: **integer**, coef: **integer**, next: Address > **type** Polinom: Address arrSuku: **array**[0..NMAX] **of** Suku firstAvail: Address p : Polinom pt: Address
AKSES: FIRST(p) NEXT(pt) DEGREE(pt) COEF(pt)	AKSES: p pt↑.next pt↑.degree pt↑.coef	AKSES: p arrSuku[pt].next arrSuku[pt].degree arrSuku[pt].coef
PRIMITIF ALOKASI/DEALOKASI: { tergantung rep. fisik }	PRIMITIF ALOKASI/DEALOKASI: { sistem, e.g., malloc/free } pt ← newSuku(...) deallocSuku(pt)	PRIMITIF ALOKASI/DEALOKASI: { Harus direalisasi } initialize(arrSuku) pt ← newSuku(...) deallocSuku(pt)

```
Program POLINOMIAL1
{ Representasi BERKAIT, dengan notasi LOJIK }

KAMUS
{ Struktur data untuk representasi polinom }
    type Address: ... { type terdefinisi }
    type Suku: < degree: integer,
                 coef: integer,
                 next: Address  >
    type Polinom: Address
    constant NIL: Address = ... { untuk address tidak terdefinisi }
    p1, p2: Polinom { operan }
    p3: Polinom     { hasil }
{ Untuk interaksi: }
    finish: boolean          { mengakhiri proses }
    pilihan: integer [0..5] { nomor tawaran }
{ Primitif operasi polinom untuk operasi internal }
    function newSuku () → Address  { Alokasi sebuah suku }
    procedure deallocSuku (input/output pt: Address) { Dealokasi sebuah suku }
    procedure createPolinom (output p: polinom) { Membuat polinom kosong P }
    procedure insertLast (input pt: Address, input/output p: Polinom,
                          input/output last: Address) { Insert pt sesudah
                          elemen terakhir p dengan address elemen terakhir
                          = last }
```

```
{ Primitif operasi polinom yang ditawarkan ke pengguna }
    procedure populatePol (output p1: polinom) { Mengisi polinom p1 }
    procedure displayPol (input p: polinom)  { Menulis polinom p }
    procedure addPol (input p1, p2: polinom, output p3: polinom)
    { Menjumlahkan p1 + p2 dan menyimpan hasilnya di p3, p3 ≠ p1 dan
      p3 ≠ p2 }
    procedure subPol (input p1, p2: polinom, output p3: polinom)
    { Mengurangkan p1 - p2 dan menyimpan hasilnya di p3, p3 ≠ p1 dan
      p3 ≠ p2 }
    procedure derivPol (input p: polinom, output p1: polinom)
    { Membuat turunan p dan menyimpan hasilnya di p1, p1 ≠ p }

ALGORITMA
{ Sama dengan untuk representasi kontigu }
```

Polinom: Diskusi

Studi Kasus:
Suku-Suku Polinom Hasil Berasal dari Operan (1)

- Persoalannya adalah jika pada operasi penjumlahan, pengurangan, dan derivasi pada paket polinom tersebut suku dari polinom hasil berasal dari polinom operan, dan hasilnya disimpan pada salah satu operan. Contoh: Jika P1 dan P2 adalah polinom, maka hendak dilakukan operasi:
  - P1 = P1 + P2
  - P1 = P1 – P2
  - P1 = P1'
- Pada persoalan ini, polinom P1 “tidak ada lagi”, karena “ditimpa” oleh hasil penjumlahan
- Implikasi terhadap perubahan spesifikasi tidak sama untuk representasi kontigu dan representasi berkait.

Studi Kasus:
Suku-Suku Polinom Hasil Berasal dari Operan (1)

- Representasi KONTIGU
  - Untuk representasi kontigu, algoritma penjumlahan dua buah polinom tidak berubah, hanya dengan menggantikan penulisan P3 menjadi P1.

Studi Kasus:
Suku-Suku Polinom Hasil Berasal dari Operan (3)

- Representasi BERKAIT
  - Perubahan spesifikasi hasil operasi tersebut membawa implikasi cukup banyak pada representasi berkait.
  - Prosesnya adalah traversal kedua list, setiap saat kita mengelola dua buah pointer Pt1 untuk traversal Polinom P1 dan Pt2 untuk traversal Polinom P2. Ada empat kemungkinan:
    - Degree(Pt1) > Degree(Pt2): **Tidak ada perubahan terhadap elemen list**, hanya pointer Pt1 yang maju, karena elemen Pt1 tetap menjadi elemen Polinom hasil.
    - Degree(Pt1) = Degree(Pt2)
      - Hasil penjumlahan koefisien tidak sama dengan nol: **Pengubahan nilai koefisien pada Pt1**. Pt1 dan Pt2 maju.
      - Hasil penjumlahan koefisien sama dengan nol: **Penghapusan elemen Pt1** karena hasil penjumlahan adalah nol. Pt2 maju.
    - Degree(Pt1) < Degree(Pt2): **Penambahan elemen baru** dengan degree dan koefisien Pt2 ke polinom P1. Pt2 maju.

Latihan

Soal 1

Tuliskan secara lengkap prosedur/fungsi untuk:

- **createPolinom** (membentuk polinom baru)
- **displayPol** (mencetak polinom)
- **addPol** (menjumlahkan dua buah polinom)
- **subPol** (mengurangkan polinom satu dengan polinom lain)
- **derivPol** (membuat turunan sebuah polinom)

baik untuk:

- Representasi kontigu dengan array (slide 8)
- Representasi berkait (slide 21)