Pengertian Tipe Data Subrange Pascal

Sesuai dengan namanya, **tipe data subrange** merupakan tipe data bentukan yang berasal dari bagian (*sub*) tipe data lain yang berada dalam sebuah jangkauan (*range*).
Sebagai contoh, kita bisa membatasi sebuah tipe data "**satuan**" dimana hanya bisa diisi dengan angka 1 hingga 9. Angka 1 sampai dengan 9 merupakan bagian (*sub*) dari tipe data **integer**. Jika kita mencoba mengisi tipe data "satuan" ini dengan angka selain 1-9, Pascal akan memberikan error.

Cara Penggunaan Tipe Data Subrange Pascal

Untuk membuat tipe data **subrange**, caranya hampir sama dengan tipe data **enumerated**, dimana kita harus mendefenisikannya di bagian **type**, kemudan baru membuatnya sebagai variabel di bagian **var**. Berikut contohnya:

```
1
2 program tipe_subrange;
3 uses crt;
4 type
5   satuan= 1..9;
6 var
7   a,b: satuan;
8 begin
9   clrscr;
10
11  a:= 2;
12  writeln('a: ',a);
13
14  b:= 7;
15  writeln('b: ',b);
16
17  readln;
18 end.
```

Dalam kode program diatas, saya mendefenisikan sebuah tipe data *subrange* '**satuan**' dengan angka 1 hingga 9. Tanda titik dua '**..**' digunakan untuk membatasi seberapa jauh jangkauan untuk satuan ini.
Jika saya ingin membuat tipe data '**puluhan**' yang terdiri dari angka 11 hingga 99 maka penulisannya adalah sebagai berikut:

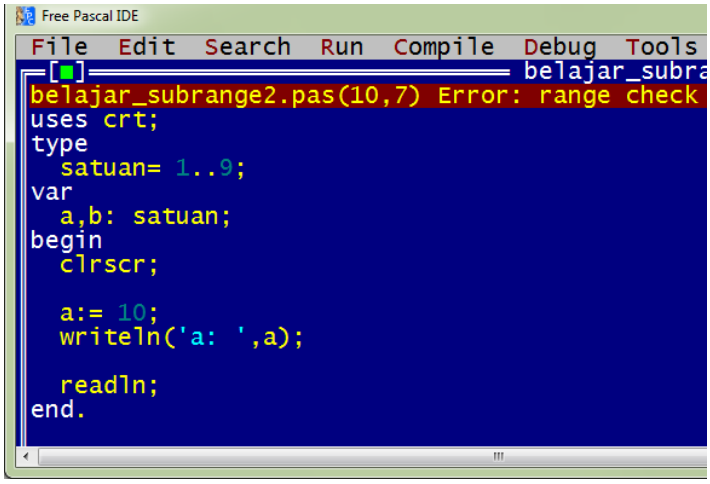
```
type
  satuan= 11..99;
```

Apa yang terjadi jika kita memberikan angka diluar jangkauan tipe data *subrange*? Berikut contohnya:

```
program tipe_subrange;
uses crt;
type
  satuan= 1..9;
var
  a,b: satuan;
begin
  clrscr;

  a:= 10;
  writeln('a: ',a);

  readln;
end.
```



Hasilnya, compiler **FreePascal** akan menghasilkan ***error: range check error while evaluating constants***, yang artinya nilai tersebut berada di luar *range* (jangkauan) yang telah ditetapkan, dimana saya mencoba memberikan nilai 10.

Kita tidak dibatasi untuk menggunakan angka saja, tapi bisa juga menggunakan karakter huruf sebagai tipe data **subrange**. Berikut contohnya:

```
program tipe_subrange;
uses crt;
type
  huruf= 'A'..'F';
var
  a,b: huruf;
begin
  clrscr;

  a:= 'A';
  writeln('a: ',a);

```

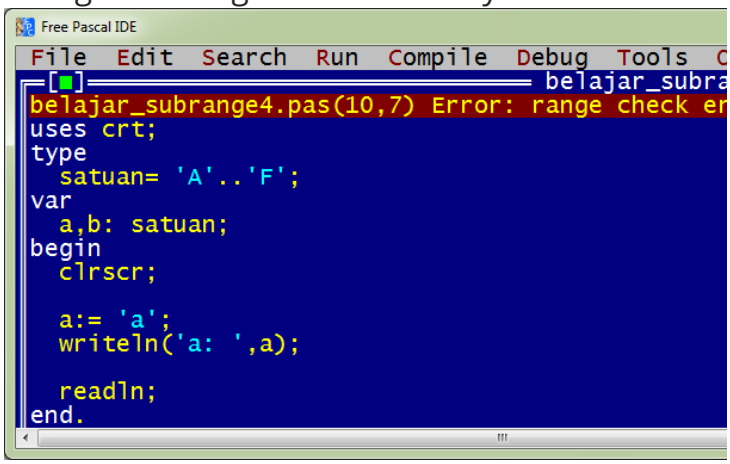
```
12   B:= 'C';
13   writeln('b: ',b);
14
15   readln;
16 end.
17
```

Kali ini saya membuat tipe data *subrange* **'huruf'** yang dibatasi dari huruf '**A**' hingga '**F**'. Perhatikan tanda kutip untuk huruf-huruf ini, karena huruf tersebut adalah bagian (*subrange*) dari tipe data char yang harus dibuat dengan tanda kutip.

Yang juga perlu diperhatikan, huruf '**A**' tidak sama dengan huruf '**a**', seperti contoh kasus berikut:

```
1
2  program tipe_subrange;
3  uses crt;
4  type
5    huruf= 'A'..'F';
6  var
7    a,b: huruf;
8  begin
9    clrscr;
10
11   a:= 'a';
12   writeln('a: ',a);
13
14   readln;
15 end.
```

Kali ini saya mencoba memberikan nilai '**a**' kepada variabel **a** yang didefenisikan sebagai subrange **'huruf'**. Hasilnya?



Pascal kembali komplain karena huruf '**a**' tidak ada di dalam *subrange*.

Walaupun sehari-hari kita menyamakan ‘A’ (huruf A besar) dengan ‘a’ (huruf ‘a’ kecil), komputer tidak memahami hal ini. Jika anda telah mempelajari tutorial tentang tipe data **char**, kedua karakter ini sebenarnya memiliki nomor **ASCII** yang berbeda, sehingga hasilnya tidak sama.

Cara Penggunaan Tipe Data Subrange dari Enumerated

Untuk membuatnya lebih kompleks (dan juga lebih fleksibel), kita bisa membuat tipe data **subrange** dari yang isinya berasal dari tipe data **enumerated**. Langsung saja kita masuk ke contoh programnya:

```
1
2program tipe_enumeration;
3uses crt;
4type
5  nama_hari= (senin,selasa,rabu,kamis,jumat,sabtu,minggu);
6  hari_kerja= senin..jumat;
7  weekend= sabtu..minggu;
8var
9  a:hari_kerja;
10  b:weekend;
11begin
12  clrscr;
13
14  a:= kamis;
15  writeln('a: ',a);
16
17  b:= sabtu;
18  writeln('b: ',b);
19
20  readln;
end.
```

Contoh diatas saya ambil dari kode program dari tutorial tipe data enumeration, dimana saya mendefenisikan **nama_hari** sebagai tipe data **enumerated**. Namun kali ini saya membuat tipe data *subrange* **hari_kerja** dan **weekend** dari **nama_hari**. Konsep penggabungan **enumerated** dan **subrange** ini mungkin cukup rumit, tapi jika anda telah memahami tutorial sebelumnya tentang enumeration, saya rasa bisa memahami cara kerja kode program diatas. Silahkan diubah-ubah dan lihat bagaimana hasilnya.

Selanjutnya, kita akan membahas tipe data array, yang (hampir) selalu hadir dalam setiap bahasa pemrograman.

Pengertian Tipe Data Array Pascal

Tipe data array adalah tipe data bentukan yang terdiri dari kumpulan tipe data lain. Daripada membuat 10 variabel yang terdiri dari *nama1*, *nama2*, *nama3*, dst, akan lebih efisien jika variabel nama ini disimpan ke dalam array.
Sebagai contoh, perhatikan kode program berikut ini:

```
1
2program tipe_array;
3uses crt;
4var
5  nilai1, nilai2, nilai3, nilai4: integer;
6begin
7  clrscr;
8
9  nilai1:= 23;
10 nilai2:= 13;
11 nilai3:= 98;
12 nilai4:= 106;
13
14 writeln('nilai1: ',nilai1);
15 writeln('nilai2: ',nilai2);
16 writeln('nilai3: ',nilai3);
17 writeln('nilai4: ',nilai4);
18
19 readln;
20end.
```

Dalam contoh tersebut saya membuat 4 variabel: *nilai1, nilai2, nilai3* dan *nilai4*. Keempat variabel ini bertipe **integer**.
Tidak ada yang salah dari kode program diatas. Tapi bayangkan apabila kita ingin menyimpan lebih dari 4 nilai, bagaimana jika 10 atau 100 nilai? Tentu tidak efisien jika kita harus membuat variabel *nilai1, nilai2, nilai3*... sampai dengan *nilai100*. Untuk hal inilah tipe data array lebih cocok digunakan.

Cara Penggunaan Tipe Data Array Pascal

Untuk membuat tipe data array di pascal, kita harus menentukan seberapa banyak element array yang ingin dibuat. **Element** adalah sebutan untuk 'anggota' / isi dari array. Sebagai contoh, untuk membuat 10 element array bertipe integer saya bisa menggunakan kode berikut:

```
1var
2  nilai: array[0..9] of integer;
```

Sekarang, variabel 'nilai' berisi array dengan 10 element bertipe integer. Perhatikan angka 0..9, ini berarti saya membuat element array dari element 0, element 1, element 2, element 3,.. hingga element 9 (total terdapat 10 element).

Bagaimana cara mengakses element ini? kita bisa mengaksesnya melalui nomor **index**. **Index** adalah urutan element di dalam sebuah array. Sebagai contoh, untuk mengakses element ke – 2, kita bisa menulis: **nilai[2]**. Untuk mengakses element ke-6, bisa menggunakan: **nilai[6]**.
Berikut contoh kode program pascal cara penggunaan tipe data array:

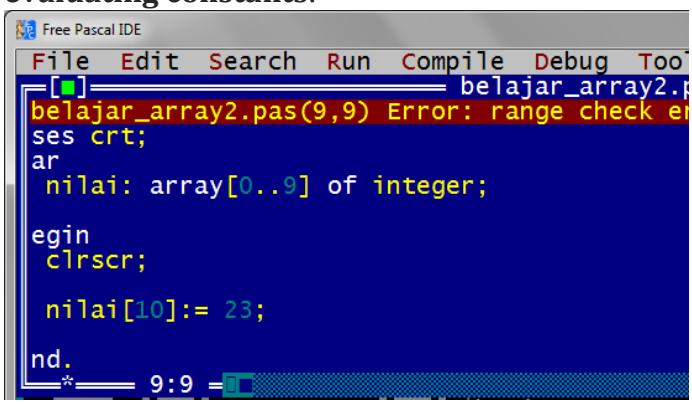
```
1
2program tipe_array;
3uses crt;
4var
5  nilai: array[0..9] of integer;
6
7  begin
8  clrscr;
9
10 nilai[0]:= 23;
11 nilai[1]:= 13;
12 nilai[2]:= 98;
13 nilai[3]:= 106;
14
15 writeln('nilai0: ',nilai[0]);
16 writeln('nilai1: ',nilai[1]);
17 writeln('nilai2: ',nilai[2]);
18 writeln('nilai3: ',nilai[3]);
19
20 readln;
end.
```

Pada contoh diatas, saya membuat variabel '**nilai**' sebagai *array* yang berisi 10 element **integer**. Di dalam variabel '**nilai**' ini, index array dimulai dari 0 hingga 9, karena saya menulisnya dengan **array[0..9] of integer**. Jika anda ingin membuat 100 element array, bisa menulisnya sebagai **array[0..99] of integer**.
Walaupun saya membuat 10 element, tapi kita tidak harus mengisi semua element ini. Pada contoh tersebut, saya hanya mengisi 4 element. Bagaimana dengan element lainnya? ini akan menggunakan nilai *default* (bawaan) pascal, biasanya berisi angka 0 untuk tipe data integer.
Selain itu, kita juga tidak harus mengisinya secara berurutan. Kita bisa mengisi element-element array ini secara acak, selama masih dalam batas yang ditetapkan. Berikut contohnya:

```
1
2program tipe_array;
3uses crt;
4var
5  nilai: array[0..9] of integer;
6
7  begin
8  clrscr;
9
10  nilai[3]:= 23;
11  nilai[9]:= 13;
12  nilai[2]:= 98;
13  nilai[0]:= 106;
14
15  writeln('nilai3: ',nilai[3]);
16  writeln('nilai9: ',nilai[9]);
17  writeln('nilai2: ',nilai[2]);
18  writeln('nilai0: ',nilai[0]);
19
20  writeln('nilai1: ',nilai[1]);
21  writeln('nilai7: ',nilai[7]);
22
23  readln;
24end.
```

Kali ini saya mengisi element secara acak, dan juga saya menampilkan element **nilai[1]** dan **nilai[7]** yang memang tidak diisi dengan nilai. Hasilnya? pascal akan menggunakan nilai default: 0.

Bagaimana jika kita melewati batas element array? Misalnya saya mengakses element ke 10? Pascal akan meneluarkan **error: Range check error while evaluating constants**.



Kita tidak hanya bisa membuat array bertipe integer saja, tapi juga bisa menggunakan tipe lain seperti **real**, **char** atau **string**. Berikut contohnya:

```
1program tipe_array;
```

```
uses crt;
var
  kata: array[20..29] of string[20];
begin
  clrscr;

  kata[24]:= 'Sedang ';
  kata[25]:= 'belajar pascal ';
  kata[26]:= 'di ';
  kata[27]:= 'Deka Ok TV';

  write(kata[24]);
  write(kata[25]);
  write(kata[26]);
  write(kata[27]);

  readln;
end.

```

Saya membuat variabel kata dengan array berjumlah 10 element yang masing-masing isinya adalah **string[20]**.
Anda bisa lihat bahwa saya menggunakan penomoran array mulai dari 20 hingga 29. Ini tidak menjadi masalah, selama kita juga mengaksesnya dengan index yang sesuai. Berikut hasil yang didapat:

Pembahasan mengenai array cukup banyak, oleh karena itu saya akan pecah menjadi beberapa tutorial. Berikutnya kita akan membahas cara pembuatan array 2 dimensi di dalam pascal

Array 1 Dimensi Pascal

Contoh array yang telah kita pelajari pada tutorial sebelumnya adalah **array 1 dimensi**, dimana setiap element array hanya terdiri satu 'lapis', seperti contoh berikut:

```
1
2program tipe_array;
3uses crt;
4var
5  nilai: array[0..2] of integer;
6begin
7  clrscr;
8
9  nilai[0]:= 10;
10  nilai[1]:= 20;
11  nilai[2]:= 30;
12
13  writeln('nilai1: ',nilai[0]);
14  writeln('nilai2: ',nilai[1]);
15  writeln('nilai3: ',nilai[2]);
16
17  readln;
18end.
```

Maksud dari 1 dimensi disini adalah, setiap element array dibahas dengan 1 index, seperti **nilai[0]**, **nilai[1]** dan **nilai[2]**.

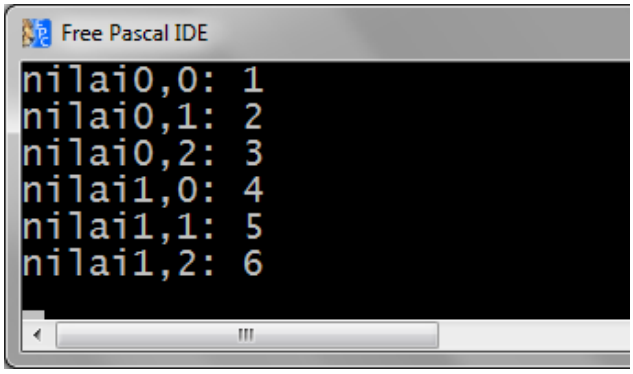
Array 2 Dimensi Pascal

Untuk penggunaan yang lebih rumit, array 1 dimensi tidak cocok lagi. Sebagai contoh, di dalam matematika kita menggunakan *grafik/diagram kartesius* yang titik koordinatnya menggunakan komposisi sumbu x dan sumbu y. Sebagai contoh **A(3,4)** berarti titik A berada di posisi 3 pada sumbu x, dan 4 pada sumbu y.
Untuk mempermudah dalam pembuatan program yang melibatkan 2 sumbu atau 2 dimensi ini, kita bisa menggunakan **array 2 dimensi**.
Cara penulisan array 2 dimensi adalah dengan menuliskan dua angka (dua jangkauan) sewaktu deklarasi array. Berikut contohnya:

```
1var
2  nilai: array[0..1,0..2] of integer;
```

Kode diatas berarti saya membuat variabel **'nilai'** sebagai array 2 dimensi. Dimana untuk dimensi pertama berisi 0 dan 1, sedangkan di dimensi kedua berisi 0, 1 dan 2. Total, variabel 'nilai' berisi 6 element (hasil dari 2 * 3).
Cara mengakses element pada array 2 dimensi ini menggunakan tanda koma sebagai pemisah, seperti: **nilai[0,2]** atau **nilai[1,1]**.
Contoh berikut akan memperjelas cara penggunaanya:

```
1
2program tipe_array;
3uses crt;
4var
5  nilai: array[0..1,0..2] of integer;
6begin
7  clrscr;
8
9  nilai[0,0]:= 1;
10  nilai[0,1]:= 2;
11  nilai[0,2]:= 3;
12  nilai[1,0]:= 4;
13  nilai[1,1]:= 5;
14  nilai[1,2]:= 6;
15
16  writeln('nilai0,0: ',nilai[0,0]);
17  writeln('nilai0,1: ',nilai[0,1]);
18  writeln('nilai0,2: ',nilai[0,2]);
19  writeln('nilai1,0: ',nilai[1,0]);
20  writeln('nilai1,1: ',nilai[1,1]);
21  writeln('nilai1,2: ',nilai[1,2]);
22
23  readln;
24end.
```



Silahkan anda pelajari sejenak kode diatas, baik cara pembuatan array 2 dimensi maupun cara mengakses tiap-tiap elemennya. Menggunakan array 2 dimensi ini akan memudahkan kita untuk membuat kode program yang lebih kompleks.

Contoh lain yang sering menggunakan array 2 dimensi adalah untuk membuat **struktur matriks**. Berikut contohnya:

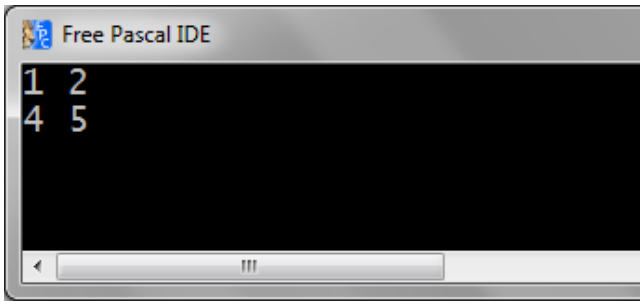
```
1program tipe_array;
2uses crt;
```

```
var
  nilai: array[0..1,0..1] of integer;
begin
  clrscr;

  nilai[0,0]:= 1;
  nilai[0,1]:= 2;
  nilai[1,0]:= 4;
  nilai[1,1]:= 5;

  write   (nilai[0,0],' ');
  writeln(nilai[0,1]);
  write   (nilai[1,0],' ');
  writeln(nilai[1,1]);

  readln;
end.
```



Contoh kali ini hampir mirip dengan contoh kode program pascal sebelumnya, tapi saya membatasi dengan element 2×2 (perhatikan cara pendeklarasikan variabel **'nilai'**). Ketika menampilkan hasil array, saya menyusunnya agar sesuai dengan bentuk matriks 2×2. Ini didapat dengan perpaduan perintah **write** dan **writeln**.
Sebagai latihan, dapatkah anda membuat struktur matriks 3×3?

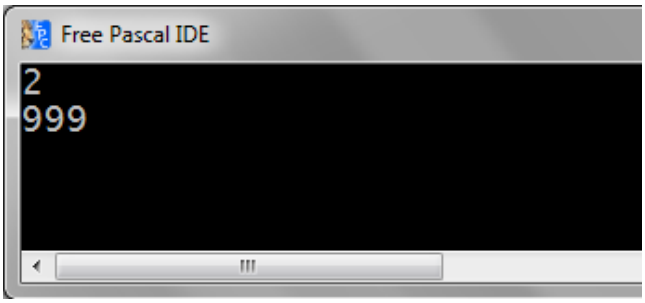
Array 3 Dimensi Pascal

Secara teori, dimensi untuk array di dalam pascal tidak terbatas. Kita juga bisa membuat array 3 dimensi. Ini diperlukan jika koordinat cartesius terdiri dari *sumbu* ***x***, *sumbu* ***y***, dan *sumbu* ***z***. Cara pembuatannya juga hampir sama. Sebagai contoh, jika saya ingin membuat array **3 dimensi 2x3x4** bisa ditulis sebagai:

```
var
  nilai: array[0..1,0..2,0..3] of integer;
```

Sampai disini, saya yakin anda sudah paham maksud kode diatas. Cara pengaksesannya pun cukup dengan menambahkan dimensi ketiga di nomor index, seperti nilai[1,2,3] atau nilai[0,0,1]. Berikut contohnya:

```
1
2 program tipe_array;
3 uses crt;
4 var
5   nilai: array[0..1,0..2,0..3] of integer;
6 begin
7   clrscr;
8
9   nilai[0,0,2]:= 2;
10  writeln(nilai[0,0,2]);
11
12  nilai[1,2,3]:= 999;
13  writeln(nilai[1,2,3]);
14
15  readln;
16 end.
```



Setelah mempelajari cara pembuatan array 2 dimensi (dan 3 dimensi), dalam tutorial pascal berikutnya saya akan membahas cara pembuatan array dinamis (*Dynamic Arrays*).