Record

(TIB26 – ALGORITMA PEMROGRAMAN)

Pertemuan 9, 10

Sub-CPMK

- Mahasiswa mampu membuat struktur record dan mendeklarasikan sebagai variabel (C3, A3)

Materi

- Pengertian Tipe Data Bentukan
- Struct
- Array Struct

Perhatian

- Tidak disarankan copy-paste kode program dari presentasi ini, karena ada beberapa symbol yang dianggap sebagai Unicode oleh editor yang anda gunakan, sehingga akan dianggap sebagai symbol yang salah oleh compiler, sebaiknya diketik ulang saja

1.

Pengertian Tipe Data Bentukan

1.1. Tipe Data

- Tipe data :
  - Nilai yang mungkin terisi ke variabel
- Variabel agar dapat digunakan harus dideklarasikan sesuai dengan tipe data yang akan ditampungnya
- Suatu variabel tidak dapat menampung data yang tidak sesuai dengan tipe data peruntukannya
- Ada dua macam tipe data
  A. Tipe data sederhana/primitif
  B. Tipe data bentukan

1.2. Tipe Data Sederhana

Merupakan tipe data bawaan dari bahasa pemrograman.

Beberapa tipe data primitif (ada yang menyebutnya tipe data sederhana) yang umum terdapat pada berbagai bahasa pemrograman

- *Boolean* → Tipe data yang hanya memperbolehkan dua nilai 1/0 atau *TRUE/FALSE* saja
- *Character* → menampung 8 bit data yang diterjemahkan menjadi karakter, *Character* termasuk tipe data *integer*
- *Integer* → bilangan bulat, Terdapat beberapa jenis bilangan *integer* berdasarkan panjang bit nya: *Byte*, *short integer*, *integer*, *long integer*
- Pecahan → bilangan pecahan, umumnya direpresentasikan dalam bentuk *floating point*, berdasarkan panjang dan ketelitiannya, *floating point* dapat dibagi menjadi *single precision* (32 bit) dan *double precision* (64 bit)

1.3. Tiga Kategori Tipe Data Primitif

- *Integral* (bulat)

  Tipe data yang memperlakukan *integer* atau bilangan tanpa bagian, contoh *integer*, *char* dan boolean

- Pecahan

  Dinyatakan dalam bentuk *Floating – point*, contoh *single*, *double*, *real*

- *Enumeration* (enumerasi)

  user-defined data type. Contoh:

  enum bulan {JAN, PEB, MAR, APR, MEI, JUN, JUL, AGU, SEP, OKT, NOP, DES};

Tipe Data Bentukan

- Tipe Data bentukan adalah tipe data yang dibentuk dari tipe data lainnya.
- Tipe data ini dibentuk menjadi struktur
- Dapat berisi satu atau lebih field dengan tipe data yang sama ataupun tipe data yang berbeda

2.

Struct

2.1 *Record / Structure*

- Rekaman atau *record* atau *structure* adalah sekumpulan data yang disusun dari tipe data yang sama atau tipe data yang berbeda.
- Sebuah record berisi beberapa variabel lain yang ‘dipaketkan’. Konsep struktur data seperti ini sedikit mirip dengan konsep class dan object dalam *object oriented programming*
- *Record/Structure* harus di definisikan terlebih dahulu

2.1 *Record / Structure (lanj…)*

- Hasil definisi *Record*/*Structure* diperlakukan seperti tipe data,
- Ketika akan digunakan, *Record*/*Structure* harus dideklarasikan dahulu pada sebuah variable
- Struct / Record yang sudah didefiniskan dapat di deklarasikan menjadi sebuah variable tunggal ataupun sebagai Array sebagaimana layaknya tipe data lainnya

2.2. Mengakses *Record*

- *Record* diakses pada *field-field*nya
- *Record* dapat diakses dengan menyebutkan terlebih dahulu nama *variable* diikuti nama *field* yang akan diakses setelah didahului tanda titik

2.3. *Record* dalam C++

- Definisi

```
struct RecordName
{
     vartype FieldName1;
     vartype FieldName2;
     vartype FieldName3;
     ...
     vartype FieldNameN;
};
```

- Deklarasi

```
RecordName varRecord;
```

- Penugasan

```
varRecord.FieldNameN = data;
```

- Mengakses Record

```
VarData = varRecord.FieldNameN;
```

2.4. *Record* dalam C++ (Lanj.)

- Definisi

```
struct Bangun
{
     int x1;
     int y1;
     int x2;
     int y2;
     int x3;
     int y3;
};
```

- Deklarasi

```
Bangun Segitiga;
```

- Penugasan

```
Segitiga.x1 = 10;
Segitiga.y1 = 16;
```

- Mengakses Record

```
Temp = Segitiga.x1;
```

3.

Array Struct

Array Struct

- Structure / Record yang sudah didefinisikan dapat dideklarasikan menjadi array
- Untuk mendeklarasaikan menjadi array dapat dilakukan dengan syntax:

```
struct namastruct namaarray[ukuranarray]
```

3.2. *Record With Array - C*

- Record definition

```
struct StructName
{
    vartype Var1Name;
    vartype Var2Name;
    vartype VarNName;
};
```

- Array declaration

```
struct StructName DataCell[ArraySize];
```

- Assignment

```
DataCell[ArrayNum].VarName = value;
```

- Accessing

```
DataCell[ArrayNum].VarName
```

- Example

```
//Struct definition
struct TheCell
{
    char Name[10];
    int Age;
};
//Declaration
struct TheCell DataMhs[250];

void main()
{
    //Assignment
    strcpy(DataMhs[1].Name,"Doraemon");
    DataMhs[1].Age = 19;
    //Accessing
    printf("%s", DataMhs[1].Name);
    printf("%d", DataMhs[1].Age);
}
```

Ringkasan

- Rekaman atau *record* atau *structure* adalah sekumpulan data yang disusun dari tipe data yang sama atau tipe data yang berbeda.
- *Record*/*Structure* harus di definisikan terlebih dahulu
- Hasil definisi *Record*/*Structure* diperlakukan seperti tipe data, sehingga ketika akan digunakan, *Record*/*Structure* harus dideklarasikan dahulu pada sebuah variabel

Terimakasih

TUHAN Memberkati Anda

Teady Matius Surya Mulyana (tmulyana@bundamulia.ac.id)