Modul 3

Tipe Data, Penamaan, dan Sekuens

PJ : Vesha Halvin Winrich Chandra

Tujuan

1. Praktikan dapat memahami konsep tipe data, penamaan, dan sekuens.
2. Praktikan dapat menerapkan konsep tipe data, penamaan, dan sekuens dalam pembuatan program.

Teori

1. Tipe Data

Tipe data merujuk pada klasifikasi data berdasarkan jenis-jenis data yang tersedia di dunia nyata dan dapat diaplikasikan ke dalam sebuah wadah bernama **Variabel**. Tipe data berperan penting karena membantu *compiler* untuk menentukan cara penggunaan data tersebut. Dalam penggunaan tipe data, tipe data dapat dibagi menjadi 2 bagian, yaitu **Tipe Data Primitif** dan **Tipe Data Bentukan**.

- **Tipe Data Primitif**

Tipe data ini adalah **tipe data paling dasar** yang disediakan oleh bahasa pemrograman yang digunakan dan **paling umum**. Berikut tipe data primitif yang tersedia di bahasa pemrograman C:

Tipe Data	Keyword	Format Specifier	Range Data
Integer	`int`	`%d`	-2,147,483,648 s.d 2,147,483,647
Float	`float`	`%f`	1.2E-38 s.d 3.4E+38
Character	`char`	`%c`	-128 s.d 127
Boolean	`bool`	`%d`	True (1) / False (0)

- **Integer**

  Merupakan tipe data yang melambangkan **bilangan bulat** dalam bahasa C, memakan memory sebanyak **4 bytes per variabel** yang menggunakan tipe data ini.

  Contoh: **-135, -1, 0, 1, 135**

Terdapat pula tipe data integer yang dapat menyimpan lebih banyak maupun lebih sedikit dari integer, yaitu:

Tipe Data	Keyword	Memory (bytes)	Format Specifier	Range Data
Short Integer	`short int`	2	`%hd`	-32,768 s,d 32,767
Unsigned Short Integer	`unsigned short int`	2	`%hu`	0 s.d 65,535
Unsigned Integer	`unsigned int`	4	`%u`	0 s.d 4,294,967,295
Long Integer	`long int`	4	`%ld`	-2,147,483,648 s.d 2,147,483,647
Long Long Integer	`long long int`	8	`%lld`	–9,223,372,036,854,775,807 s.d 9,223,372,036,854,775,807
Unsigned Long Integer	`unsigned long int`	4	`%lu`	0 s.d 4,294,967,295
Unsigned Long Long Integer	`unsigned long long int`	8	`%llu`	0 s.d 18,446,744,073,709,551,615

Arti dari tulisan **unsigned** adalah suatu **bilangan bulat** yang **tidak memiliki tanda positif** maupun **negatif**, sehingga tipe data yang diberi unsigned hanya dapat menampung **bilangan bulat positif dari 0** saja, sehingga dapat diperhatikan saat mau menggunakan suatu variabel agar dapat dipilih sesuai kebutuhan saja.

Sedangkan maksud dari kata **long** tersebut adalah untuk **menambahkan batas maksimal penyimpanan data integer** yang awalnya hanya -2,147,483,648 s.d 2,147,483,647 menjadi range yang tersedia di tabel tersebut, **short** juga memiliki kemiripan dengan long, namun short digunakan untuk **mengurangi range yang integer punya**.

- **Float**

  Merupakan tipe data yang berlambangkan **bilangan real** pada bahasa C, memakan memory sebanyak **4 bytes per variabel** yang menggunakan tipe data ini.

  Contoh: **1.0, 1.123, 200.32, -3.687, -1002.102**

  Terdapat juga tipe data integer yang dapat menyimpan lebih banyak maupun lebih sedikit dari float, yaitu:

Tipe Data	Keyword	Memory (bytes)	Format Specifier	Range Data
Double	`double`	8	`%lf`	1.7E-308 s.d 1.7E+308
Long Double	`long double`	16	`%Lf`	3.4E-4932 s.d 1.1E+4932

  Tipe data float dan double memiliki fungsi utama untuk menyimpan suatu data bersifat **bilangan real**, namun float hanya dapat menampung **7 digit desimal** dengan presisi, dan untuk double dapat menampung **16 digit desimal** dengan presisi. Perbedaan kedua ini sangat berpengaruh dan perlu diperhatikan di saat memilih suatu tipe data, agar program tidak meng-outputkan hasil yang salah.

- **Character**

  Merupakan tipe data yang melambangkan suatu character atau alfabet di bahasa C, Memakan memory sebanyak 1 bytes per variabel yang menggunakan tipe data ini.

  Contoh: '**A**', '**I**', '**K**', '**-**', '**?**', '**[**'

  Terdapat juga tipe data integer yang dapat menyimpan lebih banyak maupun lebih sedikit dari character, yaitu:

Tipe Data	Keyword	Memory (bytes)	Format Specifier	Range Data
Signed Char	`signed char`	1	`%c`	-128 s.d 127
Unsigned Char	`unsigned char`	1	`%c`	0 s.d 255

Signed char dapat diartikan juga sebagai char, dikarenakan range data yang masih sama sehingga tidak harus dituliskan signed secara default untuk menggunakan char. Lalu mengapa char memberikan range data angka tapi mengeluarkan sebuah alfabet?

Untuk menjawab pertanyaan tersebut, char menggunakan sistem yang bernama **kode ASCII** untuk menampilkan character yang ditampilkan saat aplikasi berjalan. Untuk selebihnya **kode ASCII** bisa di-searching mandiri.

- **Boolean**

  Merupakan tipe data tambahan dari library `<stdbool.h>` perlu menginclude library tersebut untuk menggunakan boolean, namun dapat digantikan dengan 1 dan 0 bila tidak mau menggunakan boolean ini, karena pada dasarnya boolean true dan false adalah 1 dan 0.

- **Tipe Data Bentukan**

  Merupakan tipe data yang dibentuk oleh programmer sendiri untuk memenuhi suatu kebutuhan. Tipe data ini merupakan turunan dari tipe data primitif atau tipe data bentukan lainnya. Tipe data baru ini dapat dibuat dengan keyword `typedef` atau menambahkan record yang dapat dibuat dengan keyword `struct` .

2. Variabel

Merupakan suatu wadah untuk menampung data yang ditentukan oleh tipe data, variabel ini perlu yang namanya **Deklarasi Variabel**, yaitu proses memperkenalkan / pembuatan. Pendeklarasian variabel dapat dilakukan dengan mengetikan tipe data terlebih dahulu lalu nama variabel yang diinginkan. Contoh:

```c
#include <stdio.h>
#include <stdlib.h>

int main(){
    int nilai; //pendeklarasian variabel integer

    return 0;
}
```

Gambar 1. Pendeklarasian variabel integer

Pendeklarasian variabel ini sendiri juga bisa menambahkan banyak integer sekaligus dalam satu baris bila ingin membuat suatu variabel dengan tipe yang sama. Contoh:

```c
#include <stdio.h>
#include <stdlib.h>

int main(){
    int tahun, bulan, hari; //pendeklarasian variabel integer

    return 0;
}
```

Gambar 2. Pendeklarasian beberapa variabel integer

Terdapat juga tipe data bentukan yang dapat di deklarasi menggunakan `typedef` di bagian paling atas program. Contoh:

```c
#include <stdio.h>
#include <stdlib.h>

typedef int npm; //pendeklarasian tipe data bentukan dengan npm sebagai integer

int main(){
    npm mahasiswa; //pendeklarasian variabel npm

    return 0;
}
```

Gambar 3. Pendeklarasian variabel data bentukan

Pendeklarasian tipe data bentukan ini juga diperlukan untuk menyimpan suatu kalimat, yang dapat dibentuk dari char menggunakan array. Contoh:

```c
#include <stdio.h>
#include <stdlib.h>

typedef char string[128]; //pendeklarasian tipe data bentukan string dari char

int main(){
    string nama; //pendeklarasian variabel nama dengan tipe data bentukan string

    return 0;
}
```

Gambar 4. Pendeklarasian variabel string

Untuk pengaksesan suatu tipe data bentukan masih sama sesuai dengan tipe data primitif yang dipakai saat membuat suatu tipe data bentukan baru tersebut, namun **khusus string** menggunakan **format specifier "`%s`" untuk menampilkan kalimat** yang sudah di simpan pada string tersebut.

3. Assignment, Inisialisasi, dan Input

Setelah variabel dideklarasikan, kita dapat melakukan inisialisasi, assignment, bahkan operasi pada variabel tersebut. **Assignment** sendiri adalah suatu proses pemberian nilai terhadap suatu variabel. Jika suatu variabel dideklarasikan dan dilakukan proses assignment, maka proses tersebut disebut dengan **inisialisasi**. Contoh:

```c
#include <stdio.h>
#include <stdlib.h>

typedef char string[128]; //pendeklarasian tipe data bentukan string dari char

int main(){
    int a,b; //pendeklarasian variabel integer
    string nama = "Vesha";
    // ^ inisialisasi variabel string

    a = 1;
    b = 2;
    // ^ assignment

    printf("%d\n", a);
    printf("%s", nama);
    // ^ menampilkan isi variabel

    return 0;
}
```

Gambar 5. Contoh inisialisasi dan assignment

Variabel juga dapat **diinputkan** oleh user pengguna program dengan menggunakan fungsi `scanf` atau `gets`, fungsi ini berguna untuk meminta inputan ke pengguna yang akan dibaca melalui terminal dan akan menyimpan ke suatu variabel yang mau dituju. Contoh:

```c
#include <stdio.h>
#include <stdlib.h>

typedef char string[128];

int main(){
    int tanggal; //pendeklarasian variabel integer
    string nama; //pendeklarasian variabel string

    printf("Masukan Tanggal : \n");scanf("%d", &tanggal);
    printf("Masukan Nama    : ");fflush(stdin);gets(nama);
    // ^ input data ke dalam variabel

    printf("%d\n", tanggal);
    printf("%s", nama);
    // ^ menampilkan isi variabel

    return 0;
}
```

Gambar 6. Contoh penggunaan input dan output

4. Penamaan

Seperti arti dari kata tersebut, memberi nama. Pada pemrograman digunakan pada berbagai objek yang tersedia dalam program. Nama tersebut hanya dapat diberikan pada **Variabel, Konstanta, Tipe Data Bentukan, Prosedur, Fungsi.**

```c
#include <stdio.h>
#include <stdlib.h>

#define phi 3.14 //konstanta

typedef char string[128]; //tipe data bentukan
typedef int npm; //tipe data bentukan

int main(){
    npm mahasiswa;  //pendeklarasian variabel npm dengan tipe data bentukan
    int tanggal;    //pendeklarasian variabel integer
    string nama;    //pendeklarasian variabel string

    return 0;
}

void printTanggal(int tanggal){ //prosedur
    printf("Tanggal : %d", tanggal);
}

int getTanggal(int tanggal){    //fungsi
    return tanggal;
}
```

Gambar 7. Contoh penamaan

Aturan penulisan nama pada bahasa pemrograman C

1. Case Sensitive (dibedakan berdasarkan *uppercase* dan *lowercase*).
2. Karakter pertama variabel harus dimulai dengan **huruf abjad (a - z) atau underscore (_)**,
3. Karakter yang diperbolehkan dalam sebuah nama adalah **huruf abjad, underscore, dan angka**,
4. Nama tidak boleh dipisahkan dengan spasi,
5. Nama variable harus selain dari keyword (`int, float, main,` dll).

Selain aturan penulisan diatas, hal lain juga perlu diperhatikan adalah kejelasan dan makna suatu nama variabel yang dinamai. Contoh `nilaiDasarPemrograman` lebih mudah dipahami oleh orang lain dibandingkan dengan `nDP`.

Contoh penamaan yang salah:

1. return
2. Nilai Dasar Pemrograman
3. Matematika;Dasar

Contoh penamaan yang salah:

1. matematika_dasar
2. nilaiDasarPemrograman
3. password_10

5. Ekspresi Aritmatika, dan Ekspresi Relasional

Pada bahasa pemrograman C, ada dua jenis operasi yang dapat dilakukan, yang pertama adalah **ekspresi aritmatika** yang merupakan kombinasi dari variabel, konstanta dan operator matematika yang digunakan dan menghasilkan **nilai numerik**. Di sisi lain, **ekspresi relasional** digunakan untuk membandingkan dua nilai dan menghasilkan **nilai logika** (benar atau salah).

Contoh ekspresi aritmatika:

```
1. a = a + b;
2. Jari_jari = 3.14 * r * r;
3. tanggal5 = getTanggal(tanggal) * 5;
```

Contoh ekspresi Relasional:

1. `npm == 220711649;` (Jika npm sama, maka nilai true (1), selain itu akan false (0),
2. `tahun < 2024;` (Jika tahun dibawah 2024, maka nilai true (1), selain itu akan false (0),
3. `angka % 2 == 0;` (Jika angka habis dibagi 2, maka nilai true (1), selain itu akan false (0),

6. Sekuens

Algoritma adalah sekumpulan instruksi yang dijalankan secara berurutan (sekuensial). Secara umum aturan sekuens algoritma adalah sebagai berikut:

1. Tiap instruksi dilaksanakan satu persatu dan biasanya tiap instruksi dibaca dari atas ke bawah dan dari kiri ke kanan,
2. Tiap instruksi dilaksanakan satu kali,

3. Urutan instruksi yang dilaksanakan pemroses sama dengan urutan aksi sebagaimana tertulis dalam algoritma yang ada, dan
4. Akhir dari instruksi terakhir merupakan akhir algoritma.

Contoh Sekuens:

```c
#include <stdio.h>
#include <stdlib.h>

int main(){
    int angka1, angka2;

    angka1 = 1;
    angka2 = 5;

    angka1 = angka1 + angka2;
    printf("\n%d", angka1);

    angka2 = angka1 - 2;
    printf("\n%d", angka2);

    return 0;
}
```

Gambar 8. Contoh sekuens

Nilai `angka1` akan menjadi 6 di saat terjadi operasi aritmatika, dan menampilkan 6, lalu `angka2` akan bernilai 4 dikarenakan operasi tersebut terjadi setelah operasi penambahan pertama terjadi, dan akan menampilkan 4 untuk `angka2`.

GUIDED

Setelah mempelajari modul Tipe Data Penamaan dan Sekuens dengan seksama, teman-teman praktikan diwajibkan membuat guided sebagai salah satu syarat mengikuti praktikum. Terdapat soal yang wajib dikerjakan

Soal:

Buatlah program yang digunakan untuk menghitung dan menampilkan data yang kita input. Data yang akan diinput adalah:

- Nama (string)
- Nama panggilan (string)
- Nilai IPA (float)
- Nilai IPS (float)
- Umur (int)
- Jenis Kelamin (char)

Setelah itu, tampilkan seluruh data diatas ditambah dengan empat perbandingan antara nilai IPA dengan IPS dengan ketentuan:

- Apakah nilai IPA **sama** dengan nilai IPS
- Apakah nilai IPA **tidak sama** dengan nilai IPS
- Apakah nilai IPA **kurang** dari nilai IPS
- Apakah nilai IPA **lebih** dari nilai IPS

(Tugas Guided, Silahkan dikerjakan tanpa kunci jawaban!) Setelah itu, buatlah dua buah variabel tambahan bernama totalNilai dan mean yang merupakan jumlah dan rata-rata dari nilai IPA dan IPS sesuai inputan, tampilkan dua data tersebut dan buatlah perbandingan antara nilai rata-rata dengan nilai IPA dan IPS sesuai dengan ketentuan di bawah:

- Apakah nilai rata-rata **lebih besar sama dengan** nilai IPA
- Apakah nilai rata-rata **lebih kecil sama dengan** nilai IPS

Note soal:

- Tidak harus mencatat seluruh comment yang terdapat pada jawaban di halaman bawah
- Cobalah untuk bereksperimen dengan code guided agar dapat memahami penggunaan format specifier, dan cara input variabel lainnya.

```c
#include <stdio.h>
#include <stdlib.h>
#include <conio.h>
#include <stdbool.h> //Dipakai ketika ingin menggunakan fungsi boolean

/*
Nama
NPM
Kelas
*/

typedef char string[128]; //Membuat tipe data bentukan string berjumlah 128 karakter dari tipe data
char

int main(int argc, char *argv[]) {
    string nama = "Vesha Halvin"; //Inisialisasi variabel ini dengan nama mu (WAJIB!!)
    string namaPanggilan;
    /*
    Tipe data string dapat menyimpan sebanyak jumlah array yang dimasukan di awal saat
    pembuatan tipe data bentukan baru
    */
    float nilaiIPA, nilaiIPS, totalNilai=0, mean=0; //inisialisasikan totalNilai dan mean menjadi 0
    int umur;
    char jenisKelamin; //tipe data char hanya bisa menyimpan 1 buah karakter
    bool boolean; //pendeklarasian variabel bernama boolean dengan tipe data boolean (true/false)

    printf("\t=== [ Input Data ] ===");
    printf("\nNama      : %s", nama);
    printf("\nNama Panggilan    : ");fflush(stdin);gets(namaPanggilan);
    /*
    fflush(stdin) berguna untuk membersihkan input user yang terdapat di penyimpanan
    program juga berguna agar tidak terjadi penumpukan input yang menyebabkan kesalahan
    hasil program

    gets(variabel) berguna sebegai salah satu cara input string, cara ini adalah salah
    satu cara yang sering digunakan
    */
    printf("Nilai IPA   : ");scanf("%f", &nilaiIPA);
    printf("Nilai IPS   : ");scanf("%f", &nilaiIPS);
    printf("Umur        : ");scanf("%d", &umur);
    printf("Jenis Kelamin (L/P) : "); jenisKelamin = getch(); printf("%c", jenisKelamin);
    /*
    getch() berfungsi untuk mengambil inputan langsung dari keyboard tanpa perlu
    menekan enter

    tipe data yang dihasilkan dari getch() adalah char
    */

    printf("\n\n\t=== [ Tampil Data ] ===");
    /*
    disaat menampilkan data, ingat untuk menggunakan format specifier yang tepat agar
    data yang ditampilkan sesuai dengan yang diinginkan
    */
    printf("\nNama      : %s", nama);
    printf("\nNama Panggilan    : %s", namaPanggilan);
    printf("\nNilai IPA : %.2f", nilaiIPA);
    printf("\nNilai IPS : %.2f", nilaiIPS);
    /*
    .2f berguna untuk mengformat agar float yang ditampilkan hanya dibulatkan menjadi dua desimal
    saja
    */
    umur++; //umur bertambah 1
    printf("\nUmur Setelah Increment    : %d", umur);
    umur--; //umur berkurang 1
    printf("\nUmur Setelah Decrement    : %d", umur);
    printf("\nJenis Kelamin (L/P)   : %c", jenisKelamin);
    /*
    umur++ dan umur-- sama dengan "umur = umur + 1" dan "umur = umur - 1"
    ini merupakan post-increment/decrement dimana data akan ditambahkan setelah baris
    tersebut dieksekusi
```

```
    ada pula pre-increment/decrement dimana data akan ditambah atau dikurang di saat
    baris kodetersebut dieksekusi, contoh = ++umur / --umur

    Pelajari dan cobalah bereksperimen dengan pre dan post increment/decrement ini!!!
    */
    printf("\n\n\t=== [ Perbandingan ] ===");
    printf("\n1 : Benar / True");
    printf("\n0 : Salah / False");

    boolean = nilaiIPA == nilaiIPS; //membandingkan apakah nilaiIPA sama dengan nilaiIPS
    printf("\nApakah nilai IPA dan nilai IPS sama? %d", boolean);

    boolean = nilaiIPA != nilaiIPS; //membandingkan apakah nilaiIPA tidak sama dengan nilaiIPS
    printf("\nApakah nilai IPA tidak sama dengan nilai IPS> %d", boolean);

    boolean = nilaiIPA < nilaiIPS; //membandingkan apakah nilaiIPA kurang dari nilaiIPS
    printf("\nApakah nilai IPA kurang dari nilai IPS? %d", boolean);

    boolean = nilaiIPA > nilaiIPS; //membandingkan apakah nilaiIPA lebih dari nilaiIPS
    printf("\nApakah nilai IPA lebih dari nilai IPS? %d", boolean);

    //LANJUTKAN CODE UNTUK TUGAS GUIDED DISINI!

    return 0;
}
```

Screenshot Hasil Program:

```
        === [ Input Data ] ===
Nama            : Vesha Halvin
Nama Panggilan  : Vesha
Nilai IPA       : 70
Nilai IPS       : 90
Umur            : 19
Jenis Kelamin (L/P)     : L

        === [ Tampil Data ] ===
Nama            : Vesha Halvin
Nama Panggilan  : Vesha
Nilai IPA       : 70.00
Nilai IPS       : 90.00
Umur Setelah Increment  : 20
Umur Setelah Decrement  : 19
Jenis Kelamin (L/P)     : L

        === [ Perbandingan ] ===
1 : Benar / True
0 : Salah / False
Apakah nilai IPA dan nilai IPS sama? 0
Apakah nilai IPA tidak sama dengan nilai IPS> 1
Apakah nilai IPA kurang dari nilai IPS? 1
Apakah nilai IPA lebih dari nilai IPS? 0

        === [ Tugas ] ===
Total Nilai     : 160.00
Nilai Rata-rata : 80.00
Apakah nilai Rata-rata lebih besar sama dengan nilai IPA? 1
Apakah nilai Rata-rata lebih kecil sama dengan nilai IPS? 1
--------------------------------
```

Ketentuan Pengerjaan Guided

Buatlah code yang menghasilkan **hasil sama persis dengan screenshot program** di atas **menggunakan aplikasi Dev-C++** kalian, dan harus menggunakan format **Project dari Dev-C++**. **Pastikan untuk merubah komponen yang diminta** untuk diubah dalam code, atau kalian akan mendapatkan **pengurangan nilai guided** ini! Diharapkan praktikan dapat **mengeksplor sendiri dan bereksperimen dengan modul** yang sudah diberikan dan kalian juga dapat membaca comment yang sudah ditulis pada kode diatas, hal ini dapat membantu kalian lebih paham dalam memahami bahasa C dan juga membantu dalam Unguided nantinya.

Ketentuan Penamaan:

- Ekstensi program file harus .c dan bukan .cpp
- Kerjakan soal tadi dalam satu buah folder dan beri nama:

  GD3_X_YYYYY.zip

  Contoh: GD3_A_11649.zip

  

  Keterangan Penamaan:

  X = Kelas

  Y = 5 digit terakhir npm praktikan

  Kesalahan dalam format penamaan dan file akan mengurangi nilai dari guided ini, mohon diperhatikan baik-baik!

  Bila ada pertanyaan bisa PC Teams atau WhatsApp.