

Modul 3

Tipe Data, Penamaan dan Sekuens

Tujuan

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

Tipe Data

Tipe Data adalah pengklasifikasian data berdasarkan jenis data tertentu agar compiler mengetahui bagaimana data tersebut akan digunakan. Dalam penggunaannya, tipe data terbagi menjadi 2 jenis, yaitu **Tipe Data Primitif** dan **Tipe Data Bentukan**.

1. Tipe Data Primitif

Tipe data primitif adalah tipe data dasar yang disediakan secara langsung oleh bahasa pemrograman yang digunakan. Berikut adalah beberapa tipe data primitive yang biasa digunakan.

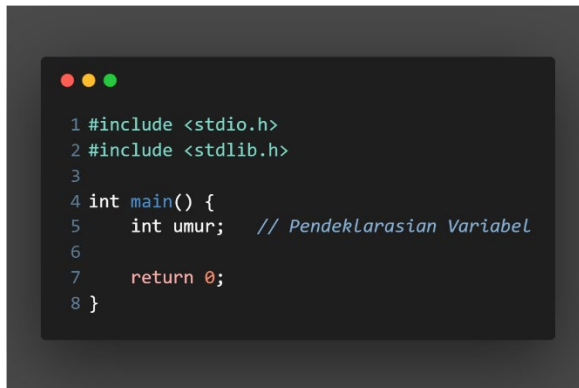
Tipe Data	Keyword	Contoh
Bilangan Bulat – Integer	Int	-100, 0, 2304
Bilangan Real – Desimal	Float	-1.67, 0.009, 10.003
Karakter – Character	Char	'A', 'z', '8', '\$', 'J'
Logika - Boolean	bool	True, false

Selain tipe data diatas, terdapat beberapa jenis tipe data lain sebagai berikut :

Keyword	Format
double	%lf
short int	%hd
unsigned int	%u
long int	%ld, %li
long long int	%lld, %lli
unsigned long int	%lu
unsigned long long int	%llu
signed char	%c
unsigned char	%c
long double	%Lf

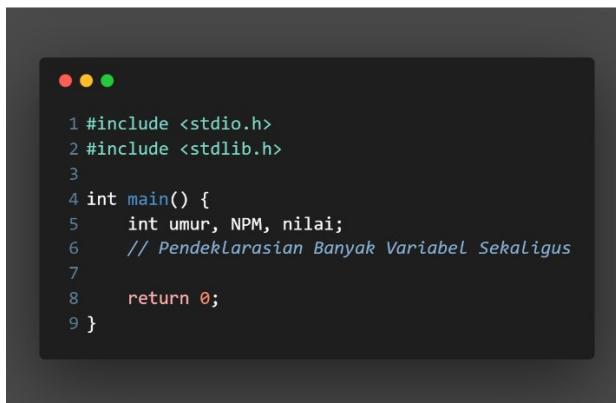
Setelah mengetahui tipe data yang akan digunakan, maka selanjutnya adalah **Deklarasi Variabel**, yaitu proses dalam memperkenalkan/pembuatan sebuah variable sebelum variable tersebut akan digunakan. Pendeklarasian variable dilakukan dengan mengetik tipe data yang akan digunakan terlebih dahulu, kemudian menuliskan nama variable.

Contoh : `int genap;`



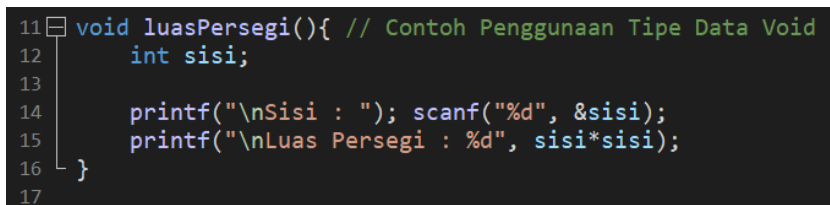
```
1 #include <stdio.h>
2 #include <stdlib.h>
3
4 int main() {
5     int umur;    // Pendeklarasian Variabel
6
7     return 0;
8 }
```

Kita juga bisa mendeklarasikan **banyak variable** sekaligus dalam satu baris dengan tipe data yang sama, yaitu dengan memisahkan nama variable dengan tanda koma. Contoh :



```
1 #include <stdio.h>
2 #include <stdlib.h>
3
4 int main() {
5     int umur, NPM, nilai;
6     // Pendeklarasian Banyak Variabel Sekaligus
7
8     return 0;
9 }
```

Terdapat juga **tipe data void** yang biasa digunakan untuk membuat modul program (kumpulan instruksi selain bagian utama program) atau yang biasa disebut dengan **prosedur**.



```
11 void luasPersegi(){ // Contoh Penggunaan Tipe Data Void
12     int sisi;
13
14     printf("\nSisi : "); scanf("%d", &sisi);
15     printf("\nLuas Persegi : %d", sisi*sisi);
16 }
17
```


2. Tipe Data Bentukan

Tipe data bentukan (*User defined types*) adalah tipe data yang dibuat sendiri sesuai dengan kebutuhan *programmer*. Tipe data bentukan merupakan turunan dari tipe data

primitive atau tipe data bentukan lain. Tipe data baru dibuat dengan keyword `typedef` atau record yang dibuat dengan keyword `struct`.


Contoh penggunaan **typedef** adalah ketika ingin membuat tipe data bentukan bernama “umur” yang berupa integer, maka dapat ditulis dengan `typedef int umur;` Kemudian tipe data tersebut bisa digunakan untuk membuat sebuah variable baru, contoh:

```
umur mahasiswa;
```



```
1 #include <stdio.h>
2 #include <stdlib.h>
3
4 typedef int umur;    // Tipe Data Bentukan
5
6 int main() {
7     umur mahasiswa;
8
9     return 0;
10 }
```

Kita juga bisa membuat tipe data **string** atau kumpulan dari karakter (kalimat) yang dibentuk dari char yang ditambah dengan array. Contoh :



```
1 #include <stdio.h>
2 #include <stdlib.h>
3
4 typedef char string[20];    // Tipe Data Bentukan String
5
6 int main() {
7     string nama;
8
9     return 0;
10 }
```

Penamaan

Penamaan yaitu pemberian identitas pada sebuah objek yang digunakan dalam program. Nama bisa diberikan kepada

1. Variabel : Pemberian nama kepada lokasi memori komputer yang digunakan untuk menyimpan nilai dengan tipe data yang sudah ada dalam program
2. Konstanta : Variabel yang nilainya tidak bisa diubah selama program dijalankan

3. Tipe data bentukan : Turunan dari tipe data dasar ataupun tipe data bentukan lainnya
4. Prosedur : Modul program tanpa nilai balikan
5. Fungsi : Modul program dengan nilai balikan bertipe data dasar ataupun bentukan



```

1 #include <stdio.h>
2 #include <stdlib.h>
3
4 #define Max 5 // Konstanta
5
6 typedef int umur; // Tipe Data Bentukan
7
8 int main() {
9     umur mahasiswa; // Variabel
10
11     return 0;
12 }
13
14 void luasPersegi(){ // Prosedur
15     int sisi; // Tipe Data Primitif
16
17     printf("\nSisi : "); scanf("%d", &sisi);
18     printf("\nLuas Persegi : %d", sisi*sisi);
19 }
20
21 int kelilingPersegi(){ // Fungsi
22     int sisi; // Tipe Data Primitif
23
24     printf("\nSisi : "); scanf("%d", &sisi);
25     printf("\nKeliling Persegi : %d", sisi*4);
26 }

```

Aturan Penulisan Nama pada 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*, angka
4. Nama tidak boleh dipisahkan dengan spasi,
5. Beberapa compiler C membatasi panjang variable maksimal 31 karakter, sehingga sebaiknya tidak menulis nama variable lebih dari 31 karakter.
6. Nama variable harus selain dari keyword (`int`, `float`, `main`, dll)

Selain aturan diatas, hal lain yang perlu diperhatikan adalah pemberian nama suatu variable harus **jelas** dan **bermakna** pada objek yang dinamai. Contoh : `nilaiFisika` akan lebih mudah dipahami oleh orang lain dibandingin dengan `nF`.

Contoh penamaan yang salah :

1. `return`

2. 5anak
3. daftar mahasiswa
4. tumbuhan;langka
5. A

Contoh penamaan yang benar :

1. sisiPersegi
2. luas_segitiga
3. anak_ke_5
4. password

Setiap tipe data akan memiliki sebuah nilai (pada variabel, maupun konstanta) atau mengembalikan sebuah nilai (fungsi). Proses pemberian nilai pada sebuah variabel disebut **assignment**. Jika suatu variabel yang dideklarasikan dilakukan proses assignment maka disebut **inisialisasi**. Nilai tersebut dapat digunakan untuk melakukan perhitungan dengan ekspresi aritmatika sehingga menghasilkan nilai numerik atau menggunakan ekspresi relasional untuk menghasilkan nilai logika.

Contoh assignment :

1. npm = 210700232;
2. nilaiAngka = 90.5;
3. nilaiHuruf = 'A';
4. nama = "Ujang";
5. rata2 = nilai;

Contoh ekspresi aritmatika :

1. a + b;
2. 3.14 * r * r;
3. luasPersegi(sisi) * 2;

Contoh ekspresi relasional dan nilai balikkannya :

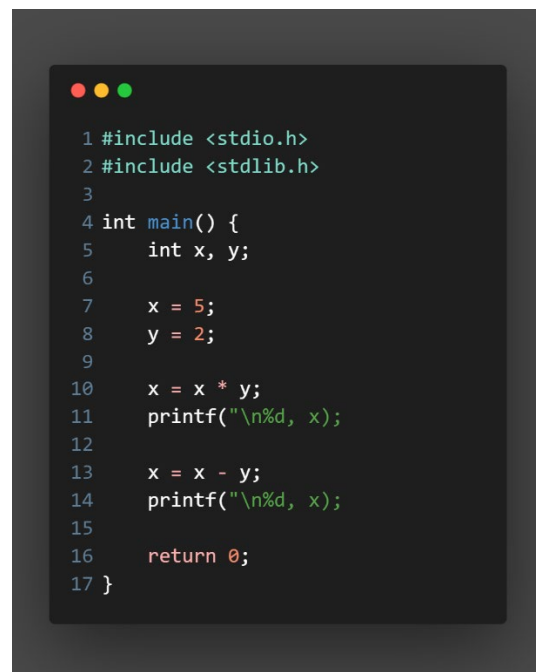
1. umur > 18 (Jika umurnya 20 maka true, tetapi jika umurnya 15 maka false)
2. tahun == 2022 (Jika tahunnya 2022 maka true, tetapi jika tahunnya 2021 maka false)
3. angka % 2 == 1 (Jika angkanya 3 maka true, tetapi jika angkanya 4 maka false)

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 didalam algoritma yang ada
4. Akhir dari instruksi terakhir merupakan akhir algoritma

Contoh sekuens :



```
1 #include <stdio.h>
2 #include <stdlib.h>
3
4 int main() {
5     int x, y;
6
7     x = 5;
8     y = 2;
9
10    x = x * y;
11    printf("\n%d, x);
12
13    x = x - y;
14    printf("\n%d, x);
15
16    return 0;
17 }
```

- Nilai x pada `printf` pertama adalah nilai 10, karena sebelumnya program menjalankan `x = x * y;` terlebih dahulu
- Nilai x pada `printf` kedua adalah 8, karena setelah `printf` pertama program akan lanjut ke `x = x - y;`

**“An error does not become a mistake until
you refuse to correct it.”**

- John F. Kennedy