MEMULAI BAHASA C

OBJEKTIF :

1. Mahasiswa mampu memahami konsep pendahuluan pada C .
2. Mahasiswa mampu memahami konsep program sederhana pada C.
3. Mahasiswa mampu memahami konsep tipe data dan variabel pada C.
4. Mahasiswa mampu memahami konsep aritmatika pada C.
5. Mahasiswa mampu memahami konsep input dan output pada C.

1.1 Pendahuluan C

C merupakan bahasa pemrograman terstruktur yang mendukung pembuatan program sebagai kumpulan prosedur, seperti Pascal dan Cobol. C merupakan bahasa dasar dari berbagai bahasa pemrograman yang lebih modern, seperti C++, C#, Java, dan lain sebagainya.

1.1.1 Sejarah Singkat C

C dikembangkan dari dua bahasa pendahulunya, yaitu BCPL dan B. BCPL dikembangkan pada tahun 1967 oleh Martin Richards sebagai bahasa pemrograman yang digunakan untuk sistem operasi dan compiler. Ken Thompson menerapkan banyak fitur dari BCPL ke dalam bahasa B dan pada tahun 1970, bahasa B digunakan untuk membuat versi awal dari sistem operasi UNIX di Bell Laboratories.
Bahasa pemrograman C dikembangkan dari bahasa pemrograman B oleh Dennies Ritchie di Bell Laboratories dan diimplementasikan pada tahun 1972. Bahasa pemrograman C sangat dikenal sebagai bahasa pengembangan untuk sistem operasi UNIX. Pada zaman sekarang, sistem operasi yang terkenal seperti Windows dan Mac dibuat dengan menggunakan bahasa C dan C++.

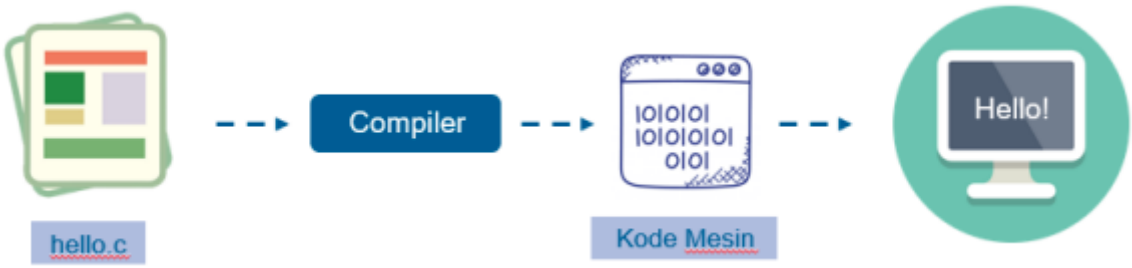
1.1.2 Interpreter dan Compiler

Interpreter adalah suatu jenis penerjemah yang menerjemahkan per baris instruksi untuk setiap saat. Saat program dieksekusi, *interpreter* harus berada dalam memori.

Berikut adalah cara kerja *interpreter*:

Source code akan diterjemahkan baris per baris oleh *interpreter* menjadi kode mesin dan langsung dieksekusi pada saat itu juga. Pada *interpreter*, tidak terdapat file exe.
Compiler merupakan jenis penerjemah yang lain selain *interpreter*.

Berikut adalah cara kerja *compiler*:



Cara kerjanya adalah menerjemahkan seluruh instruksi dalam program terlebih dahulu. Proses ini disebut kompilasi. Proses kompilasi cukup dilakukan sekali saja. Selanjutnya, hasil penerjemahan bisa dijalankan secara langsung tanpa bergantung pada kode sumber maupun *compiler*.

1.1.3 Program Development Environment

Sistem C umumnya terdiri atas beberapa bagian, yaitu:

- Program Development Environment
- Bahasa C
- Standard Library C

Dalam program development environment, program C biasanya akan melalui enam fase untuk dieksekusi, yaitu:

- **Fase 1 : Membuat Program**

  Fase pertama dapat dilakukan dengan menggunakan program *editor*. *Programmer* membuat program di dalam *editor* dan menyimpannya di dalam *disk*. Nama file program C harus menggunakan ekstensi .c.
- **Fase 2 : *Preprocessing program C***

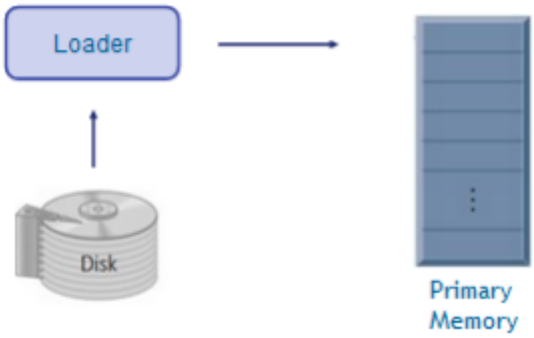
  Pada fase dua, kita memberikan *command* untuk meng-*compile* program. Dalam sistem C, sebuah program *preprocessor* secara otomatis mengeksekusi sebelum fase penerjemahan oleh *compiler* dimulai.
- **Fase 3: *Compiling program C***

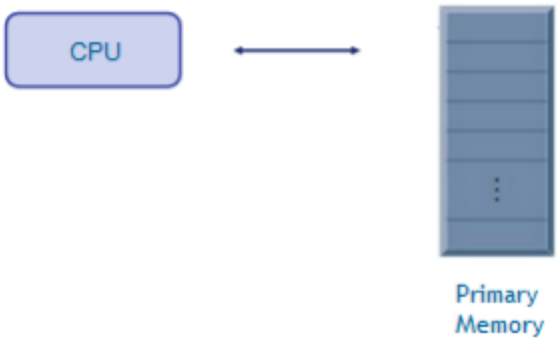
  Pada fase tiga, *compiler* akan menerjemahkan program C ke dalam bahasa mesin dapat juga disebut sebagai kode objek dan menyimpannya dalam *disk*.

- **Fase 4: *Linking***

  Fase empat disebut *linking*. Kode objek yang dihasilkan oleh *compiler* C biasanya memiliki kekosongan karena adanya bagian yang hilang. Sebuah *linker* menghubungkan kode objek dengan kode untuk fungsi yang hilang untuk menghasilkan *executable image*.
- **Fase 5: *Loading***



  Sebelum program dieksekusi, program harus ditempatkan di dalam memori. Proses ini dilakukan oleh *loader*, dimana *loader* mengambil *executable image* dari *disk* dan mentransfer ke memori. Komponen dari *shared library* yang mendukung program juga di-*load*.
- **Fase 6 : *Execution***



  CPU akan menerima masing-masing instruksi dan mengeksekusinya, mungkin menyimpan nilai data baru sebagai *program executes*.

1.1.4 Integrated Development Environment

IDE atau Integrated Development Environment merupakan aplikasi perangkat lunak yang memberikan fasilitas secara lengkap untuk proses pengembangan software. IDE biasanya mempunyai fasilitas dasar seperti:

- *Editor*
- *Compiler*
- *Debugger*

Ada banyak IDE yang dapat kita gunakan seperti:

- IntelliJ
- Eclipse
- Xcode

- Netbeans
- Code Blocks

1.2 Program Sederhana

Dalam program Bahasa C terdapat *statement* program yang berfungsi untuk mengontrol fungsi dan eksekusi program. Dalam *statement* program, terdapat *terminator statement* yang berfungsi sebagai penanda bahwa baris tersebut adalah akhir dari suatu *statement* dengan simbol *semicolon* (;).

Contoh:

```
#include <stdio.h>
int main()
{
    printf("Ini adalah program bahasa C"); // Tanda (;) adalah terminator statement

    return 0;
}
```

Ada beberapa *statement* program yang tidak membutuhkan *terminator statement*, diantaranya:

- Komentar
- Preprocessor Directives (seperti: `#include`)
- Kurung kurawal dalam awal dan akhir *block* program
- Fungsi (seperti: `main(){}`)

Statement program di atas tidak membutuhkan simbol *semicolon* (;) dikarenakan *statement* di atas bukan merupakan *statement* yang dieksekusi ataupun *statement* memanggil fungsi. *Semicolon* hanya digunakan untuk *statement* yang akan dieksekusi.
Fungsi yang dapat ditampilkan pada output adalah fungsi `printf()`. Hal ini dikarenakan fungsi `printf()` mengandung nilai parameter yang ingin dieksekusi untuk ditampilkan. Nilai yang ingin ditampilkan di layar berada dalam karakter *double quote* (“ ”), kecuali karakter *escape* yang digunakan untuk memodifikasi tata letak output.

1.2.1 Penggunaan `printf()`

Fungsi `printf` adalah untuk mengarahkan data ke standard output atau layar. Di bawah ini merupakan penggambaran fungsi `printf`:

```
printf ("Hallo, Selamat pagi %s", nama);
```

Fungsi `printf` akan mencetak *statement* yang ada di dalam simbol (dan), dan memanggil nilai dari suatu variabel atau parameter untuk dicetak ke layar. Jika pada program di atas isi dari variabel nama adalah `“Sasya”`, maka output dari fungsi `printf` di atas adalah `“Hallo, Selamat Pagi Sasya”`.
Agar nilai suatu variabel dapat dicetak ke layar, diperlukan format *identifier* yang berfungsi sebagai tanda bahwa pada bagian tersebut merupakan bagian dimana harus memanggil nilai suatu variabel. Pada program di atas, `%s` merupakan format *identifier*. `%s` digunakan untuk menyisipkan nilai suatu nilai variabel, dimana variabel tersebut atau nilai dari variabel tersebut tergolong ke dalam tipe data `string`. Selain `%s`, ada beberapa format *identifier* lainnya dengan fungsi yang berbeda.

Di bawah ini merupakan jenis-jenis format *identifier*:

Format Identifier	Fungsi
`%d`	Untuk menampilkan bilangan bulat
`%f`	Untuk menampilkan bilangan real bertipe data float
`%lf`	Untuk menampilkan bilangan real bertipe data double
`%c`	Untuk menampilkan suatu karakter
`%s`	Untuk menampilkan suatu string

1.2.2 Komentar

Komentar merupakan suatu bagian yang penting dalam pemrograman. Komentar dapat digunakan dalam bahasa pemrograman apapun. Keberadaan komentar dapat membantu *programmer* dalam melakukan penyusunan program. Salah satu fungsi komentar adalah dapat mengidentifikasi baris program. Komentar dalam bahasa pemrograman ada dua jenis, diantaranya:

- **Komentar Single-Line (`//`)**

  Komentar dengan *double slash* (`//`) merupakan jenis komentar yang hanya dapat mengkomentarkan satu baris kalimat. Biasanya, komentar jenis ini digunakan sebagai tanda atau *sign* sebelum memulai suatu block program.

  Contoh :

  ```c
  #include <stdio.h>

  int main()
  {
      // Mencetak output dengan statement printf
      printf("Hallo, Selamat pagi");

      return 0;
  }
  ```

- **Komentar Multi-Line (`/* */`)**

  Komentar dengan *double slash* dan `double asterisk` merupakan jenis komentar yang dapat mengkomentarkan lebih dari satu baris kalimat atau satu blok program. Biasanya, komentar jenis ini digunakan sebagai mengarsipkan kode program yang tidak terpakai dalam menjalankan suatu program.

  Contoh:

```c
#include <stdio.h>

int main()
{
    /* printf ("Ini baris 1 tidak dicetak");
       printf("Ini baris 2 juga tidak dicetak");*/

    return 0;
}
```

1.2.3 Standard Library

Standard library atau biasa disebut file library merupakan suatu bagian dari program yang berfungsi untuk menyisipkan fungsi yang ada pada file library ke dalam program. Ada tiga jenis file library yang dapat disertakan ke dalam program bahasa C, diantaranya:

- **File header atau library standar dari bahasa C**

  Jenis-jenis file library yang termasuk ke dalam golongan library standar bahasa C adalah seperti , , , , dan lain-lain. Dengan file ini kita tidak perlu menyertakan file lain untuk masuk ke dalam program. Di bawah ini merupakan fungsi dari beberapa file library standar bahasa C:

Jenis Library	Keterangan
`stdio.h`	Digunakan apabila terdapat operasi input/output
`math.h`	Digunakan apabila dibutuhkan fungsi untuk melakukan perhitungan matematika
`stdlib.h`	Digunakan apabila terdapat operasi perbandingan dan konversi
`string.h`	Digunakan apabila terdapat operasi manipulasi `string`
`conio.h`	Digunakan untuk menampilkan hasil antarmuka kepada user

- **File header atau library yang dibuat sendiri**

  Tujuan dari pembuatan file library adalah untuk memisahkan antara fungsi yang satu dengan fungsi lainnya berdasarkan cara kerja masing-masing fungsi. Lalu file library buatan sendiri tersebut disimpan dan dipanggil ke dalam file baru yang mana file tersebut merupakan penggabungan antara seluruh file header yang menampung fungsi masing-masing. Penggunaan file library jenis ini dapat dilakukan secara berulang-ulang pada file lain.
- **File header atau library eksternal**

  File header atau library eksternal mempunyai fungsi yang sama dengan header lain yaitu untuk memasukkan fungsi yang ada pada file library tersebut untuk dimasukkan ke dalam program yang saat ini sedang dibuat. Library eksternal contohnya adalah SDL, GTK, SQLite, OpenGL, dan lain-lain.

1.3 Tipe Data dan Variabel

1.3.1 Tipe Data

Di dalam bahasa pemrograman komputer, data merupakan bagian terpenting dari sebuah program. Bahasa pemrograman komputer membedakan data ke dalam beberapa tipe agar operasi data menjadi efisien dan efektif. Dengan kata lain, tipe data adalah sebuah pengelompokan data untuk memberitahu compiler atau intepreter bagaimana programmer ingin mengolah data tersebut.
Tipe data yang terdapat pada bahasa C yaitu tipe data dasar, tipe data turunan, tipe data pencacahan, dan tipe data void. Pada tipe data dasar (*primitive*) terdiri dari yaitu `integer`, `char`, `float`, `short`, `long`, `double` dan `long double`.

Berikut ini merupakan tabel tipe data dan ukuran memori:

Tipe Data	Ukuran	Keterangan
`char`	1 byte	Menyimpan sebuah karakter
`short`	2 byte	Dengan jangkauan -32768 sampai +32767
`int`	4 byte	Dengan jangkauan -2147483648 sampai +2147483647
`long`	4 byte	Dengan jangkauan -2147483648 sampai +2147483647
`long long`	8 byte	Dengan jangkauan -9,223,372,036,854,775,808 sampai +9,223,372,036,854,775,807
`float`	4 byte	Dengan jangkauan 10^{-38} s.d 10^{38}
`double`	8 byte	Dengan jangkauan 10^{-308} s.d 10^{308}
`long double`	12 byte	Dengan jangkauan 10^{-4932} s.d 10^{4932}
`bool`	1 byte	Menyatakan Boolean dengan kemungkinan nilai berupa *true* jika benar dan *false* jika salah

- ***Character***

  Tipe data `char` digunakan untuk menampung satu karakter. Untuk menuliskan sebuah variabel bertipe data `char`, karakter perlu ditulis di dalam tanda petik tunggal, seperti `'a'`, `'L'`, `'P'`, dan sebagainya.

  Terdapat karakter-karakter khusus yang disebut dengan *escape sequence*, berikut adalah tabel dari *escape sequence*:

Karakter	Keterangan
`\n`	Karakter *new line* (pindah baris)
`\t`	Karakter tab *horizontal*
`\v`	Karakter tab *vertical*
`\`	Karakter `\`

Karakter	Keterangan
`\'`	Karakter `'`
`\"`	Karakter `"`

- **Integer**

  `integer` merupakan sekumpulan bilangan yang merepresentasikan positif dan negatif. Variabel `integer` hanya dapat menampung bilangan bulat. Tipe data `integer` yang sering digunakan yaitu `int` dan menyimpan informasi. Berikut contoh dari bilangan tipe data `integer`:

  - 25
  - 0
  - -10

- **Floating-Point**

  Pada jenis ini, terdapat tipe data `float` dan `double` yang sering digunakan untuk bilangan pecahan atau desimal. Berikut adalah contoh dari bilangan pecahan:

  - 9.4543
  - 3.1415
  - -42.661

  Contoh:

  ```
  float nilai1;
  double nilai2;
  ```

  Tipe data `float` dideklarasikan dengan *keyword* `float` dan tipe data `double` dideklarasikan dengan *keyword* `double`.

- **Boolean**

  Tipe data boolean adalah tipe data yang bernilai *true* jika kondisi benar, dan *false* jika kondisi salah. Tipe data boolean dideklarasikan dengan *keyword bool* dan dapat digunakan jika kita menggunakan standard library .

1.3.2 Operator `sizeof`

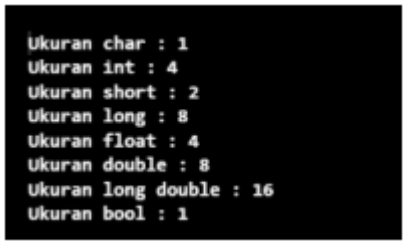
`sizeof` adalah operator yang terdapat pada C dan C++ yang digunakan untuk menghitung ukuran memori yang diperlukan suatu tipe data. Operator `sizeof` dapat digunakan pada tipe data apapun termasuk tipe data primitif.

Berikut ini adalah contoh penggunaan `sizeof`:

```
#include <stdio.h>
#include <stdbool.h>

int main()
{
    printf("Ukuran char : %d\n", sizeof(char));
```

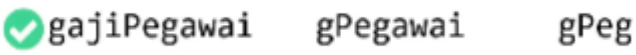
```
    printf("Ukuran int : %d\n", sizeof(int));
    printf("Ukuran short : %d\n", sizeof(short));
    printf("Ukuran long : %d\n", sizeof(long));
    printf("Ukuran float : %d\n", sizeof(float));
    printf("Ukuran double : %d\n", sizeof(double));
    printf("Ukuran long double : %d\n", sizeof(long double));
    printf("Ukuran bool : %d\n", sizeof(bool));

    return 0;
}
```

Jika program di atas dijalankan, maka akan menghasilkan output seperti berikut ini:



1.3.3 *Identifier*

Identifier atau pengenal adalah suatu nama yang biasa dipakai dalam pemrograman untuk menyatakan variabel, tipe data, dan lain sebagainya. Pemberian nama *identifier* sebaiknya menggunakan kata yang berarti dan mudah dibaca. Sebagai contoh:



C merupakan *case sensitive*, huruf kecil dan kapital tidaklah sama. Beberapa pemrograman menggunakan huruf kapital untuk menyatakan awal kata, khusus untuk kata kedua dan seterusnya yang dengan punuk unta. Berikut adalah contohnya:

1.3.4 *Keyword*

Keyword merupakan kata-kata yang telah ditentukan sebelumnya, yang digunakan dalam pemrograman yang memiliki arti khusus bagi *compiler*. Kita tidak boleh menggunakan *keyword* untuk penamaan variabel. Bahasa C standar ANSI hanya mendefinisikan 32 *keyword* dan Turbo C menambahkannya 7 keyword. Berikut adalah tabel *keyword*:

Keywords				
auto	do	goto	signed	unsigned
break	double	if	sizeof	void
case	else	int	static	volatile
char	enum	long	struct	while
const	extern	register	switch	
continue	float	return	typedef	
default	for	short	union	

1.3.5 Penamaan Variabel

Untuk menunjukkan area penyimpanan, setiap variabel harus diberi nama yang unik. Kita menamakan variabel mengikuti ketentuan penamaan variabel. Nama variabel hanyalah representasi simbolis dari lokasi memori.

Nama variabel hanya boleh memiliki:

- Huruf (huruf besar dan kecil), angka, dan garis bawah.
- Huruf pertama variabel harus berupa huruf atau garis bawah (tidak dimulai dengan angka).
- Tidak menggunakan *keyword*.

Perhatikan code berikut:

```
int for = 5;
unsigned int nilai;
```

Pada baris pertama, `int` merupakan tipe data dan `for` adalah nama variabel yang merupakan *keyword* untuk *statement* perulangan. Sedangkan pada baris kedua, `unsigned` merupakan *keyword* yang digunakan sebagai penanda bahwa nilai dari variabel `int` tidak dapat bernilai negatif. Sehingga, kode pada baris kedua yang benar. Hal tersebut dikarenakan pada kode baris pertama, menggunakan *keyword* `for` sebagai nama variabel yang. Sedangkan, nama variabel tidak boleh menggunakan *keyword*.

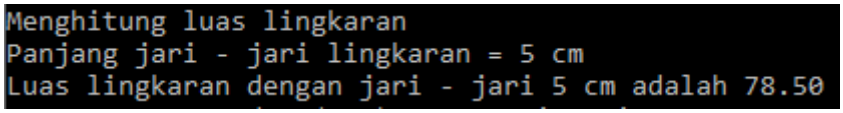
1.3.6 Variabel dan Konstanta

Variabel digunakan untuk menyimpan suatu nilai. Nilai yang ditugaskan dapat diubah selama program dieksekusi. Konstanta atau literal menyatakan nilai yang tetap, berbeda dengan variabel yang nilainya dapat berubah-ubah. Konstanta dapat digunakan dengan *keyword* `const`.

Berikut ini contoh program variabel dan konstanta:

```
#include <stdio.h>

int main(void) {

    const float pi = 3.14;
    int jejari = 5;
    float luas_lingkaran;

    printf("Menghitung luas lingkaran");
    printf("\nPanjang jari - jari lingkaran = %d cm", jejari);

    luas_lingkaran = pi * jejari * jejari;

        printf("\nLuas lingkaran dengan jari - jari %d cm adalah %.2f", jejari,
luas_lingkaran);

    return 0;
}
```

Jika program di atas dijalankan, maka akan menghasilkan output seperti berikut ini:



1.3.7 Deklarasi Variabel

Variabel adalah suatu pengenal yang digunakan sebagai tempat (`storage area`) untuk menampung data. Variabel harus dideklarasikan terlebih dahulu supaya suatu program dapat menggunakan data atau nilai yang tersimpan di dalamnya.

Deklarasi variabel dapat dilakukan dengan menggunakan tipe data dan nama variabel.

Contoh:

```
int nilaiUTS;
int nilaiUAS;
```

`int` merupakan tipe data dan diikuti dengan nama variabelnya yaitu `nilaiUTS` dan `nilaiUAS`. Kita juga bisa mendeklarasikan variabel dengan satu baris saja jika variabelnya memiliki tipe data yang sama, seperti contoh berikut ini:

```
int nilaiUTS,nilaiUAS;
```

1.3.8 Inisialisasi Variabel

Variabel merupakan nama yang diasosiasikan ke suatu nilai. Nilai pada variabel dapat mengalami perubahan (*mutable*). Nilai yang ditugaskan variabel dapat diinisialisasi setelah deklarasi ataupun saat deklarasi variabel.

Contoh:

```
int nilaiUTS;
nilaiUTS = 85;
```

atau

```
int nilaiUTS = 85;
```

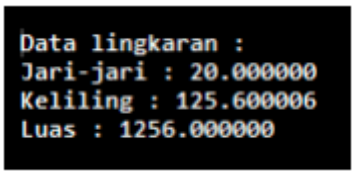
Kita dapat menginisialisasi variabel `Nilai_UTS` dengan menggunakan operator assignment atau penugasan, yaitu tanda sama dengan (`=`). Kemudian, `nilaiUTS` akan ditugaskan dengan nilai 85.

Berikut ini contoh program tipe data dan variabel:

```
#include <stdio.h>

const float PI = 3.14;

int main()
{
```

```
    float radius, keliling, luas;

    radius = 20;

    keliling = 2 * PI * radius;
    luas = PI * radius * radius;

    printf("Data lingkaran :\n");
    printf("Jari-jari : %f\n", radius);
    printf("Keliling : %f\n", keliling);
    printf("Luas : %f\n", luas);

    return 0;
}
```

Jika program di atas dijalankan, maka akan menghasilkan output seperti berikut ini:



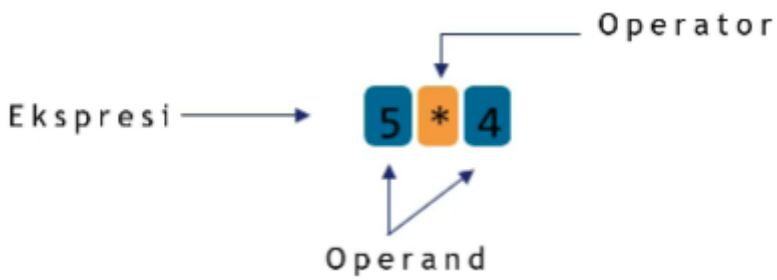
1.4 Aritmatika

1.4.1 Operator

Operator merupakan simbol yang biasa dilibatkan dalam program untuk melakukan sesuatu operasi atau manipulasi, sedangkan nilai yang dikenakan operasi disebut sebagai **operand**. Pada bahasa C, operator memiliki tiga macam sifat, yaitu:

Sifat Operator	Keterangan	Contoh
Unary	Operator ini hanya melibatkan satu *operand*	-100
Binary	Operator yang melibatkan dua *operand*	1 + 3
Ternary	Operator yang melibatkan tiga *operand*	(a > b) ? a : b

Kombinasi operator dan operand yang menghasilkan suatu nilai disebut sebagai **ekspresi**.



Untuk menugaskan ekspresi di atas, diperlukan operator assignment yang memiliki simbol (`=`) dan berfungsi untuk menugaskan suatu nilai kepada variabel. Sebagai contoh:

1.4.2 Operator Aritmatika

Operator aritmatika merupakan operator yang digunakan untuk mengoperasikan suatu bilangan untuk melakukan perhitungan matematika. Operator aritmatika terbagi menjadi dua, yaitu: operator *unary* dan *binary*.

Berikut adalah tabel operator *unary*:

Operator	Keterangan	Contoh
`+`	Tanda plus	+100
`-`	Tanda minus	-123

Berikut adalah tabel operator *binary*:

Operator	Keterangan	Contoh
`*`	Perkalian	7 * 2 = 14
`/`	Pembagian	6 / 2 = 3
`%`	Modulus	7 % 2 = 1
`+`	Penjumlahan	7 + 2 = 9
`-`	Pengurangan	7 - 2 = 5

1.4.3 Operator *Precedence* Aritmatika

Setiap operator mempunyai tingkatan didahulukannya perhitungan dari operator yang disebut operator *precedence*.

Operator	Prioritas
`*` , `/` , dan `%`	Tertinggi
`+` dan `-`	Terendah

Berikut ini adalah ekspresi dari penggunaan operator *precedence*:

```
x = 7 + 9 / 3 - 2;
```

Pada ekspresi di atas, operator `/` dievaluasi terlebih dahulu karena mempunyai *precendence* paling tinggi. Sehingga, ekspresinya akan menjadi seperti berikut:

```
x = 7 + 3 - 2
```

Operator `+` dan `-` mempunyai `precedence` yang sama, sehingga ekspresi dievaluasi dari kiri ke kanan. Maka, ekspresinya akan menjadi seperti berikut:

```
x = 10 - 2
```

Selanjutnya, kita dapat langsung menghitung ekspresi di atas. Sehingga, hasilnya menjadi seperti berikut:

```
`x = 8`
```

Hasil dari operasi tersebut akan ditugaskan kepada variabel x dengan menggunakan operator (`=`).
Kita dapat mengutamakan bagian dari ekspresi dievaluasi terlebih dahulu sebelum bagian lain, dengan menuliskan bagian tersebut dalam tanda kurung. Seperti contoh pada ekspresi berikut ini:

```
(a + b) / 4
```

Pada ekspresi di atas, penjumlahan `a + b` dievaluasi terlebih dahulu karena berada di dalam kurung, lalu hasilnya dibagi dengan 4.

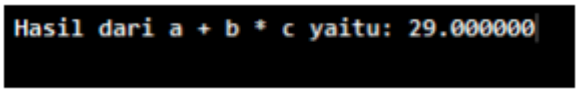
Tabel berikut adalah contoh penggunaan tanda kurung pada ekspresi:

Ekspresi	**Nilai**
(5 + 2) *4	28
10 / (5 - 3)	5
8 + 12 * (6 - 2)	56
(4 + 17) % 2 -1	0

Berikut ini contoh program aritmatika:

```c
#include <stdio.h>

int main()
{
  float a, b, c, hasil;

  a = 5;
  b = 6;
  c = 4;

  hasil = a + b * c;

  printf("Hasil dari a + b * c yaitu: %f", hasil);

  return 0;
}
```

Jika program di atas dijalankan, maka akan menghasilkan output seperti berikut ini:



1.5 Input dan Output

1.5.1 Menggunakan Fungsi `printf()`

Untuk menampilkan output data terformat, dapat menggunakan fungsi `printf()`. Setiap pemanggilan `printf()` berisi *format control string* yang mendeskripsikan format output.
Berikut adalah bentuk umum dari fungsi `printf()`:

```
printf(format control string, argumen lain);
```

Pada *format control string* berisi kode format, *flags*, *field widths*, *precisions*, dan *literal characters* dan untuk argumen lain bersifat *optional*, dapat ditambahkan pada program atau tidak.

Pada *format control string* terdapat:

- Kode Format, terdiri dari dua karakter yaitu persen (`%`) dan karakter spesial yang memberi tahu program untuk *convert data*. Sebagai contoh: `%f`.
- *Flags*, dapat digunakan sebagai suplemen atau tambahan dari output yang dihasilkan. Seperti menambahkan tanda minus atau plus. Sebagai contoh: `%+d`.
- *Field widths*, berfungsi menentukan ukuran tepat field dari data yang akan dicetak. Sebagai contoh: `%4d`
- *Precision*, biasanya digunakan untuk tipe data *floating point* untuk menampilkan berapa angka di belakang koma yang ingin kita tampilkan. Precision memiliki arti yang berbeda di setiap tipe data. Sebagai contoh: `%.4f`
- *Literal characters*, Biasanya ditandai dengan tanda kutip. Berguna untuk menampilkan karakter yang ingin dicetak. Sebagai contoh: `"Hello World"`

1.5.2 Menggunakan Fungsi scanf()

Untuk melakukan input data terformat, dapat menggunakan fungsi `scanf()`. Setiap *statement* `scanf()` berisi kode format yang mendeskripsikan format data untuk di-input.

Berikut adalah bentuk umum dari fungsi `scanf()`:

```
scanf(kode_format, &nama_variabel)
```

Pada `kode_format` mengindikasikan tipe data yang harus di-input oleh user dan `nama_variabel` yang akan di-input dalam penulisan argumen ini, dimulai dengan karakter `&` yang disebut juga *address operator*.

1.5.3 Kode Format

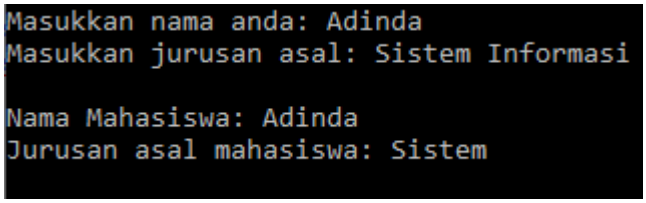
Berikut adalah tabel dari kode format beserta fungsinya yang dapat digunakan di dalam fungsi `printf()` dan fungsi `scanf()`:

Kode format	Fungsi
`%c`	Membaca/menampilkan sebuah karakter
`%s`	Membaca/menampilkan nilai string
`%d`	Membaca/menampilkan nilai desimal integer
`%i`	Membaca/menampilkan nilai desimal integer
`%u`	Menampilkan nilai desimal integer tidak bertanda (*unsigned*)
`%x`	Membaca/menampilkan nilai heksa desimal integer
`%o`	Membaca/menampilkan oktal integer
`%f`	Membaca/menampilkan nilai pecahan
`%e`	Membaca/menampilkan nilai pecahan dalam notasi *specific*
`%g`	Sebagai pengganti `%f` atau `%e` tergantung mana yang terpendek
`%h`	Membaca nilai `short` integer desimal
`%p`	Menampilkan suatu alamat memori untuk pointer

Berikut ini contoh program input dan output:

```c
#include <stdio.h>

int main() {

    char nama[20], jurusan[50];

    printf("Masukkan nama anda: ");
    scanf("%s", &nama);

    printf("Masukkan jurusan asal: ");
    scanf("%s", &jurusan);
    printf("\n");

    printf("Nama Mahasiswa: %s\n", nama);
    printf("Jurusan asal mahasiswa: %s\n", jurusan);

    return 0;
}
```

Jika program di atas dijalankan, maka akan menghasilkan output seperti berikut ini:



REFERENSI

[1] Deitel, Paul dan Harvey Deitel. 2016. *C How to Program with an introduction to C++*. Pearson Education

[2] Vine, Michael. 2008. *C Programming for the Absolute Beginner*. Thomson Course Technology PTR

[3] Kadir, Abdul. 2015 . *From Zero to a Pro Pemrograman C*. ANDI