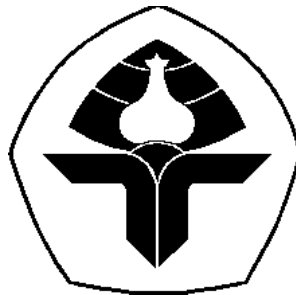


BUKU AJAR

PEMROGRAMAN DASAR

MKK-13206



Disusun Oleh
Putu Indah Ciptayani, S.Kom. M.Cs

PROGRAM STUDI MANAJEMEN INFORMATIKA
JURUSAN TEKNIK ELEKTRO
POLITEKNIK NEGERI BALI
2016

DAFTAR ISI

DAFTAR ISI.....	i
DAFTAR GAMBAR	v
DAFTAR TABEL.....	vi
DAFTAR LAMPIRAN.....	vii
PENGANTAR	viii
PRAKATA.....	ix
TUJUAN PEMBELAJARAN.....	2
BAB I BAHASA PEMROGRAMAN C.....	4
1.1 Sejarah Bahasa C.....	4
1.2 Lingkungan Pengembangan Bahasa C.....	5
1.3 Kelebihan Bahasa C	7
1.4 Penutup.....	7
BAB II DASAR-DASAR BAHASA C	9
2.1 Bahasa C	9
2.2 Contoh Program dalam Bahasa C	10
2.3 Fungsi <i>main()</i>	11
2.4 Fungsi <i>printf()</i>	11
2.5 Preprocessor	12
2.6 Memberikan Komentar pada Program	13
2.7 Karakter Khusus.....	14
2.8 Identifier/Pengenal	16
2.8 Kata Kunci	16
2.11 Contoh Soal.....	17
2.10 Penutup.....	18
BAB III TIPE DATA DAN OPERATOR	21

3.1 Tipe Data Dalam Bahasa C	21
3.2 Variabel.....	23
3.3 Konstanta	24
3.4 Perintah Masukan dengan fungsi <code>scanf ()</code>	25
3.5 Operator.....	27
3.5.2 Operator Penurunan dan Penaikan	29
3.6 Contoh Soal.....	34
3.7 Penutup.....	38
BAB IV PENGAMBILAN KEPUTUSAN	43
4.1 Percabangan	43
4.2 Perintah <code>if</code>	44
4.3 Percabangan bersarang.....	52
4.4 Contoh Soal.....	53
4.5 Penutup.....	58
BAB V PERULANGAN PROSES.....	62
5.1 Perulangan Proses	63
5.2 Perulangan <code>for</code>	63
5.3 Perulangan <code>while</code>	65
5.4 Perulangan <code>do-while</code>	67
5.5 Perintah <code>break</code>	68
5.6 Perintah <code>continue</code>	70
5.7 Perintah <code>goto</code>	72
5.8 Perulangan Bersarang.....	73
5.9 Perulangan Tak Berhingga.....	74
5.10 Contoh Soal.....	75
5.11 Penutup.....	78

BAB VI FUNGSI.....	82
6.1 Fungsi.....	83
6.2 Deklarasi dan Pemanggilan Fungsi.....	85
6.3 Argumen Fungsi.....	87
6.4 Cakupan Variabel (scope variable)	89
6.5 Fungsi Rekursi	91
6.6 Contoh Soal.....	92
6.7 Penutup.....	98
BAB VII ARRAY	102
7.1 Definisi Array.....	102
7.2 Array Satu Dimensi.....	103
7.3 Array Dua Dimensi	106
7.4 Melewatkan Array pada Fungsi	108
7.5 Mengembalikan Array pada Fungsi	109
7.6 Contoh Soal.....	110
7.7 Penutup.....	114
BAB VIII STRING	118
8.1 String	118
8.2 Fungsi String	120
8.3 Contoh Soal.....	121
8.4 Penutup.....	122
BAB IX POINTER	125
9.1 Pointer	125
9.2 Penggunaan Pointer.....	127
9.3 Pointer null	128
9.4 Aritmatik Pointer.....	129

9.5 Perbandingan Pointer	130
9.6 Contoh Soal.....	131
9.7 Penutup.....	132
BAB X STRUKTUR	134
10.1 Struktur.....	134
10.2 Pendefinisian Struktur.....	135
10.3 Pengiriman Struktur Sebagai Argumen	136
10.4 Pointer dari Struktur.....	137
10.5 Array dari Struktur	138
10.5 Contoh Soal.....	140
10.6 Penutup.....	146
TES SUMATIF	148
DAFTAR PUSTAKA	153
LAMPIRAN.....	154
INDEX	157
GLOSSARY.....	158
CURRICULUM VITAE PENGARANG	161

DAFTAR GAMBAR

Gambar 1.1 Lingkungan Pengembangan Bahasa C	6
Gambar 2.1 Kata Kunci dalam Bahasa C (Tutorials Poin, 2014)	17
Gambar 2.2. Tampilan Output Soal Latihan 1	17
Gambar 2.3. Tampilan Output Program Kasir Sederhana	20
Gambar 3.1. String Kontrol untuk Perintah <code>printf()</code> dan <code>scanf()</code>	26
Gambar 3.2. Contoh Operator Pemendekan	34
Gambar 4.1a. Diagram alir if tanpa else (Tutorials Point, 2014).....	44
Gambar 4.1b. Diagram alir if-else (Tutorials Point, 2014).....	44
Gambar 4.2. Diagram Alir Switch-Case (Tutorials Point, 2014).....	50
Gambar 5.1. Diagram Alir For (Tutorials Point, 2014)	64
Gambar 5.2. Diagram Alir Perulangan While (Tutorials Point, 2014)	66
Gambar 5.3. Diagram Alir Perulangan Do-While (Tutorials Point, 2014).....	68
Gambar 5.4. Diagram Alir Perulangan dengan Break	69
Gambar 5.5. Diagram Alir Perulangan dengan Continue	71
Gambar 5.6. Diagram Alir Goto	72
Gambar 5.7. Keluaran Perulangan Bersarang	74
Gambar 5.8. Program Perulangan	76
Gambar 5.9. Program Bintang	80
Gambar 7.1. Keluaran Program Selisih	111
Gambar 9.1. Keluaran Program Cetak Alamat	126
Gambar 9.3. Ilustasi Pointer	128
Gambar 9.4. Keluaran dari Program Aritmatik Pointer	130

DAFTAR TABEL

Tabel 3.1. Tipe Data dalam Bahasa C (Parlente, 2003)	22
Tabel 3.2. Daftar Operator Relasional	31
Tabel 3.3. Daftar Operator Logika	32
Tabel 3.4. Daftar Operator Bitwise	33
Tabel 3.5. Prioritas Operator	34
Tabel 5.1. Perbedaan Sintaks Perulangan	63
Tabel 6.1. Jenis Pemanggilan Argumen.....	87
Tabel 6.2. Jenis Variabel Berdasarkan Cakupan.....	89
Tabel 6.3. Nilai inisial variabel globa	91
Tabel 8.1. Daftar Fungsi String Umum.....	120

DAFTAR LAMPIRAN

Lampiran 1 : Simbol Flowchart	154
Lampiran 2 : Tabel ASCII.....	155

PENGANTAR

PRAKATA

Puji syukur penulis panjatkan kepada Tuhan Yang Maha Esa, karena berkat rahmat beliaulah buku ajar ini dapat terselesaikan dengan tepat waktu. Buku ajar ini dirancang dalam rangka membantu dalam pengajaran matakuliah Pemrograman Dasar di kampus. Buku ini diperuntukkan bagi mahasiswa Ilmu Komputer yang sedang mempelajari Bahasa Pemrograman Dasar, khususnya mahasiswa D3 Manajemen Informatika.

Dalam setiap bab-nya akan diulas mengenai tujuan pembelajaran, teori terkait disertai dengan contoh, contoh soal yang berisi pembahasan dan penutup yang terdiri atas rangkuman, soal latihan dan umpan balik. Contoh soal dirancang sesuai dengan kasus pada kehidupan sehari-hari agar lebih mudah dipahami oleh pembaca. Buku ini merupakan tingkatan dasar dalam mempelajari Pemrograman C. Pada sebagian besar bab, pembaca akan diajak berpikir mengenai topik yang sedang dibacanya, dengan demikian diharapkan ada rasa ingin tahu dari pembaca dan membangkitkan minatnya untuk mencoba apa yang dibacanya sampai akhirnya menemukan jawaban.

Adapun pembahasan dalam buku ini dituangkan ke dalam 10 bab sebagai berikut :

- Bab 1 memperkenalkan Bahasa C kepada pembaca, terutama pembaca yang belum pernah mengenal Bahasa C. Selain itu, bagi pembaca yang sama sekali belum mengenal Bahasa C, bab ini juga memberikan penjelasan mengenai fase yang terjadi mulai dari penulisan kode program hingga eksekusi program.
- Bab 2 membahas mengenai struktur umum dan sifat pemrograman C. Pada bab ini dibahas sintaks dasar dari C dan pembaca akan lebih banyak diajak untuk mengenali mana program yang benar dan mana yang salah, serta bagaimana memperbaiki kesalahan program.
- Bab 3 membahas berbagai tipe data dan operator yang digunakan dalam Bahasa C. Setiap program tidak bisa terlepas dari data. Dalam bab ini, pembaca diarahkan untuk memilih tipe data yang paling sesuai untuk menyimpan suatu data yang diberikan.

- Bab 4 membahas percabangan dan berbagai sintaks yang digunakan dalam menyelesaikan percabangan. Pembaca diajak untuk mengenali kasus yang merupakan percabangan dan memilih jenis percabangan yang sesuai dengan kasus yang diberikan.
- Bab 5 membahas perulangan proses dan jenis-jenis perulangan yang ada. Selain jenis perulangan, juga dibahas beberapa perintah yang kadangkala ditemui di dalam sebuah kasus perulangan.
- Bab 6 mengajak pembaca untuk mengenali lebih dalam mengenai fungsi. Meskipun istilah ini sudah diperkenalkan pada bab 2, namun itu hanya sebagai pengantar saja, sedangkan bab 6 akan membahas fungsi secara mendetail. Dengan menyelesaikan bab 6, maka pembaca akan mampu menggunakan fungsi untuk dapat membuat program lebih mudah dibaca dan terstruktur.
- Bab 7 membahas struktur data array. Pada bab ini akan dibahas mulai dari deklarasi array, pengaksesan array, sampai dengan menjadikan array sebagai parameter dalam sebuah fungsi. Array yang dibahas dalam bab ini adalah array satu dan dua dimensi.
- Bab 8 membahas mengenai string. Pada pemrograman C, tipe data string berbeda dengan pada pemrograman lainnya. Dalam pemrograman C tidak ada tipe khusus yang membahas string. Dengan demikian pada bab ini dibahas mengenai bagaimana membentuk, mengakses, menginput dan memanipulasi string dengan fungsi yang telah disediakan pada pemrograman C.
- Bab 9 membahas mengenai pointer. Pembahasan pointer pada bab ini tidak begitu mendalam, karena targetnya adalah pembaca memahami pointer sederhana dan fungsinya secara umum.
- Bab 10 membahas mengenai struktur data buatan pengguna yang bernama struktur. Bab ini akan mengajak pembaca akan diajak untuk memahami bagaimana mendeklarasikan dan mengakses sebuah struktur, memadukan struktur dengan array, fungsi dan pointer.

Diharapkan buku ini akan memberikan kontribusi bagi pembaca yang sedang mempelajari Bahasa C. Buku ini cocok bagi pembaca tingkat dasar yang belum pernah mempelajari Bahasa C atau sudah pernah namun belum begitu

mendalami Bahasa C. Bagi pembaca yang tingkat menengah buku dapat membantu pembelajaran melalui beraneka latihan soal yang disediakan.

Penulis sadar bahwa masih banyak kekurangan dalam buku ajar ini, oleh sebab itu penulis meminta maaf jika masih terdapat kesalahan dalam buku ajar ini. Penulis akan sangat terbuka bagi kritik dan masukan yang berguna dalam pengembangan dan perbaikan buku ajar ini.

Badung, September 2016

Penulis

TUJUAN PEMBELAJARAN

1. Tentang Mata Kuliah

- Nama : Pemrograman Dasar
- Kode : MKK-13206
- Semester : satu
- SKS : 4

2. Manfaat Mata Kuliah

Manfaat yang diperoleh mahasiswa setelah menempuh mata kuliah ini adalah mahasiswa diharapkan mampu menganalisis permasalahan dalam dunia nyata, dan mengimplementasikannya ke dalam bahasa pemrograman C.

3. Deskripsi Mata Kuliah

Mata kuliah Pemrograman Dasar mempelajari tentang Pengenalan Pemrograman C, Struktur Umum Pemrograman C, Instruksi Masukan Keluaran, Variabel, Tipe Data, Operator, Seleksi, Perulangan, Array, Sub Program, String, Pointer, dan Struktur. Mata kuliah ini merupakan mata kuliah teori dan praktek yang diimplementasikan di laboratorium menggunakan bahasa pemrograman C.

4. Standar Kompetensi dan Kompetensi Dasar

Standar Kompetensi

Mahasiswa diharapkan mampu menerjemahkan permasalahan dalam dunia nyata ke dalam algoritma dan mengimplementasikannya ke dalam Bahasa C setelah mempelajari mata kuliah ini.

Kompetensi Dasar

- 1) Mampu mengenali pemrograman C dan sejarahnya
- 2) Mampu memahami konsep pemrograman C
- 3) Mampu memahami struktur umum pemrograman C
- 4) Mampu menggunakan instruksi masukan dan keluaran
- 5) Mampu menggunakan variabel
- 6) Mampu menggunakan tipe data sebagai deklarasi

- 7) Mampu menggunakan operator dalam C
- 8) Mampu menggunakan pemograman seleksi dalam menyelesaikan suatu kasus
- 9) Mampu menggunakan statemen For, While dan Do..While untuk melakukan proses berulang serta mengetahui batasan awal, batasan akhir dan penambahan serta perulangan tersarang
- 10) Mampu menggunakan fungsi dan dapat memahami perbedaan antara peubah lokal dan peubah global
- 11) Mampu menggunakan variabel berindeks (array)
- 12) Mampu menggunakan String dan memanipulasinya dalam bahasa C
- 13) Mampu memahami pointer dalam bahasa C
- 14) Mampu menggunakan struktur dalam bahasa C

BAB I

BAHASA PEMROGRAMAN C

Standar Kompetensi

Mahasiswa diharapkan mampu menerjemahkan permasalahan dalam dunia nyata ke dalam algoritma dan mengimplementasikannya ke dalam Bahasa C setelah mempelajari mata kuliah ini.

Kompetensi Dasar

Setelah mempelajari materi ini, mahasiswa diharapkan mampu menjelaskan konsep dasar dan sejarah pemrograman C.

Indikator Pencapaian Kompetensi

Adapun indikator hasil belajar adalah sebagai berikut :

1. Mahasiswa mampu menjelaskan sejarah Bahasa C
2. Mahasiswa mampu menjelaskan lingkungan pengembangan Bahasa C
3. Mahasiswa mampu menjelaskan kelebihan Bahasa C

MATERI

1.1 Sejarah Bahasa C

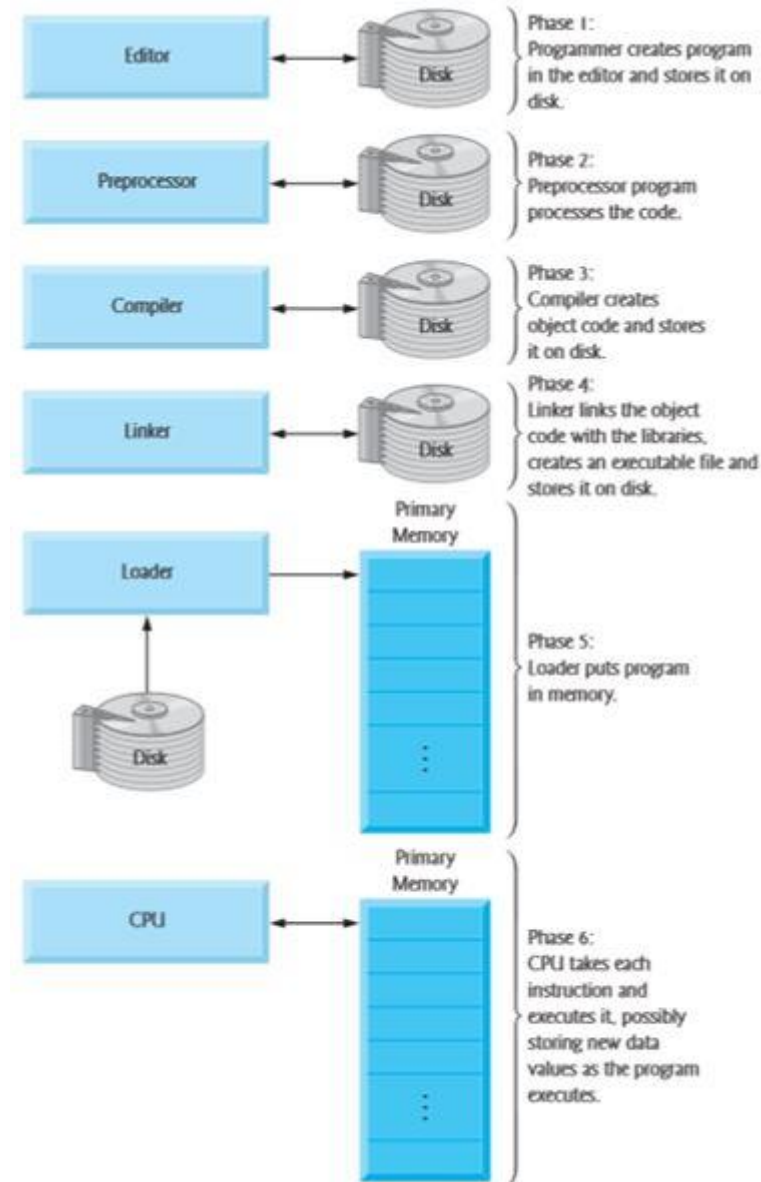
Bahasa C merupakan perkembangan dari dua Bahasa sebelumnya yaitu BCPL dan B. BCPL dikembangkan pada tahun 1967 oleh Martin Richards untuk membuat sistem operasi dan kompiler. Ken Thompson membuat banyak fitur dalam Bahasa B setelah membandingkannya dengan BCPL, dan pada tahun 1970 dia menggunakan Bahasa B untuk membuat sistem operasi UNIX di Bell Laboratories. Bahasa C diciptakan oleh Dennis Ritchie di Bell Laboratories dan diimplementasikan pada komputer DEC PDP-11 di tahun 1972. Bahasa C menggunakan banyak konsep penting dari BCPL dan B di samping menambahkan tipe data dan beberapa fitur luar biasa lainnya. Saat ini C dikenal sebagai bahasa

pengembang sistem operasi UNIX. Saat ini terdapat banyak sistem operasi yang dikembangkan dengan Bahasa C dan/atau C++. Bahasa C yang dikembangkan pada akhir tahun 1970-an, saat ini dikenal sebagai “*traditional C*.” Publikasi buku Kernighan dan Ritchie pada tahun 1978 dengan judul “The C Programming Language”, mendapatkan banyak perhatian. Buku ini menjadi buku komputer paling laris sepanjang sejarah. Ekspansi yang cepat dari Bahasa C dalam komputer yang berbeda, membuat terdapat banyak versi Bahasa C namun tidak kompatibel pada semua platform. Akhirnya pada tahun 1983, komite X3J11 dibentuk untuk menyediakan bahasa yang tidak ambigu dan independen dari mesin apapun. Pada tahun 1989, standar diterima dan standar ini diperbarui pada tahun 1999 (Deitel dan Deitel, 2008).

1.2 Lingkungan Pengembangan Bahasa C

Bahasa C memiliki enam fase untuk dieksekusi seperti yang ditunjukkan pada Gambar 1.1. Fase 1 adalah fase pembuatan program di dalam file sumber, yang dapat dilakukan dengan editor. Editor pemrograman C saat ini ada banyak, beberapa ada yang langsung terintegrasi dengan compilernya seperti misalnya editor yang disediakan Borland C, Turbo C, Code Block, Eclipse, Microsoft Visual Studio dan sebagainya. Ketika pemrogram sudah selesai dibuat di dalam editor, maka pemrogram bisa menyimpannya dengan ekstensi .c.

Fase kedua dan ketiga adalah preprosesing dan kompilasi. Kompilasi adalah proses menerjemahkan program C ke dalam bahasa mesin yang juga disebut sebagai kode objek. Dalam sistem C, program preprosesor melakukan eksekusi secara otomatis sebelum kompiler memulai penerjemahan. Preprosesor melakukan perintah khusus yang disebut sebagai *preprocessor directives*, yang menyatakan bahwa manipulasi tertentu akan dilakukan sebelum program dikompilasi. *Preprocessor directives* yang paling sering digunakan misalnya `stdio.h`. Kesalahan pada sintaks akan dideteksi pada fase ini, dimana kompiler akan memberitahu pemrogram setiap kesalahannya. Sehingga pemrogram dapat memperbaiki programnya melalui editor dan kemudian melakukan fase ini kembali. Contoh kesalahan sintaks misalnya kesalahan menuliskan kata kunci atau kesalahan penulisan perintah.



Gambar 1.1 Lingkungan Pengembangan Bahasa C

Fase keempat adalah Linking. Sebuah program C bisa umumnya berisi referensi ke fungsi yang didefinisikan di tempat lain, misalnya dalam library standar atau *library* buatan dari project tertentu. Linker akan menghubungkan kode objek dengan fungsi dari library untuk membentuk *executable image*. Dalam sistem Linux, perintah untuk melakukan kompilasi dan menghubungkan program disebut cc atau gcc). Jika program telah dikompilasi dan dihubungkan dengan benar, maka sebuah file berekstensi .out akan dihasilkan.

Fase kelima adalah disebut loading. Sebelum sebuah program dapat dieksekusi, program harus ditempatkan pada memori terlebih dahulu. Hal ini akan

dilakukan oleh loader. Loader akan mengambil *executable image* dari penyimpanan dan mengirimkannya ke dalam memori. Semua library tambahan yang mendukung program juga diambil.

Fase keenam adalah eksekusi yang dilakukan oleh CPU. Permasalahan kadang tidak hanya terjadi pada fase dua dan tiga. Namun ada jenis kesalahan yang terjadi ketika program sudah dieksekusi. Kesalahan seperti itu biasanya disebut sebagai *run-time error*. Misalnya jika pemrogram melakukan pembagian dua buah bilangan yaitu a dibagi b, dan b diinput dengan nol, maka akan terjadi *run-time error* berupa *divide by zero error*. Pemrogram harus dapat menangani kesalahan semacam itu, dengan melakukan perbaikan pada program melalui editor.

1.3 Kelebihan Bahasa C

Bahasa C memiliki beberapa kelebihan, diantaranya sebagai berikut :

- Bahasa C tersedia pada hampir semua komputer
- Bahasa C saat ini tidak tergantung pada hardware
- Program dalam Bahasa C dapat dibuat portable pada sebagian besar komputer, jika disusun dengan design yang baik.
- Mudah dipelajari
- Bahasanya terstruktur dengan baik
- Dapat menghasilkan program yang efisien
- Dapat menangani aktivitas pada tingkat rendah (tingkat mendekati mesin)
- Dapat dikompilasi pada berbagai platform komputer

1.4 Penutup

Kesimpulan

- Bahasa C merupakan perkembangan dari dari dua Bahasa sebelumnya yaitu BCPL dan B
- Bahasa C diciptakan oleh Dennis Ritchie di Bell Laboratories dan diimplementasikan pada komputer DEC PDP-11 di tahun 1972.

- Saat ini terdapat banyak sistem operasi yang dikembangkan dengan Bahasa C dan/atau C++.
- Pada tahun 1989, standar C dibentuk dan standar ini diperbarui pada tahun 1999
- Bahasa C memiliki enam fase : penyusunan program, preprosesing, kompilasi, linking, linking dan eksekusi
- Kesalahan sintaks yaitu berupa kesalahan penulisan program dapat dicek pada fase kompilasi
- Kesalahan berupa *run-time error* dapat diketahui pada fase eksekusi.
- Bahasa C memiliki beberapa kelebihan seperti : tersedia pada hampir semua komputer, tidak tergantung pada hardware, portable pada sebagian besar komputer, mudah dipelajari, menghasilkan program yang efisien, dan dapat menangani aktivitas pada tingkat rendah (tingkat mendekati mesin)

Latihan

1. Siapakah penemu Bahasa C?
2. Jelaskan kelebihan Bahasa C!
3. Jelaskan keenam fase dalam lingkungan pengembangan Bahasa C!
4. Jelaskan apa yang dimaksud dengan kompiler!
5. Jelaskan perbedaan *syntax error* dengan *run-time error*!

Umpan Balik

Setelah menyelesaikan bab ini, mahasiswa diharapkan memiliki pengetahuan tentang Bahasa C dan mampu menjelaskan bagaimana tahapan suatu program, mulai dari ditulis hingga dapat dieksekusi. Pemahaman akan bagian awal ini sangat penting untuk dapat melanjutkan kepada bab-bab berikutnya.

BAB II

DASAR-DASAR BAHASA C

Standar Kompetensi

Mahasiswa diharapkan mampu menerjemahkan permasalahan dalam dunia nyata ke dalam algoritma dan mengimplementasikannya ke dalam Bahasa C setelah mempelajari mata kuliah ini.

Kompetensi Dasar

Setelah mempelajari materi ini, mahasiswa diharapkan mampu memahami struktur umum pemrograman C dan menerapkan instruksi keluaran dalam Bahasa C.

Indikator Pencapaian Kompetensi

Adapun indikator hasil belajar adalah sebagai berikut :

1. Mahasiswa mampu menjelaskan struktur umum Bahasa C
2. Mahasiswa mampu menerapkan sintaks keluaran dan string kontrol dalam Bahasa C
3. Mahasiswa mampu menjelaskan preprosesor include dan define
4. Mahasiswa mampu menggunakan komentar dalam program C
5. Mahasiswa mampu menyusun pengenalan yang benar dalam Bahasa C
6. Mahasiswa mampu menjelaskan kesalahan dan memodifikasi sebuah program dalam Bahasa C

MATERI

2.1 Bahasa C

Bahasa C seperti bahasa pemrograman lainnya, tentu saja memiliki aturan penulisan. File program yang dihasilkan oleh program C akan memiliki ekstensi .c. Bahasa C bersifat *case sensitive*, dimana huruf besar dan kecil dibedakan. Setiap akhir dari sebuah instruksi harus diakhiri tanda *semicolon* (;). Program

dalam bahasa c akan menghasilkan tiga file yaitu : file sumber yang berkeestensi .c, file objek yang berekstensi .obj, dan file yang dapat dieksekusi berekstensi .exe.

2.2 Contoh Program dalam Bahasa C

Sebelum mengenal lebih dalam mengenai Bahasa C, ada baiknya untuk melihat dulu program C yang sederhana yaitu program C dengan struktur minimal. Sebuah program C pada dasarnya tersusun dari beberapa bagian berikut ini :

- Preprosesor
- Fungsi
- Variabel
- Pernyataan/*Statement*
- Komentar

Berikut ini adalah contoh program sederhana yang menampilkan kalimat “Hello World” ke layar.

```
#include <stdio.h>
int main()
{
    printf("Hello World");
    return 0;
}
```

Penjelasan program di atas adalah sebagai berikut :

- Baris pertama pada program yaitu **#include <stdio.h>** merupakan preprosesor yang memberitahu *compiler* untuk menyertakan file `stdio.h` sebelum melakukan kompilasi program.
- Baris kedua pada program yaitu **int main()** merupakan fungsi utama (bagian utama pada program), dimana eksekusi program dimulai.
- Baris berikutnya **printf("Hello World");** adalah perintah untuk melakukan pencetakan kalimat ke dalam layar. Adapun fungsi `printf` digunakan untuk memerintahkan *compiler* untuk mencetak ke dalam layar, sedangkan kalimat yang ingin dicetak harus diletakkan ke dalam tanda kurung dan menggunakan tanda petik dua. Setiap perintah di dalam Bahasa C harus diakhiri tanda koma, sehingga setelah perintah `printf`, maka harus dituliskan tanda ;

- Baris terakhir **return 0;** akan mengakhiri program dan memberikan nilai 0.
- Hal yang perlu diingat adalah bahwa semua perintah di dalam fungsi (dalam hal ini fungsi main) harus diapit oleh tanda kurung kurawal buka { dan kurung kurawal tutup }

2.3 Fungsi *main()*

Program C pada hakekatnya tersusun atas sejumlah blok fungsi. Sebuah program minimal mengandung sebuah fungsi. Setiap fungsi terdiri dari satu atau beberapa pernyataan yang secara keseluruhan dimaksudkan untuk melaksanakan tugas khusus.

Fungsi **main()** harus ada pada program, sebab fungsi inilah yang menjadi titik awal dan akhir eksekusi program. Tanda { diawal fungsi menyatakan awal tubuh fungsi dan juga sebagai awal program dijalankan, sedangkan tanda } di akhir fungsi merupakan akhir bagian isi fungsi dan sekaligus adalah akhir dari eksekusi program. Berikut adalah struktur dari program C.

```
int main()
{
    Isi fungsi main
}
```

2.4 Fungsi *printf()*

Fungsi **printf()** merupakan fungsi yang umum dipakai untuk menampilkan suatu keluaran pada layar peraga. Dalam bentuk umum, format **printf()** adalah sebagai berikut :

```
printf(kontrol string, daftar pernyataan);
```

String kontrol dapat berupa satu atau sejumlah karakter yang akan ditampilkan ataupun berupa penentu format yang akan mengatur penampilan dari argumen yang terletak pada **daftar argumen**. Mengenai penentu format diantaranya berupa :

%d : Untuk menampilkan bilangan bulat (integer)

%f : Untuk menampilkan bilangan pecahan (float)

%c : Untuk menampilkan sebuah karakter

%s : Untuk menampilkan sebuah string atau untaian karakter

Contoh program pencetakan dengan string kontrol yaitu :

```
#include <stdio.h>
int main()
{
    int bilangan=5;
    printf("Nilai bilangan adalah %d",bilangan);
    return 0;
}
```

Pada program di atas, `int bilangan=5;` berarti pembuat program mendeklarasikan sebuah variabel bertipe integer (bilangan bulat) dan diberi nilai 5. Pencetakan dilakukan dengan string kontrol `%d` karena yang dicetak adalah bilangan bulat. `%d` akan digantikan dengan nilai variabel bilangan yaitu 5. Adapun output dari program di atas adalah

```
Nilai bilangan adalah 5
```

2.5 Preprocessor

Preprocessor adalah proses yang dilakukan sebelum program di kompilasi. Dalam buku ini akan dibahas dua buah preprosesor yaitu `include` dan `define`. Penulisan preprosesor diawali oleh tanda (`#`).

2.5.1 Preprosesor `#include`

Preprosesor `include` diikuti oleh nama file. File biasanya disebut sebagai file header. Salah satu contoh file header yang paling penting dan mendasar dalam pemrograman C adalah `stdio.h`. File `stdio.h` berisi library untuk standar input dan output. Pemanggilan file header akan diapit oleh tanda kurung `<` dan `>`. Contoh penggunaan preprosesor `include` yaitu :

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

2.5.2 Preprosesor `#define`

Preprosesor lain yang sering digunakan yaitu `define`. `Define` ditulis dalam bentuk berikut :

```
#define IDENTIFIER pengganti
```

Kode di atas berarti bahwa IDENTIFIER yang dituliskan oleh pembuat program di dalam akan digantikan dengan pengganti. Dimanapun ditemukan IDENTIFIER tersebut di dalam program, maka identifier akan digantikan dengan pengganti. Contohnya dalam program adalah sebagai berikut :

```
#include <stdio.h>
#define mulai {
#define selesai }

int main()
mulai
    int bilangan=5;
    printf("Nilai bilangan adalah %d",bilangan);
    return 0;
selesai
```

Program di atas menggunakan preprosesor `#define` untuk menggantikan tanda kurung kurawal buka `{` menjadi kata mulai dan tanda kurung kurawal tutup `}` menjadi kata selesai. Pada program, maka setiap kurung kurawal `{` dan `}` dapat diganti dengan kata mulai dan selesai.

2.6 Memberikan Komentar pada Program

Komentar atau comment adalah naskah program yang tidak akan diproses oleh *compiler*. Pada saat proses kompilasi berlangsung, teks program yang termasuk ke dalam komentar akan diabaikan oleh compiler. Kehadiran komentar di dalam program sangat dibutuhkan terutama jika program yang dibuat sudah masuk ke skala besar dan kompleks. Setidaknya ada 3 alasan mengapa komentar perlu ditulis :

1. Dokumentasi
2. Debugging
3. Maintenance

Bahasa C menyediakan dua cara menulis komentar :

1. Karakter `“//”` digunakan untuk mengawali penulisan komentar dalam satu baris. Karakter yang ditulis sampai akhir baris akan diperlakukan sebagai komentar. Cara ini hanya bisa diterapkan pada komentar satu baris. Jika cara ini akan diterapkan pada komentar beberapa baris, maka pada setiap baris komentar karakter `“//”` harus ditulis di awal komentar.

2. Karakter “/*” digunakan untuk mengawali penulisan komentar satu baris atau lebih, sampai dijumpai karakter “*/”. Cara ini memungkinkan kita menulis komentar lebih dari satu baris tanpa harus menulis tanda komentar berulang-ulang. Cukup awali komentar dengan menulis “/*” lalu akhiri komentar dengan menulis “*/”

Contoh penggunaan komentar adalah sebagai berikut :

```
/*program ini adalah program sederhana untuk mencetak
kalimat Hello World ke dalam layar*/
#include <stdio.h>
int main()
{
    //perintah printf : untuk mencetak sebuah kalimat
    printf("Hello World");
    return 0; //perintah ini akan mengakhiri program
}
```

Pada program di atas, digunakan kedua contoh pemberian komentar. Untuk komentar yang lebih dari satu baris maka digunakan jenis komentar /* dan ditutup dengan */ pada akhir komentar, sedangkan komentar dengan satu baris menggunakan tanda // di awal komentar.

2.7 Karakter Khusus

Karakter khusus adalah sebuah karakter yang bukan huruf, angka, simbol, atau tanda baca. Karakter khusus misalnya, karakter format khusus seperti tanda ayat. Bahasa C menyediakan beberapa karakter khusus, seperti :

- \a : untuk bunyi bell (alert)
- \b : mundur satu spasi (backspace)
- \n : ganti baris baru (new line)
- \r : ke kolom pertama, baris yang sama (carriage return)
- \t : tabulasi
- \0 : nilai kosong (null)
- \' : karakter petik tunggal
- \" : karakter petik ganda
- \\ : karakter garis miring

Contoh penggunaan karakter khusus adalah sebagai berikut :

```
#include <stdio.h>
int main()
```

```
{
    printf("\\"Hello\\" \n World");
    return 0;
}
```

Keluaran dari perintah di atas adalah :

```
"Hello"
World
```

Penggunaan karakter khusus \" akan menampilkan output tanda \" di awal dan di akhir Hello, sedangkan tanda \n di belakang Hello akan menghasilkan baris baru, sehingga kata World akan berada di bawah kata Hello.

Berikut ini adalah contoh lain dari penggunaan karakter khusus \n :

```
#include <stdio.h>
int main()
{
    printf("Hello ");
    printf("World");
    printf("\nIni program pertama saya");
    return 0;
}
```

Keluaran dari perintah di atas adalah :

```
Hello World
Ini program pertama saya
```

Penulisan printf tanpa adanya karakter khusus \n tidak akan membuat baris baru. Hal itu dapat dilihat pada printf pertama dan kedua, tanpa adanya \n, maka keluaran yang dihasilkan akan menjadi satu baris yaitu Hello World, sedangkan dengan dituliskannya \n di depan printf yang ketiga akan membuat kalimat di dalamnya ditulis pada baris baru.

Pertanyaan

Tahukah Anda apakah keluaran dari program berikut ini :

```
#include <stdio.h>
int main()
{
    printf("Hello\bWorld ");
    printf("\nThis is\r My first program\t using C");
    printf("\nLet\'s see the result!!!");
    return 0;
}
```

2.8 Identifier/Pengenal

Identifier atau pengenal di dalam Bahasa C adalah sebuah nama yang digunakan untuk menandai sebuah variabel, fungsi atau apapun yang dibuat oleh pengguna. Sebuah pengenal harus diawal oleh huruf A-Z, a-z, atau underscore (_) kemudian dapat diikuti oleh huruf atau angka. Identifier dalam C tidak diperkenankan mengandung karakter @, % atau spasi. Panjang pengenal boleh lebih dari 31 karakter, tapi hanya 31 karakter pertama yang akan dianggap berarti. Pengenal di dalam C tidak boleh menggunakan kata kunci yang sudah digunakan oleh Bahasa C, seperti main, if, for dan lain sebagainya. Bahasa C adalah bahasa yang bersifat *case sensitive*, sehingga pengenal bernama Bilangan dan bilangan akan dianggap sebagai dua pengenal yang berbeda. Berikut ini adalah contoh pengenal yang benar :

bilangan	bilangan1	bilangan_satu	_bilangan
----------	-----------	---------------	-----------

Sedangkan contoh pengenal yang salah adalah sebagai berikut :

1bilangan	bilanga satu	bil@angan	bilangan%
-----------	--------------	-----------	-----------

Pertanyaan

1. Tahukah Anda mengapa pengenal di atas salah?
2. Bagaimana dengan penggunaan tanda \$ di dalam sebuah pengenal? Anda bisa mencobanya sendiri dan menemukan jawabannya.

Bahasa C. Kata kunci tidak boleh digunakan sebagai pengenal oleh pembuat program. Adapun kata kunci yang dalam Bahasa C ditunjukkan pada Gambar 2.1 :

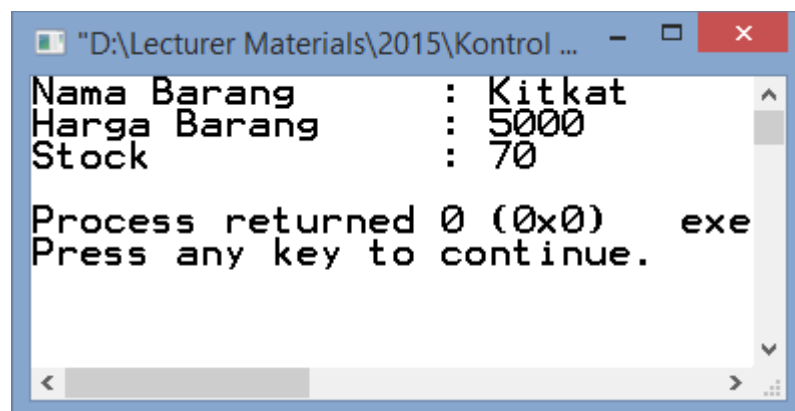
auto	else	long	switch
break	enum	register	typedef
case	extern	return	union
char	float	short	unsigned
const	for	signed	void
continue	goto	sizeof	volatile
default	if	static	while
do	int	struct	_Packed
double			

Gambar 2.1 Kata Kunci dalam Bahasa C (Tutorials Poin, 2014)

2.11 Contoh Soal

Bagian ini akan memberikan beberapa contoh soal yang terkait dengan bab yang dibahas dan sekaligus pembahasannya.

1. Buatlah program untuk memberikan tampilan berikut ini



Gambar 2.2. Tampilan Output Soal Latihan 1

Pembahasan :

```
#include <stdio.h>
void main()
{
```

```
printf("Nama Barang\t : Kitkat\n");
printf("Harga Barang\t : 5000\n");
printf("Stock\t\t : 70\n");
}
```

Untuk mencetak data pada soal agar terlihat rapi dan tanda titik duanya (:) lurus, bisa digunakan karakter khusus \t agar tidak kesulitan dalam menghitung spasi. Harga diberikan \t sebanyak dua kali karena karakternya lebih pendek dibandingkan dengan dua yang lain.

2.10 Penutup

Kesimpulan

- Bahasa C merupakan bahasa yang *case sensitive*.
- Bahasa C secara dasar tersusun atas preprosesor `include`, fungsi `main`, variabel, pernyataan dan komentar.
- Fungsi `main` harus diawali oleh tanda kurung kurawal { dan }
- Fungsi `printf` digunakan untuk mencetak tampilan ke dalam layar.
- Karakter kontrol di dalam fungsi `printf` digunakan untuk mencetak tipe bilangan, karakter dan untaian karakter.
- Preprosesor `include` digunakan untuk merujuk file library tertentu, sedangkan `define` digunakan untuk mengganti item apapun di dalam program dengan kata yang didefinisikan pada preprosesor `define`.
- Pemberian komentar pada program C bisa dilakukan dengan tanda backslash dua kali `//` untuk satu baris komentar atau `/*` dan `*/` untuk komentar yang lebih dari satu baris.
- C menyediakan beberapa karakter khusus untuk mencetak karakter yang bukan huruf, angka, simbol, atau tanda baca. Penggunaan karakter khusus selalu diawali oleh tanda backslash \
- Pengenal dalam Bahasa C memiliki panjang maksimal 31 karakter dan boleh diawali huruf atau underscore (`_`) dan bisa diikuti oleh huruf atau angka
- Pengenal di dalam Bahasa C tidak boleh mengandung spasi, `@`, atau `%`

- Kata kunci adalah kata-kata yang digunakan oleh C sebagai bagian dari sintaksnya. Kata kunci di dalam C tidak boleh digunakan sebagai pengenal.

Latihan

1. Buatlah program untuk mencetak data diri Anda berupa nama, alamat dan hobby!
2. Apakah keluaran dari program berikut ini :

```
#include <stdio.h>

int main()
{
    //mencetak string
    printf("Nim Mahasiswa\t: %s\n", "0410960044");
    printf("Nama Mahasiswa\t: %s\n", "Nadine");

    //mencetak integer
    printf("Umur \t\t: %d\n", 18);

    //mencetak float dan char
    printf("Nilai\t\t: %f, nilai huruf : %c\n", 90.5, 'A');
    return 0;
}
```

3. Ceklah apakah kesalahan dari program berikut ini dan carilah dan penyebabnya :

```
#include <stdio.h>

int Main()
{
    /Error Checking
    printf("%d Tali %d Uang\n", 1, 3);
    Printf("Dibawah ini adalah sebuah pepatah\n");
    printf("Ada %c Ada %s\n", "Gula", "Semut");
    return(0);
}
```

4. Buatlah tampilan seperti berikut ini :

```
"E:\Lecturer Materials\Prokom\modul\latihan\modul1_1.exe"

=====
"KOPERASI CLUSTER SAINS"
STRUK BELANJA
=====

No Struk      :123      Kasir    :Dino

=====
|  Barang  |  Harga  | Qty | Subtotal |
|Gerry     |    2000 |  3  |    6000 |
|Monde Butter | 60000 |  1  | 60000 |
|Total     |         |  4  | 66000 |
=====
```

Gambar 2.3. Tampilan Output Program Kasir Sederhana

Umpan Balik

Setelah menyelesaikan bab ini, mahasiswa diharapkan memahami struktur dasar program dalam Bahasa C. Mahasiswa mampu membuat program sederhana untuk menampilkan suatu teks. Selain itu mahasiswa juga mengetahui kesalahan suatu program dan dapat memperbaikinya. Pemahaman akan bab ini sangat penting untuk dapat melanjutkan pada bab berikutnya.

BAB III

TIPE DATA DAN OPERATOR

Standar Kompetensi

Mahasiswa diharapkan mampu menerjemahkan permasalahan dalam dunia nyata ke dalam algoritma dan mengimplementasikannya ke dalam Bahasa C setelah mempelajari mata kuliah ini.

Kompetensi Dasar

Setelah mempelajari materi ini, mahasiswa diharapkan mampu memahami dan menggunakan berbagai tipe data dan operator dalam Bahasa C.

Indikator Pencapaian Kompetensi

Adapun indikator hasil belajar adalah sebagai berikut :

1. Mahasiswa mampu menerapkan tipe data dalam Bahasa C
2. Mahasiswa mampu menggunakan sintaks keluaran dan masukan dengan berbagai tipe data
3. Mahasiswa mampu menerapkan berbagai operator dalam Bahasa C
4. Mahasiswa mampu menjelaskan keluaran dari program C yang melibatkan berbagai tipe data dan operator

MATERI

3.1 Tipe Data Dalam Bahasa C

Data merupakan suatu nilai yang bisa dinyatakan dalam bentuk konstanta atau variabel. Konstanta menyatakan nilai yang tetap, sedangkan variabel menyatakan nilai yang dapat diubah-ubah selama eksekusi berlangsung. Tipe data di dalam C, ditunjukkan pada Tabel 3.1.

Tabel 3.1. Tipe Data dalam Bahasa C (Parlente, 2003)

Jenis	Nama Tipe Data	Keterangan
Bilangan Bulat (Integer)	char	char merupakan tipe data dengan panjang 8 bit untuk merepresentasikan karakter (huruf, angka, tanda baca dan karakter khusus). Char sebenarnya merepresentasikan nilai ASCII setiap karakter. Banyaknya karakter yang dapat ditampung adalah 256.
	short	short merupakan tipe data yang mendefinisikan bilangan bulat dengan interval -32768 sampai 32767. Panjang memori yang dibutuhkan adalah 16 bit
	int	int merupakan tipe data bilangan bulat <i>default</i> dengan interval dengan panjang minimal 16 bit dan umumnya berukuran 32 bit atau dengan interval data -2147483648 sampai 2147483647
	long	long merupakan tipe data bilangan bulat yang besar, dengan panjang minimal 32 bit
Bilangan real (floating point)	float	Bilangan real dengan presisi tunggal. Ukuran memori 32 bit. Ketelitian sampai 6 di belakang koma
	double	Bilangan real dengan presisi ganda. Ukuran memori 64 bit. Ketelitian sampai 15 di belakang koma
	long double	Bilangan real dengan presisi ganda. Ukuran memori 80 bit. Ketelitian sampai 19 di belakang koma
Tanpa tipe	void	Tak bertipe

Penggunaan kata kunci `unsigned` di depan tipe data integer akan mengakibatkan suatu data tidak mendukung bilangan negatif. Hal ini akan

menyebabkan interval tipe data akan menjadi dua kali lipat. Misalnya saja `unsigned short` memiliki jangkauan interval antara 0 sampai 65536. Penulisan tipe `char` berbeda dengan tipe integer lainnya. Misalnya jika `x` adalah `short`, sedangkan `y` adalah `char`, maka berikut ini adalah contoh pemberian nilainya :

```
x = 75;
```

```
y = 65 atau y='A'
```

Pemberian nilai pada tipe data `char` dalam bentuk angka (tanpa tanda petik tunggal) mengindikasikan pemberian nilai ASCII-nya, sedangkan pemberian nilai dengan tanda petik tunggal mengindikasikan nilai karakter yang diberikan kepada `y`. Nilai ASCII dari `A` adalah 65, sehingga pemberian nilai `y` pada kedua contoh di atas adalah sama.

Pertanyaan :

1. Coba hitunglah jangkauan `unsigned integer`?
2. Dapatkah Anda menjelaskan hubungan antara panjang bit dengan interval tipe data integer?

3.2 Variabel

Variabel adalah suatu pengenalan (identifikasi) yang digunakan untuk mewakili suatu nilai tertentu di dalam proses program. Berbeda dengan konstanta yang nilainya selalu tetap, nilai dari suatu variabel bisa diubah-ubah sesuai kebutuhan.

Nama dari suatu variabel dapat ditentukan sendiri oleh pemrogram dengan aturan yang sama seperti pada aturan pengenalan, yaitu sebagai berikut :

- Variabel harus diawali dengan huruf (`A..Z`, `a..z`) atau karakter garis bawah (`_`).
- Variabel dapat berupa huruf, digit (`0..9`) atau karakter garis bawah atau tanda dolar (`$`).
- Panjang variabel boleh lebih dari 31 karakter, tapi hanya 31 karakter pertama yang akan dianggap berarti.
- Variabel tidak boleh menggunakan nama yang tergolong sebagai kata kunci (*reserved words*) seperti `int`, `if`, `while`, dan sebagainya.

3.2.1 Mendeklarasikan Variabel

Variabel digunakan dalam program untuk menyimpan suatu nilai, dan nilai yang ada padanya dapat diubah-ubah selama eksekusi program berlangsung. Variabel yang akan digunakan dalam program harus dideklarasikan terlebih dahulu. Pengertian deklarasi disini berarti memesan memori dan menentukan jenis data yang bisa disimpan di dalamnya. Bentuk umum deklarasi variabel adalah sebagai berikut:

```
Nama_tipe_data daftar_tipe_data
```

Pada pendeklarasian variabel, daftar variabel dapat berupa variabel atau beberapa variabel yang dipisahkan dengan koma. Contoh:

```
int var_bulat1;  
float var_pecahan1, var_pecahan2;
```

3.2.2 Memberikan nilai ke variabel

Untuk memberikan nilai ke variabel yang telah dideklarasikan, maka bentuk umum pernyataan yang digunakan adalah sebagai berikut:

```
Nama_variabel=nilai
```

Pemberian nilai suatu variabel dapat dilihat pada contoh berikut ini :

```
int var_bulat = 10;  
double var_pecahan = 10.5;  
int bilangan1;  
bilangan1=23;
```

3.3 Konstanta

Konstanta menyatakan nilai yang tetap sepanjang jalannya program. Berbeda dengan variabel, suatu konstanta tidak dideklarasikan dan konstanta tidak bisa diubah-ubah. Namun seperti halnya variabel, konstanta juga memiliki tipe. Penulisan konstanta mempunyai aturan tersendiri, sesuai dengan tipe masing-masing.

- Konstanta karakter misalnya ditulis dengan diawali dan diakhiri dengan tanda petik tunggal, contohnya: 'A' dan '@'.
- Konstanta integer ditulis tanpa mengandung pemisah ribuan dan tak mengandung bagian pecahan, contohnya : -1 dan 32767.

- Konstanta real (*float* dan *double*) bisa mengandung pecahan (dengan tanda berupa titik) dan nilainya bisa ditulis dalam bentuk eksponensial (menggunakan tanda e) contohnya: `27.5f` (untuk tipe *float*) atau `27.5` (untuk tipe *double*) dan `2.1e+5` (maksudnya $2,1 \times 10^5$)

Konstanta dituliskan dengan menggunakan preprosesor `define`, seperti contoh berikut ini :

```
#define PI 3.14f
#define BUNGA 0.5
#define HURUF 'A'
```

Pada contoh di atas terdapat tiga buah konstanta. Konstanta pertama bernama `PI` dengan tipe *float*, bernilai 3.14, sedangkan konstanta kedua bernama `BUNGA` dengan tipe *double* bernilai 0.5. Konstanta terakhir bernama `HURUF` dengan tipe *char* dan bernilai A. Untuk mempermudah dalam membedakan variabel dan konstanta, biasanya konstanta akan diberi nama dengan huruf kapital, namun hal itu tidak wajib.

Pertanyaan :

Apakah keluaran dari program berikut ini

```
#include <stdio.h>
#define HURUF 'A'

void main()
{
    char karakter = HURUF + 4;
    HURUF = 67;
    printf("Nilai huruf : %c ", HURUF);
    printf("Nilai karakter : %c ", karakter);
}
```

3.4 Perintah Masukan dengan fungsi `scanf()`

Fungsi ini digunakan untuk memasukkan berbagai jenis data. Bentuk `scanf()` sesungguhnya menyerupai `printf()`. Fungsi ini melibatkan penentu format yang pada dasarnya sama digunakan pada `printf()`. Secara umum bentuk `scanf()` adalah sebagai berikut:

```
scanf("string kontrol", daftar argumen);
```

Beberapa string kontrol dasar sudah diperkenalkan pada bab sebelumnya. Namun string kontrol sebenarnya ada banyak. Adapun daftar string kontrol yang lebih lengkap ditunjukkan pada Gambar 3.1

%u	untuk menampilkan data bilangan tak bertanda (<i>unsigned</i>) dalam bentuk desimal.
%d	untuk menampilkan bilangan integer bertanda (<i>signed</i>) dalam bentuk desimal
%i	
%o	untuk menampilkan bilangan bulat tak bertanda dalam bentuk oktal.
%x	untuk menampilkan bilangan bulat tak bertanda dalam bentuk heksadesimal
%X	(%x → notasi yang dipakai : a, b, c, d, e dan f sedangkan %X → notasi yang dipakai : A, B, C, D, E dan F)
%f	untuk menampilkan bilangan real dalam notasi : dddd.dddddd
%e	untuk menampilkan bilangan real dalam notasi eksponensial
%E	
%g	untuk menampilkan bilangan real dalam bentuk notasi seperti %f, %E atau %F
%G	bergantung pada kepresisian data (digit 0 yang tak berarti tak akan ditampilkan)
l	merupakan awalan yang digunakan untuk %d, %u, %x, %X, %o untuk menyatakan long int (misal %ld). Jika diterapkan bersama %e, %E, %f, %F, %g atau %G akan menyatakan <i>double</i>
L	Merupakan awalan yang digunakan untuk %f, %e, %E, %g dan %G untuk menyatakan <i>long double</i>
h	Merupakan awalan yang digunakan untuk %d, %i, %o, %u, %x, atau %X, untuk menyatakan <i>short int</i> .

Gambar 3.1. String Kontrol untuk Perintah printf() dan scanf()

Daftar argumen dalam perintah scanf() adalah variabel yang akan dimasukkan oleh pengguna. Adapun daftar argumen harus diawali oleh tanda dan (&) pada setiap argumennya. Tanda & menyatakan alamat dari suatu variabel yang akan diberikan masukan oleh pengguna. Contoh program yang menggunakan fungsi scanf() adalah sebagai berikut :

```
#include <stdio.h>
void main()
{
    char kar;
    int x,y;
    float f;
    printf("Masukkan sebuah karakter : "); scanf("%c",&kar);
    printf("Masukkan dua bilangan : "); scanf("%d %d",&x,&y);
    printf("Masukkan bilangan real : "); scanf("%f",&f);

    printf("Nilai karakter :%c, bilangan 1 :%d,",kar,x);
    printf(" bilangan 2 : %d dan bilangan 3 : %f",y,f);
}
```

Keluaran dari program tersebut adalah sebagai berikut :

Masukkan sebuah karakter : B
Masukkan dua bilangan : 34 789
Masukkan bilangan real : 87.5
Nilai karakter :B, bilangan 1 :34, bilangan 2 : 789 dan bilangan 3 : 87.500000

Saat dijalankan, program akan meminta pengguna untuk memasukkan sebuah karakter. Misalnya pengguna memasukkan B dan kemudian menekan *enter* dari *keyboard*, maka program akan meminta pengguna memasukkan dua buah bilangan bulat dengan dipisahkan spasi. Dalam contoh di atas, pengguna memasukkan nilai 34 dan 789. Terakhir, program akan meminta pengguna memasukkan bilangan real, dan misalnya pengguna memasukkan 87.5. Sebagai keluaran akhir, program akan menampilkan nilai-nilai dari bilangan yang telah diinputkan. Dalam contoh dapat dilihat bahwa nilai bilangan 3 memiliki 6 digit di belakang koma, hal ini karena nilai default dari suatu data bertipe `float` adalah 6 digit di belakang koma. Untuk mencetak suatu tipe data float dengan digit yang diinginkan adalah dengan cara sebagai berikut :

<pre>printf("Nilai bilangan 3 : %.2f", f);</pre>
--

Source code di atas akan memberikan dua digit di belakang tanda koma.

3.5 Operator

Operator merupakan simbol atau karakter yang biasa dilibatkan dalam program untuk melakukan sesuatu operasi atau manipulasi, seperti menjumlahkan dua buah nilai, memberikan nilai ke suatu variabel, membandingkan kesamaan dua buah nilai. Sebagian operator C tergolong sebagai operator binary, yaitu operator yang dikenakan terhadap dua buah nilai (*operand*). Contoh :

$$a + b$$

Simbol plus (+) merupakan operator untuk melakukan operasi penjumlahan dari kedua *operand*-nya (yaitu a dan b). Karena operator penjumlahan melibatkan dua operand, maka ini tergolong sebagai operator binary. Simbol minus (-) juga merupakan operator. Simbol ini termasuk sebagai operator binary dan juga unary. Operator unary yaitu operator yang hanya memiliki sebuah operand, contohnya :

$$-c$$

3.5.1 Operator Aritmatika

Operator untuk operasi aritmatika yang tergolong sebagai operator binary adalah :

- * : perkalian
- / : pembagian
- % : sisa pembagian
- + : penjumlahan
- : pengurangan

Adapun operator yang tergolong sebagai operator unary.

- : tanda minus
- + : tanda plus

Contoh pemakaian operator aritmatika dalam program adalah sebagai berikut :

```
/* program untuk menghitung keliling persegi panjang */
#include <stdio.h>
void main()
{
    int panjang, lebar, keliling;
    printf("Masukkan panjang : "); scanf("%d", &panjang);
    printf("Masukkan lebar : "); scanf("%d", &lebar);
    keliling = 2 * (panjang + lebar);
    printf("Keliling : %d", keliling);
}
```

Percobaan :

Tuliskanlah program di atas ke dalam compiler dan lihatlah apa yang dihasilkan!

Contoh operator aritmatika lainnya dapat dilihat pada program berikut :

```
#include <stdio.h>
void main()
{
    int a=5, b=2, c, d;
    c=a/b;
    d=5 % b;
    printf("Nilai c : %d dan d : %d", c, d);
}
```

Keluaran dari program di atas adalah :

Nilai c : 2 dan d : 1

Nilai c berasal dari 5 dibagi 2, karena c merupakan integer dan berasal dari pembagian bilangan integer, maka c tidak akan mendukung nilai 2.5, sehingga nilai tersebut akan dibulatkan ke bawah menjadi 2. Perlu diingat bahwa operasi untuk bilangan integer selalu dibulatkan ke bawah. Sedangkan nilai d berasal dari 5 modulo 2, yaitu sisa pembagian dari 5 dengan 2.

3.5.2 Operator Penurunan dan Penaikan

Selain operator aritmatika, C menyediakan operator yang berkaitan dengan operasi aritmatika. Operator tersebut disebut sebagai operator penaikan dan operator penurunan, yaitu :

++ : operator penaikan
-- : operator penurunan

Operator penaikan digunakan untuk menaikkan nilai variabel sebesar satu. Penempatan operator terhadap variabel dapat dilakukan di muka atau di belakangnya.

<pre>x = x+1; y = y-1; Bisa ditulis menjadi : ++x; --y; atau : x++; y--;</pre>
--

Pemberian tanda ++ di depan operandi merupakan *pre-increment*, sedangkan peletakan operator di belakang berarti *post-increment*. *Pre-increment* ditunjukkan oleh contoh pertama, sedangkan *post-increment* ditunjukkan oleh contoh kedua. *Pre-increment* berarti penambahan sebelum operasi lainnya dilaksanakan, sebaliknya *post-increment* berarti penambahan setelah operasi lainnya dilaksanakan. Peletakkan tanda -- di depan berarti sebagai *pre-decrement*, sedangkan peletakannya di belakang berarti *post-decrement*.

Contoh penggunaan operator penaikan dan penurunan dapat dilihat pada potongan listing program berikut ini :

```
b = 2;  
a = b++ *3;
```

Pada contoh di atas, nilai b pada awalnya diberikan 2. Nilai b akan dinaikkan sebesar satu, setelah operasi perkalian antara b dengan 3 dilakukan dan disimpan dalam variabel a. Nilai a pada akhirnya adalah $a=2*3=6$, sedangkan nilai b pada akhirnya akan dinaikkan sebesar satu dan menjadi $b=2+1=3$. Perhatikanlah contoh berikut ini :

```
b = 2;  
a = ++b *3;
```

Perbedaan contoh sebelumnya dengan contoh di atas adalah terletak pada nilai akhir a, dimana pada kasus pertama operasi perkalian dilakukan dahulu baru nilai b dinaikkan, sedangkan pada kasus kedua, nilai b dinaikkan dulu sebesar satu dan menjadi $b=2+1=3$, baru dilakukan operasi perkalian, sehingga nilai $a=3*3=9$.

Pertanyaan :

Coba hitung berapa nilai akhir dari setiap variabel dari ekspresi berikut ini!

```
a=2;  
b=4;  
c= a++ -1;  
b=c-- * ++a;
```

3.5.3 Operator Penugasan

Operator penugasan (*assignment operator*) digunakan untuk memindahkan nilai dari suatu ungkapan (*expression*) ke suatu pengenalan. Operator pengerjaan yang umum digunakan dalam bahasa pemrograman, termasuk bahasa C adalah operator sama dengan (=). Contohnya:

```
luas = 0.5 * alas + tinggi;
```

Maka '=' adalah operator penugasan yang akan memberikan nilai dari ungkapan : $0.5 * \text{alas} + \text{tinggi}$ kepada variabel luas.

Bahasa C juga memungkinkan dibentuknya *statement* penugasan menggunakan operator pengerjaan jamak dengan bentuk sebagai berikut :

pengenal1 = pengenal2 = ... = ungkapan ;

Misalnya :

`a = b = 15;`

maka nilai variabel a akan sama dengan nilai variabel b yaitu 15.

3.5.4 Operator Relasional

Operator relasional adalah operator yang mengecek hubungan antara 2 buah operand. Nilai kembalian dari operator ini adalah boolean. Boolean adalah suatu nilai data yang terdiri dari dua yaitu true atau false. Dalam C, true akan tertulis 1, sedangkan false akan tertulis 0. Adapun daftar operator relasional ditunjukkan pada Tabel 3.2. Penggunaan operator relasional biasanya ditemui pada struktur kontrol pemilihan maupun perulangan.

Tabel 3.2. Daftar Operator Relasional

Operator	Arti	Contoh
==	Sama dengan	$3==2$ (false)
>	Lebih besar	$2>3$ (false)
<	Lebih kecil	$2<3$ (true)
>=	Lebih besar atau sama dengan	$2>=2$ (true)
<=	Lebih kecil atau sama dengan	$4<=3$ (false)
!=	Tidak sama dengan	$1!=2$ (true)

Perbedaan operator == dengan = adalah bahwa operator yang pertama adalah operator relasional yang mengecek apakah operand di kiri sama dengan operand di kanan, sedangkan operator = adalah operator penugasan yaitu

memberikan nilai operand yang di kanan kepada operand di kiri. Berikut adalah contoh potongan program yang melibatkan kedua operator tersebut :

```
int a=25,b=15,c;
printf("a=b : %d", (a==b));
c=a;
printf("Nilai c : %d", c);
```

Keluaran dari program di atas adalah 0 untuk a=b, karena nilai a tidak sama dengan b, sehingga ekspresi (a==b) menghasilkan nilai false atau 0. Sedangkan nilai c adalah sama dengan a yaitu 25.

3.5.5 Operator Logika

Operator logika merupakan operator yang digunakan untuk menggabungkan pernyataan yang mengandung operator relasional. Daftar operator logika ditunjukkan pada Tabel 3.3.

Tabel 3.3. Daftar Operator Logika

Operator	Arti	Contoh
&&	Logical And	((3==2) && (2<3)) (false)
	Logical Or	((3==2) (2<3)) (true)
!	Logical Not	!(5>3) (false)

Contoh penggunaan operator logika ditunjukkan pada potongan listing program berikut ini :

```
int praktikum=75, teori=70;
printf("Status kelulusan : %d", (praktikum>70 && teori>60))
```

Hasil dari program di atas adalah true atau 1 untuk status kelulusan. Dari program dapat dilihat bahwa status kelulusan dinyatakan true (lulus) apabila praktikum di atas 70 dan teori di atas 60. Karena pada contoh nilai praktikum 75 sedangkan teori 70, maka status lulus adalah lulus (true).

3.5.6 Operator Bitwise

Operator ini beroperasi pada nilai bit data. Operator ini digunakan pada pemrograman level bit. Daftar operator bitwise ditunjukkan pada Tabel 3.4

Tabel 3.4. Daftar Operator Bitwise

Operator	Arti	Contoh
&	Bitwise AND	2 & 3 (10 & 11 =10) bernilai 2
	Bitwise OR	2 & 3 (10 11 =11) bernilai 3
^	Bitwise XOR	2 ^ 3 (10 ^ 11 =01) bernilai 1
<<	Bitwise SHIFT LEFT	2<<1 (10 <<1=100) bernilai 4
>>	Bitwise SHIFT RIGHT	2>>1 (10 >>1=1) bernilai 1

Saat menggunakan operator bitwise maka perlu diingat hal berikut ini :

- AND : bernilai 1 apabila kedua operand bernilai 1, selain itu 0
- OR : bernilai 0 apabila kedua operand bernilai 0, selain itu 1
- XOR : bernilai 0 apabila kedua operand bernilai sama, selain itu 1

Operator << akan menggeser bit ke kiri, sehingga menambah 0 di bagian belakang atau sama dengan dikalikan 2. Sedangkan >> akan menggeser bit ke kanan, sehingga akan ada nilai yang hilang mulai dari paling kanan, atau sama dengan dibagi 2.

3.5.7 Operator Kombinasi (Pemendekan)

C menyediakan operator yang dimaksudkan untuk memendekkan penulisan operasi penugasan semacam

```
x = x + 2;  
y = y * 4;  
menjadi  
x += 2;  
y *= 4;
```

Daftar seluruh kemungkinan operator kombinasi dalam suatu pernyataan serta pernyataan padanannya ditunjukkan pada Gambar 3.2.

$x += 2;$	kependekan dari $x = x + 2;$
$x -= 2;$	kependekan dari $x = x - 2;$
$x *= 2;$	kependekan dari $x = x * 2;$
$x /= 2;$	kependekan dari $x = x / 2;$
$x \% = 2;$	kependekan dari $x = x \% 2;$
$x << = 2;$	kependekan dari $x = x << 2;$
$x >> = 2;$	kependekan dari $x = x >> 2;$
$x \& = 2;$	kependekan dari $x = x \& 2;$
$x = 2;$	kependekan dari $x = x 2;$
$x \wedge = 2;$	kependekan dari $x = x \wedge 2;$

Gambar 3.2. Contoh Operator Pemendekan

3.5.8 Prioritas Operator Aritmatika

Tabel 3.5 memberikan penjelasan mengenai prioritas dari masing-masing operator. Operator yang mempunyai prioritas tinggi akan diutamakan dalam hal pengerjaan dibandingkan dengan operator yang memiliki prioritas lebih rendah.

Tabel 3.5. Prioritas Operator

Prioritas	Operator	Urutan Pengerjaan
Tertinggi	()	Dari kiri ke kanan
	! ++ -- + -	Dari kiri ke kanan ^{*)}
	* / %	Dari kiri ke kanan
	+ -	Dari kiri ke kanan ^{*)}
Terendah	= += -= *= /= %=	Dari kiri ke kanan

^{*)} Bentuk **unary** + dan **unary** – memiliki prioritas yang lebih tinggi daripada bentuk **binary** + dan **binary** –

3.6 Contoh Soal

Sesi berikut akan membahas beberapa contoh soal dan sekaligus pembahasannya.

1. Buatlah program untuk menghitung jumlah penghasilan bersih seorang pegawai. Penghasilan bersih berasal dari penghasilan kotor dikurangi pajak. Besarnya pajak adalah 5% dari penghasilan kotor. Penghasilan kotor

diperoleh dari gaji pokok, tunjangan dan gaji lembur yang dimasukkan oleh pengguna.

Pembahasan :

```
#include <stdio.h>
#define PAJAK 0.05
void main()
{
    int gaji_pokok,gaji_lembur,tunjangan;
    int gaji_kotor,gaji_bersih;

    printf("Masukkan gaji pokok :");
    scanf("%d",&gaji_pokok);
    printf("Masukkan gaji lembur :");
    scanf("%d",&gaji_lembur);
    printf("Masukkan tunjangan :");
    scanf("%d",&tunjangan);

    gaji_kotor=gaji_pokok+gaji_lembur+tunjangan;
    gaji_bersih =gaji_kotor - (gaji_kotor*PAJAK);

    printf("Gaji Bersih : %d",gaji_bersih);
}
```

Pada soal di atas, akan diinput dahulu semua variabel yang dibutuhkan yaitu gaji pokok, gaji lembur dan tunjangan. Pajak bernilai fiks 5% atau 0.05, sehingga dapat dideklarasikan sebagai konstanta. Setelah memasukkan tiga data yang diperlukan, maka dilakukan perhitungan gaji kotor dengan menjumlahkan ketiga data. Perhitungan gaji bersih diperoleh dari gaji kotor dikurangi pajak. Nilai pajak dalam rupiah dapat dihitung dari besarnya penghasilan kotor dikalikan pajak sebesar 5%.

2. Buatlah program untuk menghitung total perbelanjaan seseorang dan kembaliannya. Total perbelanjaan dihitung dari harga barang dikalikan dengan jumlah barang yang dibeli. Harga dan jumlah beli dimasukkan pengguna. Setelah total dihitung, total akan ditampilkan dan pengguna diminta memasukkan besarnya pembayaran. Lalu hitung dan cetaklah kembalian.

Pembahasan :

```
#include <stdio.h>
void main()
{
    int harga, jumlah, total, bayar, kembalian;

    printf("Masukkan harga barang :");
    scanf("%d",&harga);
    printf("Masukkan jumlah barang :");
    scanf("%d",&jumlah);

    total = bayar * jumlah;
    printf("Total belanja : %d",total);

    printf("Masukkan bayar :");
    scanf("%d",&bayar);

    kembalian = bayar - total;

    printf("Kembalian : %d",kembalian);
}
```

Pada soal di atas, hal pertama yang harus dilakukan adalah meminta masukan harga dan jumlah. Setelah harga dan jumlah dihitung dan disimpan pada variabel total, baru total bisa ditampilkan. Variabel bayar dimasukkan setelah total ditampilkan, agar pengguna bisa memasukkan nilai uang yang sesuai. Nilai kembalian akhirnya bisa dihitung dan ditampilkan.

3. Buatlah program untuk menghitung nilai akhir sebuah proposal kewirausahaan, jika diketahui parameter penilaian sebagai berikut. Terdapat dua orang penilai, dimana nilai akhir merupakan rata-rata dari keduanya.

Nama/Klp Pengusul	Nilai_Aspek							Total Nilai
	Integritas/Softskill			Penguasaan Bisnis				
	Kesesuaian Data / Info	Motivasi dan Kepercaaa	Keberanian Mengambil Resiko	Pasar	Teknis	Organisasi	Keuangan	

SKALA PENILAIAN

7 Sangat Baik
5 Baik
3 Kurang Baik
1 Tidak Baik

Penilai 1

Penilai 2

.....

Pembahasan :

Pada soal tersebut terdapat dua orang penilai, sehingga input disiapkan untuk setiap aspek penilaian. Langkah yang harus dilakukan adalah menerima input dari kedua penilai untuk setiap aspek penilaian. Kemudian merata-rata nilai setiap penilai dan terakhir merata-rata nilai rata-rata dari kedua penilai. Maka berikut ini adalah program penyelesaiannya :

```
#include <stdio.h>
void main()
{
    int  KD1, MK1, KMR1, pasar1, teknis1, organisasi1,
    keuangan1, KD2, MK2, KMR2, pasar2, teknis2, organisasi2,
    keuangan2;
    float rata1, rata2, rata;

    printf("Input Penilai 1\n");
    printf("Masukkan Nilai Kesesuaian Data :");
    scanf("%d",&KD1);
    printf("Masukkan Nilai Motivasi & Kepercayaan Diri
:");
    scanf("%d",&MK1);
    printf("Masukkan Nilai Keberanian Mengambil Resiko
:");
    scanf("%d",&KMR1);
    printf("Masukkan Nilai Pasar :");
    scanf("%d",&pasar1);
    printf("Masukkan Nilai Teknis :");
    scanf("%d",&teknis1);
    printf("Masukkan Nilai Organisasi :");
    scanf("%d",&organisasi1);
    printf("Masukkan Nilai Keuangan :");
    scanf("%d",&keuangan1);

    printf("Input Penilai 2\n");
    printf("Masukkan Nilai Kesesuaian Data :");
    scanf("%d",&KD2);
    printf("Masukkan Nilai Motivasi & Kepercayaan Diri
:");
    scanf("%d",&MK2);
    printf("Masukkan Nilai Keberanian Mengambil Resiko
:");
    scanf("%d",&KMR2);
```



```

printf("Masukkan Nilai Pasar :");
scanf("%d",&pasar2);
printf("Masukkan Nilai Teknis :");
scanf("%d",&teknis2);
printf("Masukkan Nilai Organisasi :");
scanf("%d",&organisasi2);
printf("Masukkan Nilai Keuangan :");
scanf("%d",&keuangan2);

rata1= (KD1 + MK1 + KMR1 + pasar1 + teknis1 +
organisasi1 + keuangan1)/7;
rata2= (KD2 + MK2 + KMR2 + pasar2 + teknis2 +
organisasi2 + keuangan2)/7;
rata=(rata1+rata2)/2;

printf("Nilai akhir = %.2f",rata);
}

```

3.7 Penutup

Kesimpulan

- Tipe data dalam Bahasa C terdiri dari tipe data bilangan bulat : char, short, int, long; tipe data bilangan *floating point* : float, double, long double; dan tipe void atau tanpa tipe data
- Variabel adalah suatu pengenal (identifier) yang digunakan untuk mewakili suatu nilai tertentu.
- Pemberian nama pada variabel mengikuti pemberian nama identifier, yaitu maksimal 31 karakter, diawali huruf atau underscore, dapat diikuti karakter huruf atau angka, tidak boleh mengandung spasi dan tidak mengandung karakter khusus seperti @, %.
- Konstanta merupakan nilai yang tetap sepanjang jalannya program. Deklarasi konstanta dapat menggunakan preprosesor define dengan langsung memberikan nilai.
- Perintah untuk membaca masukan pengguna adalah menggunakan `scanf()`. Penggunaan `scanf()` harus dilengkapi dengan string kontrol dan alamat variabel yang diinput. Alamat variabel diawali dengan penulisan tanda & di depan variabel.

- Terdapat lima operator aritmatika yaitu : penambahan (+), pengurangan (-), perkalian (*), pembagian (/), sisa hasil pembagian (%)
- Operator + dan – juga bisa menjadi operator unary dan binary
- Operator penurunan dan kenaikan ada dua yaitu ++ untuk menaikkan nilai sebesar satu, sedangkan -- untuk menurunkan nilai sebesar satu. Peletakkan tanda ++ dan -- di depan variabel bermakna mengubah nilai variabel dulu, baru melakukan operasi lainnya, sedangkan peletakkan di belakang berarti sebaliknya.
- Operator penugasan = digunakan untuk memberikan nilai operand di kanan pada operand di sebelah kiri tanda sama dengan (=)
- Operator relasional terdiri dari enam yaitu : ==, >=, >, <=, < dan !=
- Operator logika terdiri dari tiga yaitu : AND (&&), OR (||) dan NOT (!)
- Operator bitwise terdiri dari lima yaitu : |, &, ^, <<, >>
- Urutan pengerjaan operator dari tertinggi ke terendah adalah : () , ! ++ -- + -, * / %, + -, = += -= *= /= %=

Latihan

1. Buatlah program untuk menghitung luas lingkaran. Gunakan konstanta untuk menyatakan nilai PI. Mintalah pengguna memasukkan jari-jari lingkaran!
2. Buatlah program untuk menghitung sisi miring suatu segitiga siku-siku, jika pengguna diminta untuk menginput nilai alas dan tingginya! Perhitungan akar y bisa menggunakan fungsi `sqrt(y)`
3. Apakah keluaran dari program berikut ini :

```
#include <stdio.h>

void main()
{
    int a=5, b=1, c=-2,d;
    a +=-4;
    d = c-- *a;
    b *= (++c -d);
    printf("a : %d, b : %d, c : %d, d : %d",a,b,c,d);
}
```

4. Diberikan program seperti berikut :

```
#include <stdio.h>

void main()
{
    int a=5, b=2;
    float x,y;
    x = a/b;
    y = 9/a;
    printf("x : %.2f, y : %.2f",x,y);
}
```

- a. Jelaskan mengapa nilai x menjadi 2.00 dan y menjadi 1.00 setelah program dieksekusi!
 - b. Modifikasi bagian dari program, agar nilai x sesuai dengan 5 dibagi 2 yaitu 2.50 dan nilai y sesuai dengan 9 dibagi 5 yaitu 1.8!
5. Sebuah perusahaan kurir menetapkan tarif pengiriman sebesar 15000 per kilo untuk 1 kilo pertama, dan selanjutnya sebesar 10000/kg. Buatlah program untuk menerima input berupa total barang yang dikirim (dalam kilo) dan harga pengiriman.
6. Sebuah toko menjual kopi dalam kemasan tas. Buatlah program yang menerima input berupa berat tas (dalam kilo), jumlah pembelian tas, harga kopi per kilo. Keluarkan output berupa total pembayaran kopi, apabila diketahui pajak adalah sebesar 10% dari total belanja.
7. Buatlah program untuk melakukan konversi nilai detik menjadi jam, menit, detik. Misalnya user memberikan input detik sebesar 7300, maka keluaran program adalah 2 jam, 1 menit 40 detik.
8. Diberikan program berikut ini :

```
#include <stdio.h>
void main()
{
    int a=1, b=5,c;

    c=a==b;
    printf("\n\n");
    printf("a = %d\n", a);
    printf("b = %d\n", b);
}
```

```
printf("c = %d\n", c);  
}
```

- a. Berapakah output nilai c? Mengapa outputnya demikian?
 - b. Bagaimana jika pernyataan `c=a==b` diganti menjadi `c=a=b`? Apakah outputnya berbeda? Mengapa?
9. Diberikan program seperti berikut :

```
#include <stdio.h>  
  
void main()  
{  
    int a=7, b=5, c;  
  
    c = a++ % --b;  
  
    printf("\n\n");  
    printf("a = %d\n", a);  
    printf("b = %d\n", b);  
    printf("c = %d\n", c);  
}
```

- a. Jelaskanlah mengapa output dari `a = 8`, `b = 4` dan `c=3`!
 - b. Jika pernyataan `c = a++ % --b` diganti dengan `c= ++a % --b`, maka berapakah outputnya? Jelaskan mengapa demikian? Lalu bagaimana jika diganti dengan `c= a++ % b--` atau `c = ++a % b--`?
10. Sebuah toko kopi menjual kopi dalam kemasan 2kg saja, yaitu seharga 50000 per kemasan. Ketika customer melakukan pemesanan, maka kopi yang dipesan akan dikirim dalam bentuk box. Toko memiliki 3 jenis box, yaitu box besar mampu menampung 20 kemasan, box sedang mampu menampung 10 kemasan, dan box kecil mampu menampung 5 kemasan. Harga box besar adalah 10000, box sedang 7500, dan box kecil adalah 5000. Dalam melakukan packing, toko menggunakan aturan bahwa box besar dan box sedang tidak boleh memiliki ruang kosong (harus penuh), sedangkan box kecil boleh saja kosong. Toko ingin meminimalkan penggunaan box, dengan cara melakukan pemilihan box mulai dari box terbesar dulu baru medium dan selanjutnya box terkecil. Misalnya jika customer memesan 52 kemasan kopi, maka akan digunakan 2 box besar, 1 box sedang dan 1 box kecil. Buatlah program untuk menerima input berupa total kemasan yang dipesan oleh

seorang customer dan menentukan berapa jumlah masing-masing tas dan biaya total yang harus dibayar, yaitu harga box dan harga kopi yang dibeli.

Umpan Balik

Setelah menyelesaikan bab ini, mahasiswa diharapkan mengetahui berbagai tipe data dan operator yang didukung oleh Bahasa C. Ketika diberikan permasalahan, mahasiswa mampu menentukan data apa saja yang akan disimpan dan apa tipe datanya. Selain itu mahasiswa diharapkan mampu menuliskan suatu operasi yang melibatkan data sebagai operan dengan operator yang didukung oleh data tersebut. Mahasiswa juga diharapkan mampu menampilkan data dengan berbagai tipe ke dalam layar. Pemahaman akan bab ini sangat dibutuhkan agar mahasiswa bisa melanjutkan pada bab selanjutnya.

BAB IV

PENGAMBILAN KEPUTUSAN

Standar Kompetensi

Mahasiswa diharapkan mampu menerjemahkan permasalahan dalam dunia nyata ke dalam algoritma dan mengimplementasikannya ke dalam Bahasa C setelah mempelajari mata kuliah ini.

Kompetensi Dasar

Setelah mempelajari materi ini, mahasiswa diharapkan mampu memahami dan menerapkan percabangan dalam Bahasa C.

Indikator Pencapaian Kompetensi

Adapun indikator hasil belajar adalah sebagai berikut :

1. Mahasiswa mampu memahami bahwa kasus yang diberikan adalah kasus percabangan atau bukan.
2. Mahasiswa mampu menggunakan perintah if dalam menyelesaikan kasus percabangan yang diberikan.
3. Mahasiswa mampu menggunakan perintah switch-case dalam menyelesaikan kasus percabangan yang diberikan.
4. Mahasiswa mampu memahami perbedaan perintah if dan switch-case.
5. Mampu menggunakan percabangan bersarang dalam sebuah program C.

MATERI

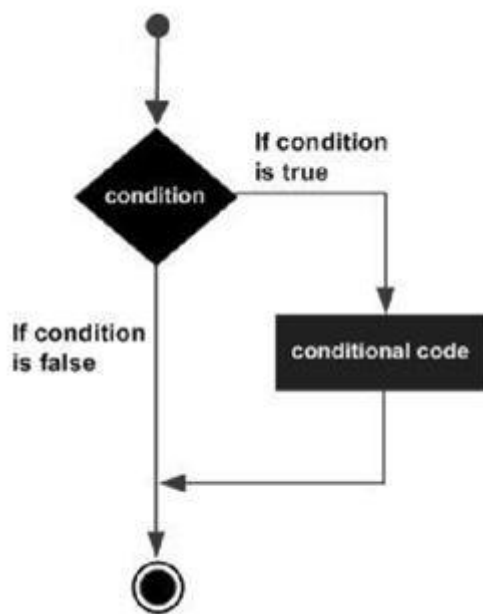
4.1 Percabangan

Percabangan adalah suatu kondisi dimana terdapat satu atau beberapa kondisi yang harus dievaluasi oleh program kemudian diikuti oleh pernyataan atau adanya pernyataan yang akan dieksekusi apabila kondisi dinyatakan benar atau pernyataan lainnya yang akan dieksekusi apabila kondisi dinyatakan salah. Dalam

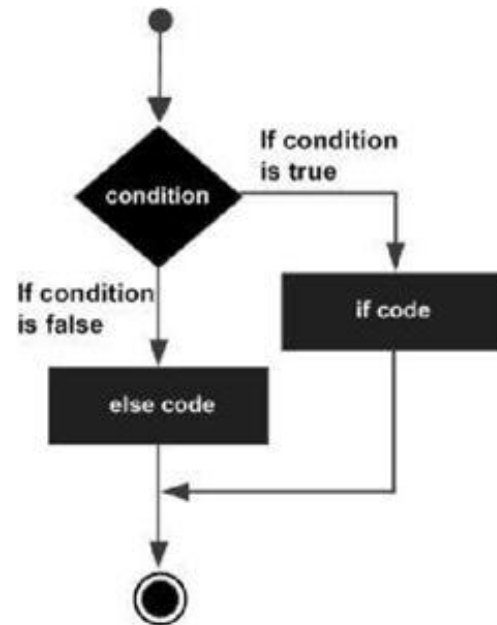
pemrograman C, nilai non-null atau bukan nol akan dianggap sebagai `true`, sedangkan nilai null atau nol akan dianggap sebagai `false`.

4.2 Perintah `if`

Perintah `if` merupakan perintah dalam bahasa pemrograman C yang digunakan untuk mengevaluasi dan mengerjakan percabangan. Perintah `if` yang paling sederhana hanya terdiri dari satu blok `if` saja, tanpa `else`. Gambar 4.1 menunjukkan diagram alir percabangan.



Gambar 4.1a. Diagram alir `if` tanpa `else`
(Tutorials Point, 2014)



Gambar 4.1b. Diagram alir `if-else`
(Tutorials Point, 2014)

Pada Gambar 4.1a, terdapat pengecekan kondisi dan pernyataan yang dilakukan apabila kondisi bernilai benar, dan apabila kondisi bernilai salah, maka tidak ada yang pernyataan khusus yang dilakukan, melainkan hanya melanjutkan blok program. Gambar 4.1b menunjukkan bahwa setelah adanya pengecekan kondisi, maka akan ada pernyataan khusus yang dilakukan apabila kondisi benar, dan apabila kondisi salah maka akan ada pernyataan lain yang dilakukan. Pernyataan lain tersebut dituangkan ke dalam `else`.

4.2.1 Perintah `if` sederhana

Perintah if sederhana hanya menguji sebuah kondisi dan melakukan sebuah aksi apabila kondisi benar dan tidak melakukan aksi apapun jika kondisi salah. Struktur if sederhana adalah sebagai berikut :

```
if (pengecekan kondisi)
{
    Pernyataan yang akan dieksekusi
}
```

Pengecekan kondisi di dalam if, akan mengandung operator perbandingan. Misalnya mengecek apakah seseorang akan mengulang mata kuliah. Mengulang mata kuliah apabila nilainya di bawah 70. Maka kondisi yang dicek adalah nilai seseorang. Adapun operator yang digunakan adalah <. Potongan source code dari kasus tersebut adalah :

```
int nilai;
printf("Masukkan nilai : "); scanf("%d",&nilai);
if (nilai<70)
{
    printf("Anda mengulang mata kuliah\n");
}
```

Keluaran dari program di atas akan sangat tergantung pada nilai yang dimasukkan oleh pengguna. Apabila pengguna menginput 75, maka program tidak akan memberikan hasil apapun karena nilai 70 ke atas tidak dicakup dalam blok if. Sedangkan apabila nilai diinput di bawah 70, misalnya 50, maka program akan mengeluarkan pesan “Anda mengulang mata kuliah”

4.2.2 Perintah if-else

Dalam suatu percabangan, kadang terdapat dua alternatif yang dijalankan, yang mana keduanya bergantung pada nilai kebenaran dari kondisi yang dicek. Struktur if-else adalah sebagai berikut :

```
if (pengecekan kondisi)
{
    Pernyataan 1 yang akan dieksekusi
}
else
{
    Pernyataan 2 yang akan dieksekusi
}
```


Jika pengecekan kondisi menghasilkan nilai benar, maka pernyataan 1 akan dieksekusi dan program tidak akan melaksanakan blok else. Sebaliknya apabila pengecekan kondisi menghasilkan nilai salah, maka pernyataan 1 tidak akan dieksekusi dan program akan segera menuju blok else dan mengeksekusi pernyataan 2. Contoh kasus if-else yaitu, kelulusan mahasiswa akan ditentukan berdasarkan nilainya. Jika nilainya 70 ke atas maka lulus, sedangkan jika di bawah 70 akan dinyatakan tidak lulus. Potongan program untuk kasus tersebut adalah :

```
int nilai;
printf("Masukkan nilai : "); scanf("%d",&nilai);
if(nilai>=70)
{
    printf("Anda lulus");
}
else
{
    printf("Anda tidak lulus");
}
```

Pada program di atas, pengecekan kondisi dilakukan terhadap nilai, sedangkan operator yang digunakan adalah >=, karena nilai lebih dari atau sama dengan 70 dinyatakan lulus. Keluaran dari program akan tergantung pada masukkan nilai. Jika nilai melebihi diinput 80 (70 ke atas), maka program akan mengeksekusi perintah dalam blok if yaitu "Anda lulus", sedangkan jika dimasukkan 60 (di bawah 70), maka program akan menuju blok else dan mengeksekusi perintah di dalamnya dan menghasilkan keluaran "Anda tidak lulus."

4.2.3 Perintah if-else if

Percabangan yang cukup kompleks, memiliki alternatif yang banyak. Untuk kasus dengan alternatif lebih daripada dua, bisa digunakan if-else if. Struktur if-else if adalah sebagai berikut :

```
if(pengecekan kondisi 1)
{
    Pernyataan 1 yang akan dieksekusi
}
else if(pengecekan kondisi 2)
{
    Pernyataan 2 yang akan dieksekusi
}
```

```
.....  
else  
{  
    Pernyataan n yang akan dieksekusi  
}
```

Pada sebuah if-else if, kadangkala blok else tidak ada. Hal itu tergantung pada alternatif dan syaratnya. Contohnya dapat dilihat pada kasus berikut. Sebuah bilangan bulat memiliki status yaitu :

Bulat positif : bilangan lebih daripada 0

Nol : bilangan sama dengan 0

Bulat negatif : bilangan kurang daripada 0

Jika dilihat pada soal di atas, terdapat tiga buah alternatif, dimana pengecekannya dilakukan berdasarkan bilangannya. Alternatif pertama akan masuk ke dalam blok if, sedangkan sisanya akan masuk ke dalam blok else. Berikut ini adalah potongan listing programnya :

```
int bilangan;  
printf("Masukkan bilangan : "); scanf("%d",&bilangan);  
if(bilangan > 0)  
{  
    printf("Bulat Positif");  
}  
else if(bilangan == 0)  
{  
    printf("Nol");  
}  
else if(bilangan < 0)  
{  
    printf("Bulat Negatif");  
}
```

Pada kasus di atas, jika bilangan yang dimasukkan 5, maka blok if akan bernilai true, sehingga akan tampil “Bulat Positif” ke dalam layar dan program tidak akan mengecek lagi blok else if yang lainnya, sedangkan jika yang diinput adalah 0, maka blok if akan bernilai false, sehingga akan dilakukan pengecekan pada blok else if pertama dan menampilkan teks “Nol” pada layar. Sedangkan jika dimasukkan bilangan -3 maka, blok if dan else if yang pertama akan false dan yang dicek adalah blok else if kedua dan bernilai true, maka dicetaklah “Bulat Negatif” ke dalam layar.

4.2.4 Perintah if dengan Operator Logika

Kondisi yang dicek di dalam percabangan bisa saja lebih dari satu pernyataan. Misalnya untuk menentukan kelulusan digunakan dua buah nilai yaitu nilai praktikum lebih daripada 70 dan nilai teori lebih daripada 60. Dari syarat tersebut terlihat ada dua syarat dalam kondisi yang dicek. Untuk menggabungkan beberapa syarat bisa digunakan operator logika. Jawaban untuk contoh tersebut adalah sebagai berikut :

```
int nilai_praktikum, nilai_teoris;
printf("Masukkan nilai teori : ");
scanf("%d", &nilai_teoris);
printf("Masukkan nilai praktikum: ");
scanf("%d", &nilai_praktikum);

if(nilai_praktikum>70 && nilai_teoris>=60)
{
    printf("Anda lulus");
}
else
{
    printf("Anda tidak lulus");
}
```

Pada soal, pengecekan kedua syarat menggunakan operator AND (&&) karena syarat kelulusan adalah mutlak keduanya yaitu praktikum melebihi 70 dan teori melebihi 60.

Contoh operator logika yang lainnya misalnya adalah, seseorang akan mendapatkan potongan harga sebesar 5000, jika berbelanja barang dengan jumlah lebih dari 10 item atau berbelanja dengan total melebihi 50000. Penyelesaian kasus tersebut ditampilkan dalam potongan listing program berikut ini :

```
int jumlah, total, potongan=0;
printf("Masukkan jumlah : ");
scanf("%d", &jumlah);
printf("Masukkan total : ");
scanf("%d", &total);

if(jumlah > 10 || total >= 50000)
{
    potongan=5000;
}
```

Pada contoh di atas syarat potongan ada dua yaitu jumlah atau total. Operator logika yang digunakan adalah atau karena cukup salah satu syarat saja dipenuhi, maka pembeli berhak mendapatkan potongan. Jika salah satu dari kedua syarat tidak terpenuhi, maka potongan bernilai nol, sesuai dengan nilai inisial yang diberikan pada program, sehingga tidak diperlukan blok else.

Pertanyaan

1. Bagaimana dengan kasus bahwa pembeli yang diberikan diskon adalah pembeli yang berusia 20 dan 21. Apakah operator logika yang tepat?
2. Diberikan source berikut :

```
#include <stdio.h>
int main()
{
    char nama[12];
    int nilai;
    printf("Program Mengetes Kondisi \n");
    printf("===== \n");
    printf("Masukkan Nilai=");scanf("%d",&nilai);
    if (nilai>=60)
    {
        printf("LULUS BAIK \n");
    }
    if ((nilai>=0) && (nilai<=59))
    {
        printf("TIDAK LULUS \n");
    }
}
```

Apakah perbedaannya jika pernyataan `if ((nilai>=0) && (nilai<=59))` diganti menjadi `else if ((nilai>=0) && (nilai<=59))` ? Jelaskan manakah yang lebih efisien!

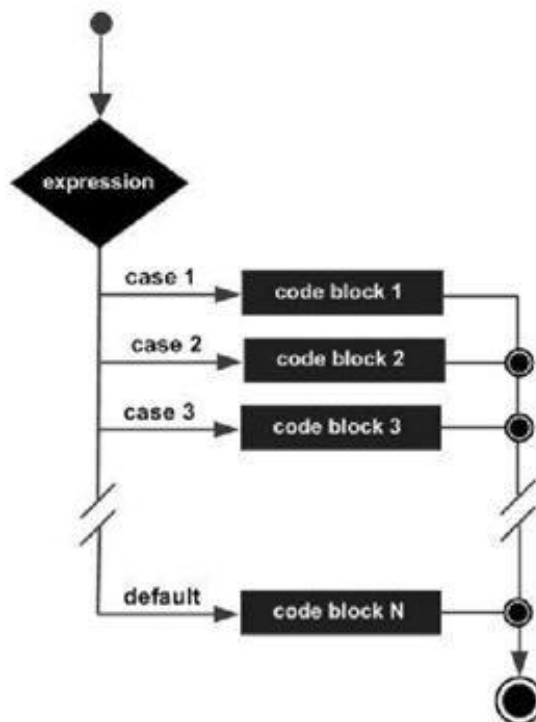
4.2.5 Perintah switch-case

Perintah switch-case pada dasarnya sama seperti perintah if, yaitu untuk menyelesaikan percabangan. Perintah switch-case pada bahasa C memiliki cakupan kasus yang lebih sempit dibandingkan dengan perintah if. Switch-case hanya bekerja pada tipe data ordinal. Tipe data ordinal meliputi tipe bilangan

bulat dan char. Adapun diagram alir untuk switch-case ditunjukkan pada Gambar 4.2. Sedangkan struktur switch-case adalah sebagai berikut :

```
switch (ekspresi)
{
    case constant_1 :
        statements_1; break;
    case constant_2 :
        statements_2; break;
    ...
    case constant_N :
        statements_N; break;
    default :
        statements;
}
```

Ekspresi yang dicek di dalam switch adalah nama variabel yang menyimpan data yang dicek di dalam kondisi. Sedangkan `constant_1`, `constant_2` dan seterusnya adalah nilai kemungkinan dari variabel tersebut. `Statements_1` sampai `statements_N` adalah pernyataan yang akan dieksekusi jika nilai tertentu terpenuhi. Kemungkinan nilai selain yang didefinisikan pada setiap `constant` dinyatakan dalam `default`. Setiap kemungkinan nilai variabel harus diakhiri oleh `break`, kecuali pada `default`.



Gambar 4.2. Diagram Alir Switch-Case (Tutorials Point, 2014)

Contoh kasus penggunaan switch-case misalnya adalah status nilai seseorang berdasarkan nilai hurufnya yaitu :

A : Istimewa

B : Baik

C : Cukup

D : Kurang

E : Kurang Sekali

Maka jawaban dari kasus di atas dapat dikerjakan dengan if maupun switch-case, namun akan terlihat lebih singkat dan terstruktur dengan switch-case.

```
char nilai;
printf("Masukkan nilai : ");scanf("%c",&nilai);
switch(nilai)
{
    case 'A' : printf("Istimewa"); break;
    case 'B' : printf("Baik"); break;
    case 'C' : printf("Cukup"); break;
    case 'D' : printf("Kurang"); break;
    case 'E' : printf("Sangat Kurang"); break;
    default : printf("Input salah");
}
```

Keluaran dari program di atas akan tergantung pada masukan nilai, dimana nilai berupa karakter. Perintah switch akan memuat nama variabel yang akan dicek yaitu nilai, sementara untuk setiap case, akan terdapat nilai-nilai yang mungkin yaitu A sampai E. Jika diinput nilai C, maka program akan mengecek ke case A dan memberikan nilai false, sehingga program akan melanjutkan ke case berikutnya yaitu B dan kembali lagi memberikan nilai false, sampai akhirnya ditemukan case C dan akan mengeksekusi perintah di dalam case C dan langsung keluar dari switch-case karena ditemukan perintah break. Default hanya akan dipanggil jika nilai dimasukkan bukan salah satu dari A, B, C, D ataupun E.

Pertanyaan

Bagaimana jika perintah break dalam switch-case dihilangkan? Apakah yang terjadi, cobalah memberikan input nilai C dan lihatlah apa yang terjadi!

4.3 Percabangan bersarang

Percabangan seringkali memiliki percabangan lain di dalamnya. Percabangan seperti ini disebut sebagai percabangan bersarang. Contoh kasus percabangan bersarang misalnya perhitungan biaya pengiriman dokumen. Adapun aturan biaya pengiriman adalah sebagai berikut :

Lama Pengiriman (hari)	Jenis Dokumen	Biaya/kg
1	Golongan 1	30000
	Golongan 2	20000
2-3	Golongan 1	20000
	Golongan 2	15000
Lebih dari 3	Semua golongan	5000

Dalam kasus di atas terdapat 3 alternatif untuk lama pengiriman, sehingga dapat menggunakan if-else if, sedangkan untuk setiap waktu pengiriman terdapat dua alternatif berdasarkan jenis dokumennya, kecuali untuk pengiriman yang lebih daripada tiga hari. Letak perulangan bersarang adalah pada waktu 1 dan 2 sampai 3 hari. Adapun penyelesaian dari kasus tersebut ditunjukkan pada potongan program berikut :

```
int lama,jenis, berat,biaya;
printf("Masukkan lama kirim : ");scanf("%d",&lama);
printf("Masukkan jenis : ");scanf("%d",&jenis);
printf("Masukkan berat : ");scanf("%d",&berat);
if(lama==1)
{
    if(golongan==1)
    {
        biaya=berat*30000;
    }
    else if(golongan==2)
    {
        biaya=berat*20000;
    }
}
else if(lama==2)
{
    if(golongan==1)
    {
        biaya=berat*20000;
    }
    else if(golongan==2)
    {
```

```

        biaya=berat*15000;
    }

}
else
{
    biaya=berat*5000;
}

```

4.4 Contoh Soal

Sesi berikut akan membahas beberapa contoh soal yang berkaitan dengan percabangan dan sekaligus pembahasannya.

1. Buatlah program untuk menentukan staus kelulusan seseorang. Status kelulusan seseorang ada tiga yaitu :

Lulus : nilai 70-100

Lulus bersyarat : 50-69

Tidak lulus : 0-49

Pembahasan

Dalam contoh di atas, terdapat tiga alternatif kelulusan berdasarkan nilainya. Alternatif pertama dapat dimasukkan ke dalam blok if, sedangkan alternatif lainnya dimasukkan ke dalam blok else if. Berikut ini adalah pembahasan kasus di atas

```

#include<stdio.h>
void main()
{
    int nilai;
    printf("Masukkan nilai : "); scanf("%d",&nilai);
    if(nilai>=70 && nilai<=100)
    {
        printf("Anda lulus");
    }
    else if(nilai>=50 && nilai<69)
    {
        printf("Anda lulus bersyarat");
    }
    else if(nilai>=0 && nilai <=49)
    {
        printf("Anda tidak lulus");
    }
}

```


Pengecekan data berinterval pada if dapat menggunakan tanda &&. Pada contoh di atas, nilai ≥ 70 && nilai ≤ 100 akan memberikan nilai true jika nilai yang dimasukkan adalah antara 70 sampai 100.

2. Buatlah program untuk menghitung total uang yang harus dibayarkan oleh seorang konsumen pada suatu minimarket. Konsumen akan berhak mendapatkan diskon dengan aturan sebagai berikut :

Total belanja di atas 500000, mendapatkan diskon 10%,

Total belanja di atas 100000-500000, mendapatkan diskon 5%,

Total belanja di bawah 500000, tidak mendapatkan diskon

Pembahasan

Input dari program di atas adalah total belanja. Total belanja ini kemudian akan dicek apakah memenuhi ketiga kriteria yang telah ditentukan untuk mengetahui diskonnya. Setelah dikurangi diskon, total uang yang harus dibayar selanjutnya dicetak ke dalam layar. Pengecekan syarat melibatkan data interval, sehingga digunakan operator logika && untuk menggabungkan keduanya.

```
#include<stdio.h>
void main()
{
    int total, bayar;
    printf("Masukkan total belanja : ");
    scanf("%d",&total);
    if(total > 500000)
    {
        bayar = total - (total*0.1);
    }
    else if(total >=100000 && total <=500000)
    {
        bayar = total - (total*0.05);
    }
    else
    {
        bayar=total;
    }

    printf("Bayar = %d",bayar);
}
```

3. Buatlah validasi untuk input pada kasus penilaian PMW pada studi kasus 3.3, dimana nilai yang diperbolehkan adalah antara 1-7 untuk setiap aspek penilaian.

Pembahasan

Dalam kasus ini program harus mampu membatasi input 1-7 saja untuk setiap aspek. Oleh sebab itu, program bisa menggunakan percabangan dengan pengecekan range pada setiap input. Program akan memberikan pesan kesalahan apabila input tidak dalam range.

```
#include <stdio.h>
void main()
{
    int KD1, MK1, KMR1, pasar1, teknis1, oragnisasil,
    keuangan1, KD2, MK2, KMR2, pasar2, teknis2,
    oragnisasi2, keuangan2;
    float rata1, rata2, rata;

    printf("Input Penilai 1\n");
    printf("Masukkan Nilai Kesesuaian Data :");
    scanf("%d",&KD1);
    if((KD1<1 || KD1>7) || KD1%2==0) {
        printf("Nilai yang diperbolehkan hanya 1-7);
        return;
    }
    printf("Masukkan Nilai Motivasi & Kepercayaan Diri
    :");
    scanf("%d",&MK1);
    if(MK1<1 || MK1>7 || MK1%2==0) {
        printf("Nilai yang diperbolehkan hanya 1-7);
        return;
    }

    printf("Masukkan Nilai Keberanian Mengambil Resiko
    :");
    scanf("%d",&KMR1);
    if(KMR1<1 || KMR1>7 || KMR1%2==0) {
        printf("Nilai yang diperbolehkan hanya 1-7);
        return;
    }

    printf("Masukkan Nilai Pasar :");
    scanf("%d",&pasar1);
    if(pasar1<1 || pasar1>7 || pasar1%2==0) {
        printf("Nilai yang diperbolehkan hanya 1-7);
        return;
    }

    printf("Masukkan Nilai Teknis :");
```

```

scanf("%d",& teknis1);
if(teknis1<1 || teknis1>7 || teknis1%2==0) {
    printf("Nilai yang diperbolehkan hanya 1-7);
    return;
}

printf("Masukkan Nilai Organisasi :");
scanf("%d",& organsasil);
if(organsasil<1 || organsasil>7 || organsasil%2==0)
{
    printf("Nilai yang diperbolehkan hanya 1-7);
    return;
}

printf("Masukkan Nilai Keuangan :");
scanf("%d",& keuangan1);
if(keuangan1<1 || keuangan1>7 || keuangan1%2==0) {
    printf("Nilai yang diperbolehkan hanya 1-7);
    return;
}

printf("Input Penilai 2\n");
printf("Masukkan Nilai Kesesuaian Data :");
scanf("%d",&KD2);
if(KD2<1 || KD2>7 || keuangan1%2==0) {
    printf("Nilai yang diperbolehkan hanya 1-7);
    return;
}

printf("Masukkan Nilai Motivasi & Kepercayaan Diri
:");
scanf("%d",& MK2);
if(MK2<1 || MK2>7 || MK2%2==0) {
    printf("Nilai yang diperbolehkan hanya 1-7);
    return;
}

printf("Masukkan Nilai Keberanian Mengambil Resiko
:");
scanf("%d",& KMR2);
if(KMR2<1 || KMR2>7 || KMR2%2==0) {
    printf("Nilai yang diperbolehkan hanya 1-7);
    return;
}

printf("Masukkan Nilai Pasar :");
scanf("%d",& pasar2);
if(pasar2<1 || pasar2>7 || pasar2%2==0) {
    printf("Nilai yang diperbolehkan hanya 1-7);
    return;
}

```

```

printf("Masukkan Nilai Teknis :");
scanf("%d",& teknis2);
if(teknis2<1 || teknis2>7 || teknis2%2==0) {
    printf("Nilai yang diperbolehkan hanya 1-7);
    return;
}

printf("Masukkan Nilai Organisasi :");
scanf("%d",& organisasi2);
if(organisasi2<1 || organisasi2>7 || organisasi2%2==0)
{
    printf("Nilai yang diperbolehkan hanya 1-7);
    return;
}

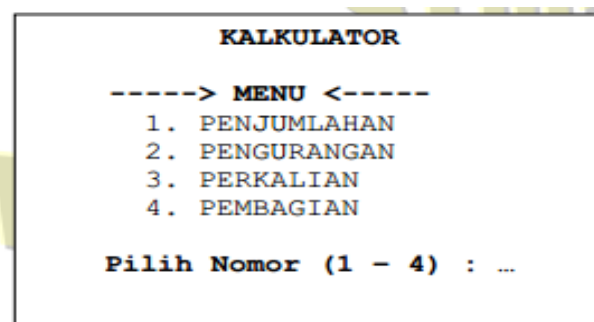
printf("Masukkan Nilai Keuangan :");
scanf("%d",& keuangan2);
if(keuangan2<1 || keuangan2>7 || keuangan2%2==0) {
    printf("Nilai yang diperbolehkan hanya 1-7);
    return;
}

rata1= (KD1 + MK1 + KMR1 + pasar1 + teknis1 +
oragnisasi1 + keuangan1)/7;
rata2= (KD2 + MK2 + KMR2 + pasar2 + teknis2 +
oragnisasi2 + keuangan2)/7;
rata=(rata1+rata2)/2;

printf("Niai akhir = %.2f",rata);
}

```

4. Buatlah program untuk membuat tampilan sebagai berikut :



Gambar 4.3. Keluaran Soal Latihan Kalkulator

Pada contoh di atas terdapat empat pilihan. Pengguna diminta memasukkan angka 1 sampai 4, sehingga bisa digunakan switch-case untuk operasi di dalam setiap pilihannya. Sebelum melakukan operasi dari setiap pilihan, pengguna diminta untuk menginputkan dua buah bilangan sebagai operand

dan kemudian menampilkan hasilnya setelah operasi dipilih. Berikut ini adalah listing program kasus di atas.

```
int operasi;
float bil1,bil2,hasil;
printf("\t\t KALKULATOR\n\n");
printf("\t----->MENU<-----\n");
printf("\t 1. Penjumlahan\n");
printf("\t 2. Pengurangan\n");
printf("\t 3. Perkalian\n");
printf("\t 4. Pembagian\n\n");
printf("\t Masukkan pilihan Anda : ");
scanf("%d",&operasi);
printf("\t Masukkan bilangan 1 : ");scanf("%f",&bil1);
printf("\t Masukkan bilangan 2 : ");scanf("%f",&bil2);
switch(operasi)
{
    case 1 : hasil=bil1+bil2;
             printf("\t Hasil %.2f",hasil);break;
    case 2 : hasil=bil1-bil2;
             printf("\t Hasil %.2f",hasil);break;
    case 3 : hasil=bil1*bil2;
             printf("\t Hasil %.2f",hasil);break;
    case 4 : hasil=bil1/bil2;
             printf("\t Hasil %.2f",hasil);break;
    default : printf("Pilihan salah");
}
```

4.5 Penutup

Kesimpulan

- Percabangan adalah suatu kondisi dimana terdapat satu atau beberapa kondisi yang harus dievaluasi oleh program kemudian diikuti oleh pernyataan atau adanya pernyataan yang akan dieksekusi apabila kondisi dinyatakan benar atau pernyataan lainnya yang akan dieksekusi apabila kondisi dinyatakan salah.
- Sebuah percabangan bisa dikerjakan dengan perintah `if` atau `switch-case`
- Setiap perintah `if` selalu memiliki kondisi yang dicek

- Perintah if sederhana hanya terdiri dari satu blok pengecekan kondisi, yang mana jika kondisi benar maka akan ada aksi yang dilakukan, namun jika salah tidak akan ada aksi
- Varian lain dari if sederhana adalah if-else dan if-else if
- Perintah if-else digunakan jika hanya ada dua alternatif yaitu kondisi benar dan salah
- Perintah if-else if digunakan jika terdapat lebih dari dua alternatif, dimana alternatif pertama masuk ke dalam if, alternatif ke dua sampai n-1 masuk ke dalam else if, sedangkan alternatif terakhir dapat dimasukkan ke dalam blok else-if atau else saja
- Pengecekan kondisi di dalam perintah if kadang bisa lebih dari satu kondisi, sehingga semua kondisi bisa dirangkai dengan operator logika AND (&&), OR (||) atau NOT (!)
- Perintah switch-case hanya dapat digunakan pada percabangan dengan tipe data char dan integer
- Switch akan diikuti oleh nama variabel yang dicek, sedangkan case diikuti oleh kemungkinan nilai variabel
- Setiap statement yang dieksekusi dalam case harus diakhiri break
- Kemungkinan yang tidak terdapat di dalam case akan ditampung pada default
- Sebuah percabangan bisa terdiri dari cabang-cabang lainnya.

Latihan

1. Buatlah sebuah program untuk menentukan nama hari berdasarkan inputan, contohnya sebagai berikut:
(1 = Senin)
(2 = Selasa)
(3 = Rabu)
(4 = Kamis)
(5 = Jumat)
(6 = Sabtu)
(7 = Minggu)
2. Buatlah program untuk menentukan apakah akar-akar persamaan kuadrat imajiner, kembar atau dua akar nyata yang berbeda. Akar persamaan kuadrat dihitung dengan rumus $x_1 = \frac{-b+\sqrt{D}}{2a}$, $x_2 = \frac{-b-\sqrt{D}}{2a}$, dengan $D = b^2 - 4ac$. Adapun jenis akar persamaan ditentukan dengan aturan berikut :
Imajiner : $D < 0$
Kembar : $D = 0$
Dua akar nyata yang berbeda : $D > 0$
Nilai a, b dan c akan dimasukkan oleh pengguna. Sedangkan output yang diharapkan adalah jenis akar. Cetaklah x1 dan x2 jika akarnya bukan imajiner
3. Buatlah program untuk menghitung total gaji yang didapatkan oleh seorang pegawai. Gaji ditentukan berdasarkan gaji pokok, tunjangan suami/istri dan tunjangan anak. Tunjangan suami/istri sebesar 10% dari gaji pokok jika status sudah menikah. Sedangkan tunjangan anak adalah 5% dari gaji pokok per anak, dengan maksimal jumlah anak yang ditanggung adalah 3. Gaji pokok, status dan jumlah anak akan dimasukkan pengguna. Total gaji masih akan dikurangi lagi dengan pajak yaitu, pajak sebesar 10% untuk total gaji di atas 5000000, sedangkan gaji antara 2000000 sampai 5000000 dikenai pajak sebesar 5%. Hitunglah total gaji setelah dikenai pajak!
4. Apakah yang terjadi jika operator AND (&&) contoh soal 1 diganti menjadi operator OR (||)? Jelaskan jawaban Anda!

5. Modifikasilah contoh soal 1 agar ketika diinputkan nilai di atas 100 dan di bawah 0, mengeluarkan pesan data yang dimasukkan salah.
6. Ubahlah program pada contoh soal 4 agar menggunakan perintah if!
7. Buatlah program untuk melakukan pengecekan username dan password yang dimasukkan pengguna ke dalam sistem. Username yang benar adalah 123, sedangkan password yang benar adalah 321. Berikan pesan “Selamat” jika login benar, sedangkan “Anda gagal” jika login salah.
8. Buatlah program untuk menentukan apakah seseorang akan diberikan sertifikata atau tidak. Seseorang diberikan sertifikat berdasarkan nilainya. Nilai yang diberikan sertifikat yaitu nilai A, nilai B.
9. Buatlah program untuk mengetahui apakah suatu bilangan yang diinputkan adalah kelipatan 3 atau tidak!

Umpan Balik

Setelah menyelesaikan bab ini, mahasiswa diharapkan mampu menyelesaikan kasus percabangan yang diberikan, serta memilih bentuk perulangan yang paling sesuai dengan kasus. Mahasiswa diharapkan mengerti perbedaan antara kasus yang merupakan percabangan atau kasus dapat diselesaikan dengan operasi sederhana tanpa percabangan. Selain itu mahasiswa diharapkan mampu menggunakan operator logika sesuai dengan kebutuhan. Pemahaman akan bab ini sangat penting, karena tanpa memahami bab ini, mahasiswa akan mengalami kesulitan untuk mempelajari bab selanjutnya.

BAB V

PERULANGAN PROSES

Standar Kompetensi

Mahasiswa diharapkan mampu menerjemahkan permasalahan dalam dunia nyata ke dalam algoritma dan mengimplementasikannya ke dalam Bahasa C setelah mempelajari mata kuliah ini.

Kompetensi Dasar

Setelah mempelajari materi ini, mahasiswa diharapkan mampu memahami dan menerapkan perulangan dalam Bahasa C untuk menyelesaikan kasus tertentu.

Indikator Pencapaian Kompetensi

Adapun indikator hasil belajar adalah sebagai berikut :

1. Mahasiswa mampu memahami bahwa kasus yang diberikan adalah kasus perulangan atau bukan.
2. Mahasiswa mampu menggunakan perulangan for untuk menyelesaikan suatu kasus.
3. Mahasiswa mampu menggunakan perulangan while untuk menyelesaikan suatu kasus.
4. Mahasiswa mampu menggunakan perulangan do-while untuk menyelesaikan suatu kasus.
5. Mahasiswa memahami perbedaan antara perulangan for, while dan do-while, dan mampu menentukan bentuk perulangan yang paling tepat pada kasus yang diberikan.
6. Mahasiswa mampu menggunakan break, continue dan goto dalam program Bahasa C.
7. Mahasiswa mampu menggunakan perulangan bersarang untuk menyelesaikan suatu kasus.

MATERI

5.1 Perulangan Proses

Kadangkala suatu perintah tertentu di dalam program harus dieksekusi selama beberapa kali untuk memperoleh hasil tertentu yang diinginkan. Misalnya jika seseorang ingin mencetak namanya 10 kali atau ingin mencetak nilai dari 1 sampai 100. Hal seperti itu bisa dilakukan dengan perulangan proses atau *looping*. Proses pencetakan akan dilakukan berulang kali sesuai dengan kebutuhan, daripada menuliskan proses yang diperlukan berkali-kali. Menuliskan proses yang sama berkali-kali tentu saja akan menghamburkan sumber daya, baik penyimpanan maupun waktu pembuatan kode.

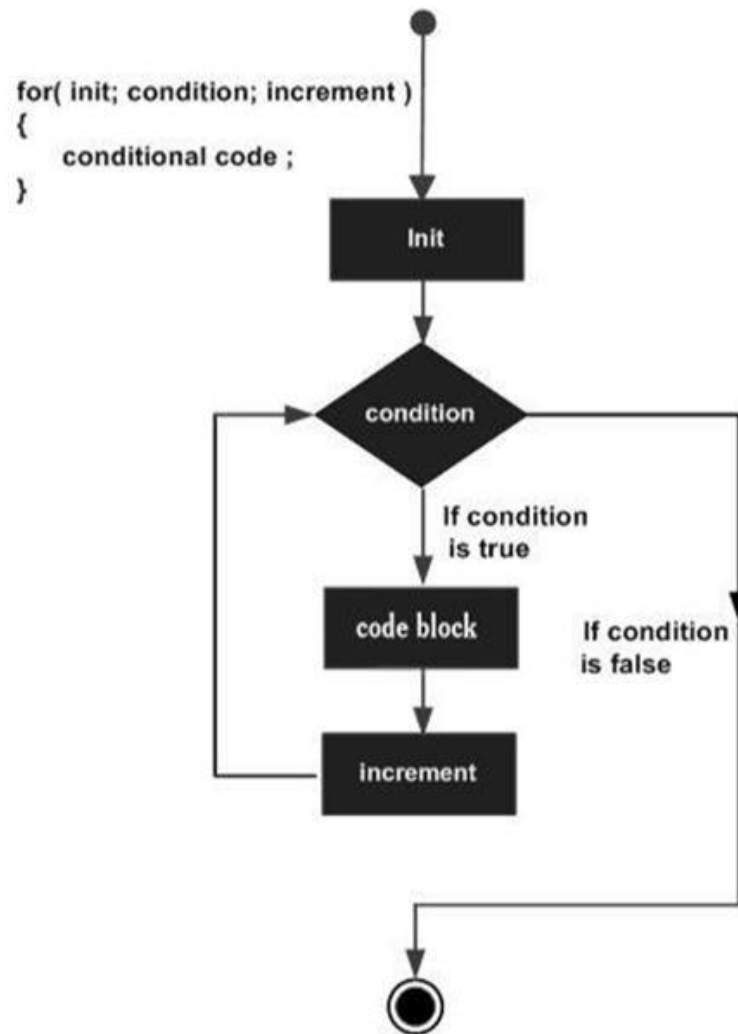
Bahasa C menyediakan tiga sintaks perulangan, yaitu `for`, `while` dan `do-while`. Perbedaan dari ketiga bentuk perulangan tersebut tampak pada Tabel 5.1. Perintah `for` dan `while` melakukan pengecekan terlebih dahulu pada kondisi, sedangkan `do-while` akan melakukan proses di dalam perulangan dahulu kemudian melakukan pengecekan untuk eksekusi berikutnya. Oleh sebab itu dapat disimpulkan bahwa minimal eksekusi `for` dan `while` adalah nol kali, sedangkan `do-while` adalah satu kali.

Tabel 5.1. Perbedaan Sintaks Perulangan

Perulangan	Keterangan
<code>for</code>	Mengeksekusi urutan pernyataan beberapa kali dan mempersingkat kode yang mengelola variabel perulangan.
<code>while</code>	Mengulangi pernyataan atau sekelompok pernyataan ketika kondisi tertentu benar. Kondisi dicek dahulu sebelum mengeksekusi badan perulangan.
<code>do-while</code>	Perulangan ini seperti pernyataan <code>while</code> , namun pengecekan kondisi dilakukan pada akhir badan perulangan.

5.2 Perulangan `for`

Perulangan `for` merupakan sintaks perulangan yang paling sederhana dan singkat. Perintah ini mampu mengeksekusi pernyataan di dalam bloknnya sesuai dengan kondisi awal dan akhir yang diberikan. Gambar 5.1 merupakan diagram alir untuk perulangan `for`.



Gambar 5.1. Diagram Alir For (Tutorials Point, 2014)

Perulangan `for` memiliki sebuah variabel pencacah, yaitu variabel yang digunakan untuk mengontrol kapan suatu perulangan akan berhenti. Sebelum masuk ke dalam pengecekan, maka nilai variabel pencacah harus diberikan inisial terlebih dahulu. Kemudian dilakukan pengecekan kondisi pada nilai pencacah apakah sudah mencapai batas akhir perulangan. Jika kondisi benar atau pencacah belum mencapai akhir perulangan, maka deretan perintah di dalam blok

perulangan akan dikerjakan dan dilakukan perubahan nilai pada pencacah, bisa berupa kenaikan nilai atau penurunan nilai, bergantung pada perintah di dalam `for`. Selanjutnya program akan kembali melakukan pengecekan, dan demikian seterusnya hingga pencacah mencapai akhir perulangan yaitu kondisi bernilai `false`. Saat kondisi bernilai `false`, maka program akan keluar dari blok `for` dan melanjutkan ke perintah berikutnya di bawah blok `for`.

Adapun bentuk dari perulangan `for` adalah sebagai berikut :

```
for(init, kondisi, increment)
{
    Perintah-perintah di sini
}
```

Init berisi nilai awal pencacah. Kondisi merupakan pengecekan kondisi dimana di dalamnya melibatkan operator perbandingan. Increment pada perintah `for` bisa berupa positif (menaikkan nilai pencacah) atau increment negatif (menurunkan nilai pencacah).

Contoh kasus `for`, misalnya akan dicetakn semua bilangan bulat dari 1 sampai 10. Maka kondisi awal adalah 1, karena pencetakan dimulai dari 1, kondisi perulangan akan terus dilakukan selama pencacah belum melebihi nilai 10 (lebih kecil atau sama dengan 10) dan karena yang dicetak semua bilangan bulat, maka peningkatannya adalah sebesar 1. Sehingga potongan kodenya adala sebagai berikut :

```
int i;
for(i=1, i<=10, i++)
{
    printf("%d\n", i);
}
```

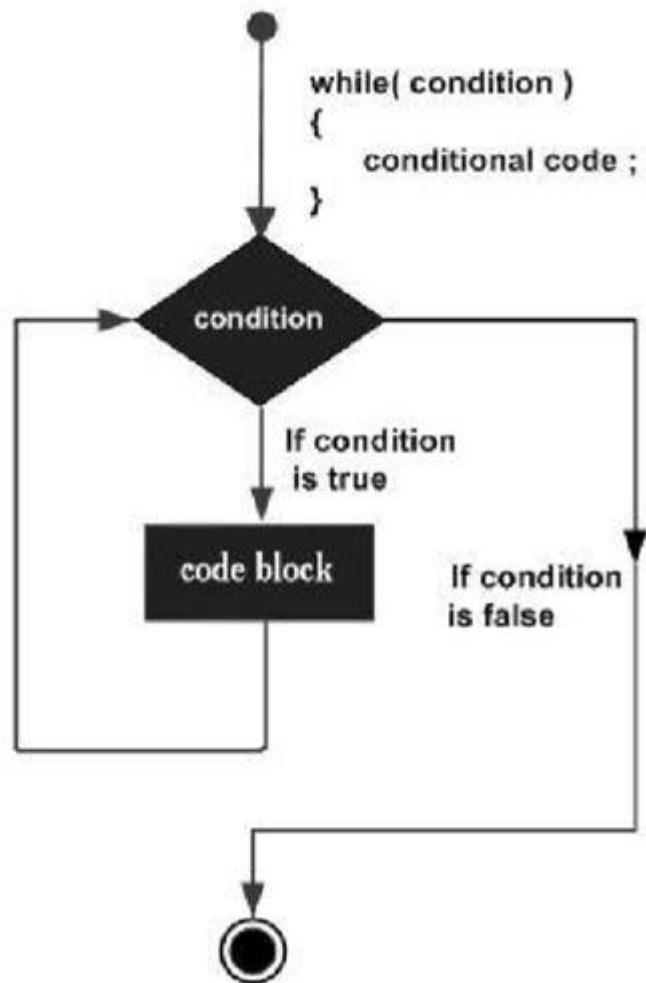
Hasil dari perulangan di atas adalah angka 1 sampai dengan 10 dan dicetak ke bawah karena adanya `\n` di akhir setiap pencetakan. Adanya increment `i++` mengindikasikan bahwa nilai `i` dinaikkan sejumlah 1. Perulangan akan terus dilakukan selama nilai pencacah `i` masih lebih kecil atau sama dengan 10.

Pertanyaan

Bagaimana jika perulangan diganti dengan `for(i=1, i<=10, i--)`?

5.3 Perulangan `while`

Perintah `while` akan melakukan perintah di dalam bloknnya selama kondisi yang diberikan bernilai benar. Tidak seperti `for`, `increment` berada pada `while` berada di dalam bloknnya sendiri. Secara algoritma sebenarnya `for` dan `while` hampir sama, hanya saja peletakan `init` dan `increment` saja yang berbeda. Gambar 5.2. merupakan diagram alir dari perulangan `while`.



Gambar 5.2. Diagram Alir Perulangan While (Tutorials Point, 2014)

Adapun struktur dari perulangan `while` adalah sebagai berikut :

```

init
while(kondisi)
{
    perintah-perintah
    increment
}
  
```

Jika kasus pencetakan bilangan bulat dari 10 sampai 1 dikerjakan dengan menggunakan `while`, maka potongan sintaksnya adalah sebagai berikut :

```
int i=10;
while(i>=1)
{
    printf("%d\n",i);
    i--;
}
```

Letak perbedaannya dengan `for`, hanya pada peletakan `init` yaitu `i=1`, berada sebelum `while`, dan `increment` berada di dalam blok `while` itu sendiri. Kondisi inisial dimulai dari 10 karena pencetakan dimulai dari 10, perulangan proses pencetakan akan selalu dilakukan selama nilai `i` lebih besar atau sama dengan satu karena nilai akhir pencetakan adalah 1. Pencetakan dari 10 menuju ke 1 bersifat menurun, sehingga pencacah diturunkan sejumlah 1.

Pertanyaan

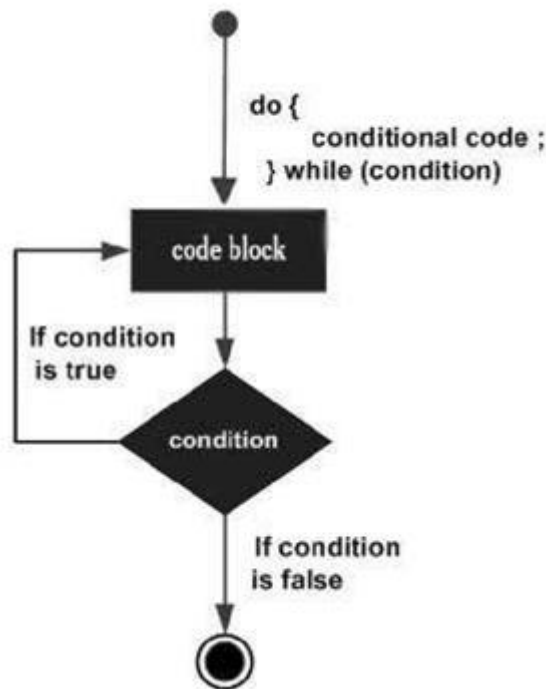
1. Bagaimana jika kondisi di dalam `while` diganti menjadi `i<=1`?
2. Kembalikan kondisi menjadi `i>=1`, lalu coba hapus `i--` di dalam blok `while`, apakah yang akan terjadi?

5.4 Perulangan do-while

Perintah `do-while` hampir sama dengan `while`, hanya saja peletakan `while` ada di bagian paling bawah blok perulangan. Karena `while` diletakkan di bagian paling bawah, maka pengecekan kondisi dilakukan di bagian akhir perulangan. Gambar 5.3 merupakan diagram alir untuk `do-while`. Tampak pada gambar bahwa perintah di dalam blok perulangan dilakukan dahulu, kemudian baru dilakukan pengecekan kondisi.

Adapun bentuk dari perulangan `do-while` adalah sebagai berikut :

```
init
do
{
    perintah-perintah
    increment
} while(kondisi);
```



Gambar 5.3. Diagram Alir Perulangan Do-While (Tutorials Point, 2014)

Misalnya ingin dicetak bilangan bulat dari 1 sampai 10 dengan menggunakan do-while, maka potongan perintahnya adalah sebagai berikut :

```

int i=1;
do
{
    printf("%d\n",i);
    i++;
} while(i<=10);
  
```

Nilai inisial dimulai dari 1 karena pencetakan dimulai dari 1. Increment adalah positif 1 karena pencetakan menaik dengan selisih sebesar 1. Pengecekan dilakukan setelah pencacah dinaikkan, dan perulangan tetap dilakukan selama nilai pencacah 10 ke bawah.

Pertanyaan

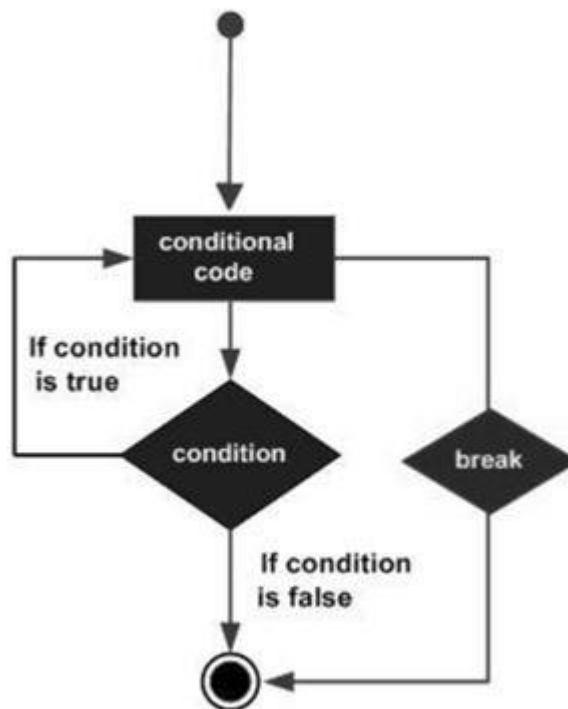
Bagaimana jika kondisi di dalam while diganti menjadi $i \geq 10$?

5.5 Perintah break

Perintah `break` di dalam program C digunakan untuk melakukan dua hal berikut ini :

- Untuk keluar dari suatu perulangan, jika `break` diletakkan di dalam blok perulangan. Setelah keluar dari blok perulangan, maka program akan melanjutkan kembali pada pernyataan berikut setelah blok perulangan.
- Digunakan untuk keluar dari perintah `switch-case` (telah dijelaskan pada bab percabangan)

Jika terdapat perulangan di dalam perulangan lainnya (perulangan bersarang), maka perintah `break` akan menghentikan perulangan terdalam di mana posisi `break` diletakkan, dan akan melanjutkan perulangan di luarnya (perulangan induk). Adapun diagram alir `break` ditunjukkan pada Gambar 5.4. Pada Gambar 5.4 `break` diletakkan pada suatu kondisi tertentu, karena memang pada sebuah perulangan yang menggunakan `break` akan terdapat suatu pengecekan kondisi, dimana jika kondisi yang diberikan benar, maka perintah `break` akan dieksekusi dan program akan keluar dari blok perulangan.



Gambar 5.4. Diagram Alir Perulangan dengan Break (Tutorials Point, 2014)

Contoh penggunaan `break` adalah program yang meminta pengguna melakukan input 10 bilangan bulat atau perulangan akan berakhir jika pengguna

menginput 0, meskipun bilangan belum mencapai 10. Potongan kodenya adalah sebagai berikut :

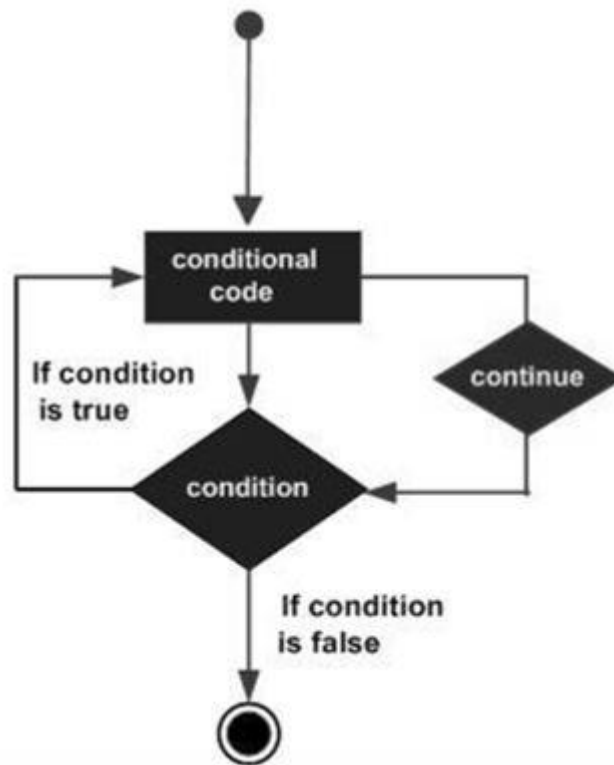
```
int i,bilangan;
for (i=1;i<=10;i++)
{
    printf("Masukkan bilangan bulat : ");
    scanf("%d",&bilangan);
    if(bilangan==0)
    {    break; }
}
```

Program di atas melakukan pengulangan mulai dari pencacah *i* bernilai 1 hingga maksimal *i* bernilai 10. Pada setiap langkahnya akan meminta input bilangan bulat, namun program akan keluar dari blok perulangan dan berhenti meminta input bilangan bulat apabila pengguna memasukkan nilai bilangan 0, walaupun nilai *i* belum mencapai 10.

5.6 Perintah *continue*

Perintah *continue* dalam program C mirip seperti *break*, namun tidak langsung mengeluarkan program dari blok perulangan. Perintah *continue* hanya akan melompati perintah pada saat tertentu dan kemudian melanjutkan kembali perulangan secara normal. *Continue* pada sebuah perulangan akan melakukan pengecekan kondisi dan menaikkan nilai pencacah perulangan. Penggunaan *continue* pada *while* dan *do-while* harus disertai dengan kenaikan pencacahnya pada setiap kondisi *continue* terpenuhi, karena jika tidak demikian maka perulangan tidak akan melebihi batas yang diinginkan. Gambar 5.5 merupakan diagram alir *continue*.

Contoh penggunaan *continue* misalnya pengguna diminta melakukan input tepat 10 bilangan bulat dan menghitung semua bilangan yang diinput, kecuali bilangan genap. Bilangan genap adalah semua bilangan bulat yang habis dibagi dua. Sehingga dalam pengecekannya digunakan tanda modulo (%). Karena setiap bilangan genap tidak dianggap dan perulangan akan dilanjutkan untuk menginput bilangan berikutnya, maka digunakanlah *continue* agar program dapat terus berjalan dan meminta input bilangan berikutnya.



Gambar 5.5. Diagram Alir Perulangan dengan Continue (Tutorials Point, 2014)

Adapun penyelesaian dari kasus sebelumnya dapat dilihat pada potongan program berikut ini :

```

int i=1,bilangan,total=0;
for(i=1;i<=10;i++)
{
    printf("Masukkan bilangan : ");
    scanf("%d",&bilangan);
    if(bilangan % 2==0)
    {
        continue;
    }
    total=total+bilangan;
}
printf("Total : %d",total);

```

Pemberian nilai inisial pada total sebesar nol diperlukan karena nilai variabel total akan digunakan untuk menyimpan jumlah dari semua bilangan bulat ganjil yang dimasukkan. Jika total tidak diberi nilai inisial maka nilai variabel total di dalam C adalah sebuah bilangan bulat acak. Hal itu akan membuat total tidak mencerminkan total nilai yang dimasukkan. Penggunaan continue pada for akan

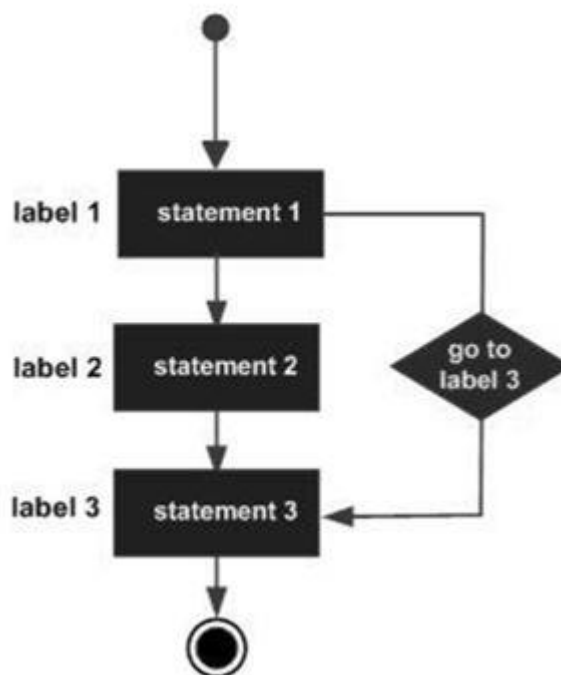
otomatis melanjutkan perulangan dengan mengabaikan perintah `total = total + bilangan`. Nilai pencacah akan otomatis dinaikkan.

Pertanyaan

Dapatkah Anda mengubah perulangan di atas dengan menggunakan perintah `while` atau `do-while`?

5.7 Perintah goto

Perintah `goto` akan melakukan lompatan menuju baris dengan `label` yang dinyatakan dalam `goto`, namun tetap dalam satu blok fungsi. Penggunaan `goto` sebenarnya tidak begitu dianjurkan karena alur program menjadi susah ditelusuri. Selain itu program menjadi susah dimengerti dan sudah dimodifikasi. Bahkan Dijkstra pernah menuliskan sebuah paper berjudul '*Goto Statement Considered Harmful*' (Banahan dkk, 1991). Diagram alir dalam perintah `goto` ditunjukkan pada Gambar 5.6.



Gambar 5.6. Diagram Alir Goto (Tutorials Point, 2014)

Struktur perintah `goto` adalah sebagai berikut :

```
goto label;  
..  
label: statement;
```

Contoh dari penggunaan perintah `goto` dapat dilihat pada program di bawah ini :

```
int i=1;
blok1: /*ini adalah label bernama blok1*/
printf("%d\n",i);
i++;
if(i<=10)
{
    goto blok1;
}
```

Pada perintah di atas terdapat sebuah label bernama blok1. Pemberian nama label mengikuti aturan pada pemberian nama identifier, karena label juga merupakan identifier. Label ditandai dengan adanya tanda titik dua setelah nama label. Perintah di atas akan mencetak nilai i yaitu 1, kemudian menaikkan nilai i menjadi 2 dan akan menuju blok `if` dan mengecek apakah i kurang dari atau sama dengan 10. Karena kondisi dalam `if` bernilai `true`, maka program akan menuju label blok1, sehingga program kembali mencetak i yaitu 2, kemudian melakukan hal yang sama lagi. Perintah `goto` tidak akan dieksekusi apabila blok `if` salah, yaitu ketika i di atas 10, sehingga keluaran dari program di atas adalah angka 1 sampai 10. Program seperti di atas dapat dituliskan kembali dengan menggunakan perulangan dan menghindari perintah `goto`.

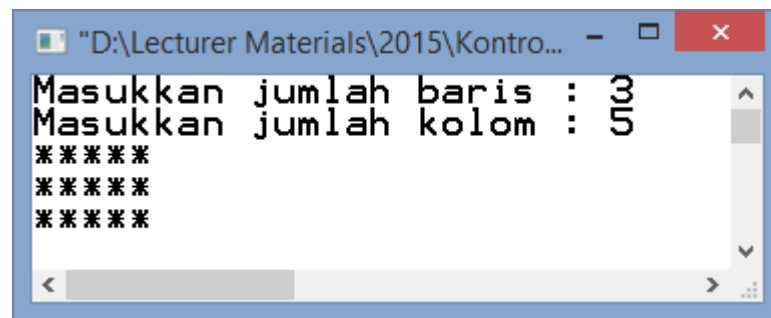
5.8 Perulangan Bersarang

Sebuah perulangan di dalam perulangan lainnya disebut perulangan bersarang. Ketika terdapat perulangan bersarang, maka program akan menghabiskan terlebih dahulu perulangan yang paling dalam, baru kemudian melanjutkan perulangan yang di luarnya. Contoh perulangan bersarang misalnya adalah jika pengguna diminta untuk mencetak tanda bintang ke samping sejumlah m dan ke bawah sejumlah n dengan m dan n adalah masukan pengguna. Berikut adalah potongan program kasus tersebut :

```
int i,j,m,n;
printf("Masukkan jumlah baris : "); scanf("%d",&n);
printf("Masukkan jumlah kolom : "); scanf("%d",&m);
for (i=1;i<=n;i++)
{
    for(j=1;j<=m;j++)
    {
        printf("*");
    }
}
```

```
}  
printf("\n");  
}
```

Pada program di atas terdapat dua perulangan yaitu dengan pencacah *i* dan *j*. Misalnya dimasukkan *n* 3 dan *m* 5, maka perulangan akan dimulai dengan nilai *i*=1 lalu melanjutkan *j* mulai dari 1 sampai dengan 5. Untuk setiap *j* mulai dari 1 sampai 5 dicetak tanda * ke samping. Setelah nilai *j* melebihi 5, maka dicetaklah baris baru dan *i* dilanjutkan menjadi 2 dan *j* dimulai lagi dari 1 sampai 5, oleh sebab itu 5 buah bintang berikutnya dicetak di bawahnya. Demikian berikutnya hingga *i* melebihi 3 baru baru program berakhir. Keluaran dari program di atas adalah :



Gambar 5.7. Keluaran Perulangan Bersarang

5.9 Perulangan Tak Berhingga

Sebuah perulangan bisa menjadi perulangan tak berhingga (*infinite loop*) jika kondisi di dalam perulangan tidak pernah bernilai false. Perulangan tak berhingga tidak akan pernah berakhir dan untuk keluar dari program tersebut, pengguna bisa menekan Ctrl+C. Perintah `for` seringkali digunakan dalam perulangan seperti ini. Inisialisasi, kondisi dan increment bisa dikosongi dalam `for` untuk membuat perulangan yang tak berhingga. Adapun contohnya adalah sebagai berikut :

```
for(;;)  
{  
    printf("Perulangan ini tak pernah berakhir.\n");  
}
```

Pertanyaan

Dapatkah Anda memberikan contoh lain dari perulangan tak berhingga?

5.10 Contoh Soal

Sesi berikut akan membahas beberapa contoh soal yang berkaitan dengan perulangan proses dan sekaligus pembahasannya.

1. Buatlah program untuk mencetak bilangan seperti berikut : 14 11 8 5 2 -1

Pembahasan

Untuk mencetak angka tersebut dapat dilihat bahwa angka dimulai dari 14 dan berakhir pada nilai -1. Adapun pencetakan bersifat menurun dengan penurunan 3. Oleh sebab itu bisa digunakan for dengan init $i=14$ dan perulangan akan tetap dilakukan selama $i \geq -1$, dan increment adalah -3. Adapun program untuk kasus tersebut adalah sebagai berikut :

```
#include <stdio.h>

void main()
{
    int i;
    for (i=14; i>=-1; i=i-3)
    {
        printf("%d ", i);
    }
}
```

2. Buatlah program untuk meminta input bilangan bulat kepada pengguna dan kemudian merata-ratakan semua bilangan yang diinput. Hentikan input data dan tampilkan rata-rata jika pengguna menginput bilangan 0.

Pembahasan

Program akan melakukan permintaan input bilangan bulat dan berakhir ketika diinput 0. Karena dalam hal ini pengguna minimal memasukkan satu bilangan bulat, maka bisa digunakan bentuk do-while. Kondisi dilakukan dengan menggunakan ekspresi selama input bukan 0 ($\text{bilangan} \neq 0$). Adapun perintah yang dilakukan adalah menjumlahkan bilangan-bilangan yang dimasukkan. Perhitungan rata-rata dilakukan setelah perulangan. Berikut ini adalah penyelesaian dari kasus di atas :

```
#include <stdio.h>

void main()
{
    float bilangan, total=0, rata, jumlah_data=0;
```

```

do
{
    printf("Masukkan bilangan : ");
    scanf("%f",&bilangan);
    total=total+bilangan;
    jumlah_data++;
}while(bilangan!=0);
rata=total/(jumlah_data-1);
printf("Rata : %.2f",rata);
}

```

Variabel jumlah_data digunakan untuk menghitung berapa banyaknya data yang dimasukkan. Hal ini penting karena pada saat menghitung rata-rata, total data akan dibagi dengan jumlah data. Pada saat pembagian, jumlah_data dikurangi satu karena data dengan nilai 0 tidak ikut dalam perhitungan rata-rata.

3. Apakah yang dilakukan oleh program di bawah ini :

```

#include <stdio.h>

main()
{
    long int total;
    int panjang,i;
    printf("Masukkan panjang deret : ");
    scanf("%d",&panjang);
    total =0;
    for (i=0;i<=panjang;i++)
        total +=pow(2,i);
    printf("Total : %ld",total);
}

```

Gambar 5.8. Program Perulangan

Pembahasan

Fungsi pow pada program di atas berarti perpangkatan. Perintah pow(2,i) berarti mengangkat angka 2 dengan i atau 2^i . Program di atas akan meminta input panjang dan menjumlahkan perpangkatan dari 2^0 sampai dengan 2^{panjang} . Contoh keluaran program di atas adalah :

```

Masukkan panjang deret : 4
Total : 31

```

Nilai 31 diperoleh dari : $2^0 + 2^1 + 2^2 + 2^3 + 2^4$

Nilai 31 diperoleh dari : $1 + 2 + 4 + 8 + 16 = 31$

4. Buatlah program untuk memberikan keluaran berikut ini :

```
Masukkan n : 4
*
**
***
****
```

Pembahasan

Program di atas meminta input sebuah bilangan bulat. Program akan mencetak bintang ke bawah sejumlah n yang diinput, dan untuk setiap barisnya akan dicetak bintang sejumlah baris bersangkutan. Misalnya pada baris 1, dicetak 1 bintang, pada baris 2 dicetak 2 bintang dan seterusnya. Oleh sebab itu akan terdapat perulangan bersarang, dimana perulangan utama melakukan perulangan sebanyak n kali dan perulangan di dalamnya akan melakukan perulangan untuk mencetak bintang sebanyak nilai baris saat itu. Adapun penyelesaian dari program di atas adalah sebagai berikut :

```
#include <stdio.h>

void main()
{
    int i,j,n;
    printf("Masukkan n : ");scanf("%d",&n);
    for (i=1;i<=n;i++)
    {
        for (j=1;j<=i;j++)
        {
            printf("*");
        }
        printf("\n");
    }
}
```

Perulangan j mulai dari 1 sampai dengan i pada saat itu. Hal itu dilakukan agar bintang yang dicetak ke samping mengikuti nilai baris saat itu.

Pertanyaan

Bagaimana jika perulangan j diganti dengan `for (j=1;j<=n;j++)` ?

5.11 Penutup

Kesimpulan

- Suatu perintah tertentu di dalam program yang dieksekusi selama beberapa kali untuk memperoleh hasil tertentu yang diinginkan disebut dengan perulangan proses.
- Perulangan proses dalam Bahasa C dapat dilakukan dengan tiga sintaks yaitu `for`, `while` dan `do-while`.
- Perulangan `for` merupakan sintaks perulangan yang paling sederhana dan singkat.
- Bentuk perulangan `for` di dalamnya ada nilai inisial, pengecekan kondisi dan increment. Perulangan akan dimulai dari nilai inisial dan selama kondisi bernilai `true`. Increment adalah perubahan nilai pencacah perulangan.
- Perulangan `while` melakukan pengecekan kondisi terlebih dahulu kemudian melakukan sederetan perintah di dalamnya.
- Inisialisasi pencacah dilakukan sebelum `while`. Kondisi dicek pada `while` dan increment diletakkan di dalam blok `while`.
- Perulangan `do-while` melakukan sederetan perintah di dalam blok perulangan terlebih dahulu, kemudian melakukan pengecekan kondisi.
- Struktur perulangan `do-while` mirip dengan `while`, kecuali pada peletakan sintaks `while` di bagian akhir blok loop.
- Perulangan `for` dan `while` minimal melakukan nol kali eksekusi, sedangkan `do-while` minimal sekali.
- `Break` digunakan untuk keluar dari satu blok perulangan terdalam dimana posisi `break` berada.
- `Continue` digunakan untuk mengabaikan perintah di dalam perulangan dan melanjutkan perulangan dengan nilai pencacah berikutnya.
- `Goto` digunakan untuk melompat pada label tertentu. Penggunaan `goto` harus menggunakan label.
- Penggunaan `goto` dalam program tidak disarankan karena membuat program menjadi susah dimengerti dan alurnya susah untuk ditelusuri.

- Perulangan bersarang biasanya digunakan untuk menyelesaikan kasus yang menuntut adanya ekspresi berulang yang dieksekusi secara bertingkat.
- Perulangan tak berhingga merupakan perulangan yang tidak pernah berhenti. Salah satu bentuknya adalah dengan mengosongkan kondisi di dalam for.

Latihan

1. Modifikasilah conoth soal nomor 3 dengan menggunakan while dan do-while.
2. Buatlah program untuk membuat tampilan berikut :

Masukkan baris : 5

Masukkan kolom : 6

1	2	3	4	5	6
2	4	6	8	10	12
3	6	9	12	15	18
4	8	12	16	20	24
5	10	15	20	25	30

3. Buatlah program untuk mencetak semua pembagi dari suatu bilangan yang diinput. Bilangan m dikatakan sebagai pembagi n jika m habis membagi n.
4. Buatlah program untuk mengecek apakah username dan password yang dimasukkan benar. Username yang benar 123 sedangkan password 321. Jika benar, berikan pesan sukses, namun jika salah, maka minta pengguna untuk memasukkan username dan password kembali sampai maksimal tiga kali mencoba.
5. Buatlah program untuk mengelola sebuah tabungan dengan menu : Tabung, Tarik, Cek Saldo dan Keluar. Saldo awal dianggap nol. Menu akan selalu ditampilkan sampai pengguna memasukkan Pilihan Keluar. Jika menu pilih dipilih maka akan meminta input jumlah tabungan dan saldo akan bertambah. Jika menu Tarik dipilih maka akan diminta jumlah uang yang ditarik, kemudian akan dicek apakah saldo mencukup. Jika saldo cukup, kurangi saldo, sebaliknya berikan pesan tidak bisa. Cek saldo akan menampilkan saldo tabungan.

- Masukkan baris : 6



Gambar 5.9. Program Bintang

7. Buatlah program untuk menginput bilangan m dan n , kemudian kalikan semua bilangan mulai dari m sampai dengan n .
8. Buatlah program untuk meminta input n buah bilangan bulat, kemudian berikan keluaran bilangan terbesar dan terkecil dari semua bilangan yang diinput tersebut.
9. Program berikut dimaksudkan untuk melakukan pencetakan “Hai” sebanyak 10 kali. Ceklah apa ada yang salah dalam program berikut ini! Jika ada, jelaskan penyebabnya dan perbaikilah!

```
#include <stdio.h>

void main()
{
    int i;
    for(i=1;i<=10;i++);
    {
        printf("Hai\n");
    }
}
```

Umpan Balik

Setelah menyelesaikan bab ini, mahasiswa diharapkan mampu memahami konsep dasar perulangan, dan dapat menentukan apakah suatu kasus yang diberikan merupakan perulangan atau bukan. Mahasiswa diharapkan mampu menggunakan perulangan yang paling tepat sesuai dengan kasus yang diberikan, memahami beberapa perintah yang biasanya menyertai perulangan seperti break dan continue. Selain itu mahasiswa diharapkan mampu mengetahui penyebab perulangan tidak berhingga dan dapat membuatnya berhenti jika diperlukan. Pemahaman akan bab ini sangat penting agar mahasiswa dapat melanjutkan mempelajari bab berikutnya.

BAB VI

FUNGSI

Standar Kompetensi

Mahasiswa diharapkan mampu menerjemahkan permasalahan dalam dunia nyata ke dalam algoritma dan mengimplementasikannya ke dalam Bahasa C setelah mempelajari mata kuliah ini.

Kompetensi Dasar

Setelah mempelajari materi ini, mahasiswa diharapkan mampu memahami dan menerapkan fungsi dan sub program dalam Bahasa C untuk membuat program menjadi lebih rapi dan efisien.

Indikator Pencapaian Kompetensi

Adapun indikator hasil belajar adalah sebagai berikut :

1. Mahasiswa mampu membuat fungsi dalam penyelesaian suatu permasalahan.
2. Mahasiswa mampu menentukan nilai kembalian suatu fungsi
3. Mahasiswa mampu menerapkan parameter dalam suatu fungsi
4. Mahasiswa dapat melakukan pemanggilan fungsi
5. Mahasiswa mampu menjelaskan perbedaan variabel lokal dan global
6. Mahasiswa mampu menggunakan metode melewati argumen dengan tepat
7. Mahasiswa mampu menyusun fungsi rekursi dari suatu permasalahan yang diberikan.

MATERI

6.1 Fungsi

Fungsi merupakan suatu bagian dari program yang dimaksudkan untuk mengerjakan suatu tugas tertentu dan letaknya terpisah dari program yang memanggilmnya. Setiap program di dalam Bahasa C minimal memiliki satu fungsi yaitu fungsi `main()`, dan sebagian besar program dapat mendefinisikan program tambahan. Suatu kode program dapat dipecah ke dalam beberapa fungsi. Cara pemecahannya akan berbeda-beda untuk setiap orang, namun secara logika pembagiannya akan didasarkan pada tugas spesifik yang dilakukannya. Fungsi dideklarasikan dengan nama fungsi, tipe kembalian, dan parameternya. Suatu fungsi akan memiliki tubuh fungsi, dimana perintah-perintah tertentu dituangkan. Bahasa C menyediakan beberapa fungsi bawaan di dalam library-nya. Misalnya fungsi `pow` dalam library `math.h` untuk melakukan perpangkatan. Sebuah fungsi kadangkala disebut sebagai *method*, *sub-rutin* atau *prosedur*.

Keuntungan menggunakan fungsi :

- Program besar dapat dipisah menjadi program-program kecil.
- Dapat dikerjakan oleh beberapa orang sehingga koordinasi mudah.
- Kemudahan dalam mencari kesalahan-kesalahan karena alur logika jelas dan kesalahan dapat dilokalisasi dalam suatu modul tertentu saja.
- Modifikasi program dapat dilakukan pada suatu modul tertentu saja tanpa mengganggu program keseluruhan.
- Mempermudah dokumentasi.
- Reusability: Suatu fungsi dapat digunakan kembali oleh program atau fungsi lain

Suatu fungsi dapat didefinisikan dalam bentuk berikut ini :

```
tipe_kembalian nama_fungsi( daftar parameter ) //header
{
    Tubuh fungsi
}
```

Suatu definisi fungsi terdiri dari bagian header dan tubuh. Berikut ini adalah penjelasannya :

- Tipe kembalian : sebuah fungsi mungkin akan memberikan nilai kembalian tertentu. Tipe kembalian akan berupa tipe data dari nilai yang akan dikembalikan oleh fungsi. Beberapa fungsi juga mungkin saja tidak memberikan nilai kembalian, sehingga tipe kembalian akan diisi dengan void.
- Nama fungsi : merupakan nama dari suatu fungsi. Pemberian nama fungsi mengikuti aturan penamaan identifier karena nama fungsi merupakan identifier. Nama fungsi dan daftar parameter di dalamnya disebut sebagai signatur fungsi.
- Parameter : Ketika sebuah fungsi dipanggil, pengguna bisa mengirimkan nilai pada parameter. Nilai ini disebut parameter aktual atau argumen. Daftar parameter terdiri dari tipe data, urutan dan jumlah parameter di dalam fungsi. Parameter fungsi bersifat opsional, sehingga jika tidak diperlukan, maka fungsi bisa saja tidak berisi parameter.
- Tubuh fungsi : tubuh fungsi berisi sekumpulan perintah yang mendefinisikan apa yang dilakukan oleh suatu fungsi.

6.1.1 Fungsi Tanpa Berparameter

Sebuah fungsi tanpa parameter adalah fungsi yang tidak memiliki parameter. Pendefinisianannya adalah sebagai berikut :

```
tipe_kembalian nama_fungsi( )  
{  
    Tubuh fungsi  
}
```

Contohnya adalah sebuah fungsi yang mencetak kata Hai sebanyak 10 kali dapat dilihat pada contoh berikut :

```
void cetakHai()  
{  
    int i;  
    for(i=1;i<=10;i++)  
    {  
        printf("Hai\n");  
    }  
}
```

Pada contoh di atas, fungsi hanya terdiri dari tipe, nama dan tubuh. Tipe fungsi adalah void, sehingga tidak memberikan nilai kembalian. Fungsi bernama cetakHai. Tubuh fungsi melakukan pencetakan Hai sebanyak 10 kali.

6.1.2 Fungsi Berparameter

Fungsi berparameter seringkali diperlukan pada saat membuat sebuah fungsi. Parameter menyimpan berbagai variabel yang nilainya bisa dikirimkan dari pemanggil fungsi. Adapun contohnya misalnya suatu fungsi yang akan menghitung nilai faktorial dari suatu bilangan.

```
int faktorial(int n)
{
    int i, fakto=1;
    for(i=1;i<=n;i++)
    {
        fakto=fakto*i;
    }
    return fakto;
}
```

Program di atas terdiri dari satu parameter bertipe int yang bernama n. Nilai kembalian fungsi adalah integer, sehingga di dalam fungsi harus memberikan nilai return variabel bertipe integer. Nilai kembalian akan tergantung pada nilai parameter n yang dikirimkan.

6.2 Deklarasi dan Pemanggilan Fungsi

Deklarasi fungsi akan memberitahu kepada *compiler* bagaimana memanggil fungsi. Deklarasi fungsi hanya menuliskan header fungsi saja dan diakhiri dengan tanda titik koma (;). Implementasi fungsi bisa saja dilakukan terpisah. Contoh cara mendeklarasikan fungsi adalah sebagai berikut :

```
int faktorial(int n);
float bagi(int , int);
```

Cara pendeklarasin sebenarnya hanya mementingkan tipe data dari daftar parameter, sehingga nama parameter tidak akan menjadi masalah jika tidak dituliskan. Pendeklarasian fungsi harus dilakukan jika fungsi didefinisikan pada suatu file sumber dan fungsi dipanggil pada file lain. Pendeklarasian fungsi harus ditulis di bagian atas file yang memanggil fungsi tersebut.

Ketika membuat sebuah fungsi dan telah mendefinisikan suatu fungsi, fungsi tersebut harus dipanggil untuk melakukan tugas yang telah didefinisikan. Ketika sebuah program memanggil fungsi, maka program akan langsung berpindah ke dalam fungsi hingga baris fungsi berakhir, kemudian baru kembali ke dalam program utama yang memanggil fungsi. Ketika memanggil sebuah fungsi, maka hanya perlu dilakukan penulisan nama fungsi dan daftar parameternya. Jika suatu fungsi mengembalikan nilai, maka nilai tersebut bisa disimpan ke dalam variabel tertentu.

Berikut ini adalah contoh program yang berisi fungsi dan pemanggilannya :

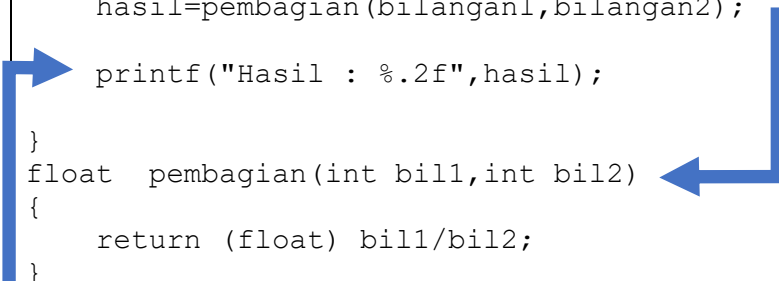
```
#include <stdio.h>

float  pembagian(int ,int );

void main()
{
    int bilangan1, bilangan2;
    float hasil;
    printf("Masukkan bilangan 1 : ");
    scanf("%d",&bilangan1);
    printf("Masukkan bilangan 2 : ");
    scanf("%d",&bilangan2);

    hasil=pembagian(bilangan1,bilangan2);
    printf("Hasil : %.2f",hasil);
}

float  pembagian(int bill,int bil2)
{
    return (float) bill/bil2;
}
```



Pada contoh di atas terdapat deklarasi satu fungsi bernama pembagian yang akan membagi sebuah bilangan dengan bilangan lainnya. Ada dua parameter bertipe integer di dalamnya dan nilai kembalian adalah float yaitu suatu hasil pembagian antara parameter ke-1 dengan parameter ke-2. Program akan berjalan dari program utama yaitu main. Program meminta masukan dua buah bilangan bulat, kemudian program memanggil fungsi pembagian, sehingga program melompat ke definisi fungsi pembagian. Pengiriman parameter harus sesuai dengan urutan yang diinginkan pada deklarasi fungsi. Setelah melakukan pembagian, maka program kembali ke program utama dan melanjutkan untuk

mencetak variabel hasil. Penyimpanan nilai kembalian suatu fungsi bisa dilakukan pada variabel dengan tipe data yang sesuai.

6.3 Argumen Fungsi

Jika fungsi menggunakan argumen, maka harus mendeklarasikan variabel yang menerima nilai-nilai argumen. Variabel ini disebut parameter formal fungsi. Parameter formal berperilaku seperti variabel lokal di dalam fungsi dan diciptakan pada saat masuk ke dalam fungsi dan dihancurkan setelah keluar. Ketika memanggil fungsi, ada dua cara di mana argumen dapat dikirimkan ke fungsi.

Tabel 6.1. Jenis Pemanggilan Argumen

Jenis Pemanggilan	Keterangan
Call by Value	Metode ini melakukan penyalinan nilai parameter aktual ke dalam parameter formal yang dimiliki fungsi. Perubahan terhadap parameter yang dilakukan di dalam fungsi tidak akan memberikan perubahan pada nilai argumen setelah keluar dari fungsi.
Call by Reference	Metode ini melakukan penyalinan alamat pada sebuah argumen ke dalam parameter formal fungsi. Di dalam fungsi, alamat digunakan untuk melakukan akses ke dalam argumen yang digunakan pada pemanggilan fungsi. Hal ini berarti setiap perubahan terhadap parameter akan berpengaruh pada argumen.

Contoh pemanggilan argumen dengan call by value adalah pada contoh pemanggilan fungsi faktorial sebelumnya. Contoh lainnya adalah sebagai berikut :

```
#include <stdio.h>

void  naikkan(int awal ,int perubahan )
{
    awal += perubahan;
}

void main()
{
    int n=2,m=4;
    printf("Nilai n = %d, m = %d\n",n,m);
    naikkan(n,m);
    printf("\nSetelah fungsi naikkan dipanggil\n");
    printf("Nilai n = %d",n);
}
```

Pada program di atas, nilai n adalah 2 dan m adalah 4. Seharusnya ketika fungsi naikan dipanggil, maka nilai n harusnya naik sebesar 4 menjadi 6. Namun yang terjadi adalah nilai n tetap 2 ketika dicetak dari program utama. Hal ini karena model pemanggilan pass by value hanya mengkopi nilai argumen n ke dalam parameter awal, sehingga yang diubah menjadi 6 sebenarnya adalah awal dan bukan n. Variabel awal dan n adalah dua variabel yang berbeda, sehingga perubahan pada variabel awal tidak akan mempengaruhi variabel n pada program utama.

Pada kasus di atas, sebaiknya digunakan call by reference untuk variabel awal, agar nilai perubahan juga berpengaruh pada n. Sedangkan untuk variabel kenaikan tidak akan bermasalah jika menggunakan call by value, karena nilai m tidak diubah sepanjang eksekusi program. Adapun setelah dimodifikasi, program di atas menjadi

```
#include <stdio.h>

void  naikan(int *awal ,int perubahan )
{
    *awal += perubahan;
}

void main()
{
    int n=2,m=4;
    printf("Nilai n = %d, m = %d\n",n,m);
    naikan(&n,m);
    printf("\nSetelah fungsi naikan dipanggil\n");
    printf("Nilai n = %d",n);
}
```

Perbedaan antara penyelesaian pertama dan kedua adalah, pada penyelesaian kedua dilakukan penyalinan alamat variabel n ke dalam awal, sehingga awal pada fungsi naikan menunjuk pada alamat variabel n, apapun perubahan pada variabel awal akan mempengaruhi nilai n pada program utama. Pemanggilannya pada program utama, menggunakan tanda & di depan nama variabel agar mengacu pada alamat variabel n. Keluaran dari program di atas adalah akhirnya n bernilai 6 karena nilai pada alamat n dinaikkan sebesar 4.

6.4 Cakupan Variabel (scope variable)

Cakupan variabel adalah area dimana sebuah variabel dikenali. Ada tiga jenis variabel berdasarkan tempat dimana suatu variabel dapat dideklarasikan dalam pemrogram C seperti ditunjukkan pada Tabel 6.2

Tabel 6.2. Jenis Variabel Berdasarkan Cakupan

Jenis	Keterangan
Variabel lokal	Dideklarasikan di dalam blok fungsi atau blok. Cakupannya hanya di dalam fungsi atau blok dimana variabel tersebut dideklarasikan.
Variabel global	Dideklarasikan di luar blok fungsi, biasanya di bagian atas program. Cakupannya adalah seluruh program, sehingga semua fungsi dapat memanggil variabel tersebut. Jika terdapat nama variabel lokal dan global yang sama, maka variabel lokal yang akan digunakan pada suatu fungsi.
Parameter formal	Dideklarasikan sebagai parameter fungsi. Cakupannya hanya di dalam fungsi itu saja. Jika terdapat nama parameter formal dan global yang sama, maka parameter formal yang akan digunakan.

Contoh pendeklarasian variabel lokal adalah seperti ditunjukkan pada program berikut :

```
#include <stdio.h>

void cetakHai(int n )
{
    int i;
    for(i=1;i<=n;i++)
        printf("Hai\n");
}

void main()
{
    int jumlah;
    printf("Masukkan jumlah cetak : ");
    scanf("%d",&jumlah);
    cetakHai(jumlah);
}
```

Variabel *i* adalah variabel lokal dari fungsi *cetakHai* dan hanya dapat digunakan dalam fungsi *cetakHai*, dan tidak bisa diakses dalam fungsi *main*. Pengaksesan akan menyebabkan kesalahan sintaks. Demikian juga dengan

variabel jumlah, hanya dapat diakses pada fungsi main saja dan tidak dapat dikenal pada fungsi cetakHai. Variabel lokal yang tidak diberikan nilai oleh pemrogram akan mengambil *garbage value* pada memorinya sehingga seringkali sebuah variabel lokal yang tidak diberi nilai inisial akan memberikan hasil yang tidak sesuai harapan.

Contoh lain dari penggunaan variabel lokal dapat dilihat pada contoh program berikut ini :

```
#include <stdio.h>

void main()
{
    int i;
    for (i=1;i<=10;i++)
    {
        int x=2*i;
        printf("%d\n",x);
    }
}
```

Variabel x adalah variabel lokal di dalam blok for i, sehingga x hanya dikenali di dalam blok for dan tidak di seluruh fungsi main. Jika x dipanggil di luar blok for (meskipun di dalam fungsi main), maka akan terjadi kesalahan sintaks.

Contoh penggunaan variabel global adalah sebagai berikut :

```
#include <stdio.h>

int bilangan;
void naikan(int perubahan )
{
    bilangan += perubahan;
}

void main()
{
    int m=4;
    printf("Bilangan sebelum dinaikkan = %d\n",bilangan);
    naikan(m);
    printf("\nSetelah fungsi naikan dipanggil\n");
    printf("Bilangan = %d",bilangan);
}
```

Pada contoh di atas, variabel bilangan adalah variabel global yang mana dideklarasikan di luar semua fungsi yang ada, sehingga variabel bilangan dapat

diakses dari semua fungsi, baik fungsi naikan maupun fungsi main. Nilai variabel bilangan diberikan nilai inisial nol oleh program, sehingga sebelum memanggil fungsi naikan, nilainya nol dan setelah dinaikkan nilainya ditambah 4 menjadi 6. Tidak seperti variabel lokal, nilai variabel global yang tidak diberikan nilai inisial akan diberi nol untuk sebuah integer. Tabel 6.3 adalah nilai inisial yang diberikan sistem jika variabel global tidak diberikan nilai inisial oleh pemrogram.

Tabel 6.3. Nilai inisial variabel global

Tipe Data	Nilai inisial oleh sistem
int	0
char	'\0'
float	0
double	0
pointer	null

Parameter formal dideklarasikan di dalam fungsi. Contohnya yaitu variabel perubahan pada fungsi naikan. Variabel tersebut hanya bisa diakses di dalam fungsi naikan saja dan tidak dari luar.

6.5 Fungsi Rekursi

Suatu fungsi dapat memanggil fungsi lain atau fungsi itu sendiri. Rekursi adalah suatu proses dari fungsi yang memanggil dirinya sendiri secara berulang-ulang. Karena proses dilakukan berulang, maka harus ada suatu kondisi yang mengakhiri prosesnya. Jika tidak maka proses tidak berhenti sampai memori yang digunakan tidak dapat menampung lagi.

Contoh fungsi rekursi adalah sebagai berikut :

```
int faktorial(int n)
{
    if(n=0)
        return 1;
    else
        return faktorial(n-1)*n;
}
```

Fungsi faktorial di atas merupakan fungsi rekursi karena fungsi tersebut memanggil dirinya sendiri, namun dengan nilai parameter yang berbeda. Nilai parameter yang dikirim selalu berkurang sebesar 1, dan pemanggilan akan berakhir pada saat nilai parameter adalah 0. Misalnya jika dipanggil `faktorial(5)`, maka akan diperoleh hasil sebagai berikut :

$$\begin{aligned}\text{Faktorial}(5) &= \text{faktorial}(4) * 5 \\ &= \text{faktorial}(3) * 4 * 5 \\ &= \text{faktorial}(2) * 3 * 4 * 5 \\ &= \text{faktorial}(1) * 2 * 3 * 4 * 5 \\ &= \text{faktorial}(0) * 1 * 2 * 3 * 4 * 5 \\ &= 1 * 1 * 2 * 3 * 4 * 5 \\ &= 120\end{aligned}$$

Pertanyaan

Bagaimana jika fungsi faktorial dipersingkat menjadi :

```
int faktorial(int n)
{
    return faktorial(n-1)*n;
}
```

6.6 Contoh Soal

Sesi berikut akan membahas beberapa contoh soal yang berkaitan dengan fungsi dan sekaligus pembahasannya.

1. Buatlah program untuk melakukan perhitungan gaji pegawai! Gaji ditentukan oleh tiga jenis penghasilan yaitu, gaji pokok, tunjangan dan gaji lembur. Gaji lembur tergantung pada jam lembur. Gaji lembur per jam adalah 20000. Buatlah fungsi untuk menghitung gaji lembur dan gaji total!

Pembahasan

Akan terdapat dua fungsi lain selain `main` yaitu `hitungGajiTotal` dan `hitungGajiLembur`. Fungsi `hitungGajiTotal` adalah untuk menghitung gaji total dengan tiga parameter yaitu gaji pokok, tunjangan dan gaji lembur, sedangkan fungsi `hitungGajiLembur` melakukan perhitungan gaji lembur berdasarkan parameter jam lembur. Tipe kembalian kedua fungsi adalah integer, yaitu

sesuai dengan variabel yang akan dihitungnya. Berikut ini adalah kode programnya :

```
#include <stdio.h>
#define GAJIPERJAM 20000

int hitungGajiTotal(int gajiPokok,int tunjangan, int gajiLembur)
{
    return gajiPokok+tunjangan+gajiLembur;
}
int hitungGajiLembur(int jamLembur)
{
    return GAJIPERJAM*jamLembur;
}
void main()
{
    int gajiPokok, tunjangan, jamLembur;
    int gajiLembur,gajiTotal;
    printf("Masukkan gaji pokok : ");
    scanf("%d",&gajiPokok);
    printf("Masukkan tunjangan : ");
    scanf("%d",&tunjangan);
    printf("Masukkan jam lembur : ");
    scanf("%d",&jamLembur);
    gajiLembur=hitungGajiLembur(jamLembur);
    gajiTotal=hitungGajiTotal(gajiPokok,tunjangan
                              ,gajiLembur);
    printf("Gaji Total : %d",gajiTotal);
}
```

2. Buatlah program untuk melakukan pertukaran dua buah bilangan dengan sebuah fungsi bernama tukar!

Pembahasan

Program melakukan pertukaran dua buah nilai, sehingga memerlukan dua buah paramener. Menukar nilai berarti harus mengubah nilai parameter, oleh sebab itu pemanggilan argumen perlu dilakukan dengan cara call by reference.

```
#include <stdio.h>

void tukar(int *bil1,int *bil2)
{
    int temp;
    temp=*bil1;
    *bil1=*bil2;
    *bil2=temp;
}
void main()
```



```

{
    int bilangan1, bilangan2;
    printf("Masukkan bilangan 1: ");
    scanf("%d",&bilangan1);
    printf("Masukkan bilangan 2: ");
    scanf("%d",&bilangan2);
    tukar(&bilangan1,&bilangan2);
    printf("Bilangan 1: %d", bilangan1);
    printf("Bilangan 2: %d", bilangan2);
}

```

Pertanyaan

1. Apakah yang terjadi jika pemanggilan argumen pada jawaban di atas diubah menjadi call by value?
2. Apakah Anda bisa memodifikasi program di atas agar pertukaran tetap bisa dilakukan, namun pemanggilan argumen dilakukan by value

3. Diberikan program berikut, tebaklah outputnya!

```

#include <stdio.h>

int a;
void f_1(int a)
{
    printf("Nilai a di fungsi f_1 : %d\n",a);
    {
        int a=4;
        printf("Nilai a di blok f_1 : %d\n",a);
    }
}

void main()
{
    f_1(5);
    printf("Nilai a di global : %d\n",a);
}

```

Pembahasan

Program di atas menggunakan jenis variabel lokal, global dan parameter formal. Dalam sebuah program, jika ada nama variabel yang sama, maka variabel lokal yang diutamakan, kemudian parameter formal, dan akhirnya variabel global. Sehingga keluaran dari program di atas adalah :

Nilai a di fungsi f_1 : 5
Nilai a di blok f_1 : 4
Nilai a di global : 0

```
#include <stdio.h>

int a;
void f_1(int a)
{
    printf("Nilai a di fungsi f_1 : %d\n",a); ←Parameter formal, a=5
    {
        int a=4;
        printf("Nilai a di blok f_1 : %d\n",a); ←Lokal blok, a=4
    }
}

void main()
{
    f_1(5);
    printf("Nilai a di global : %d\n",a); ←Global, a=0
}
```

4. Buatlah fungsi untuk menghitung nilai soft skill, penguasaan bisnis dan nilai akhir proposal PMW berdasarkan kasus pada Bab III, soal latihan nomor 3 (halaman 36). Hitunglah untuk satu orang juri saja!

Pembahasan

Dalam kasus ini, dibutuhkan tiga buah fungsi, yaitu untuk menghitung nilai soft skill, penguasaan bisnis dan nilai akhir. Fungsi yang digunakan akan bertipe float karena akan merupakan hasil perhitungan rata-rata dari keseluruhan input.

```
#include <stdio.h>

float rata_softSkill(int KD, int MK, int KMR)
{
    return (KD+MK+KMR)/3;
}

float rata_penguasaan_bisnis (int pasar, int teknis,
int organisasi,int keuangan)
{
```

```

        return (pasar+ teknis + organisasi + keuangan)/4;
    }

float      nilai_akhir(float      soft_skill,float
penguasaan_bisnis)
{
    return (soft_skill+penguasaan_bisnis)/2;
}

void main()
{
    int  KD,  MK,  KMR,  pasar,  teknis,  organisasi,
keuangan;
    float soft_skill,penguasaan_bisnis;

    printf("Input Penilaian\n");
    printf("Masukkan Nilai Kesesuaian Data :");
    scanf("%d",&KD);
    printf("Masukkan Nilai Motivasi & Kepercayaan
Diri :");
    scanf("%d",&MK);
    printf("Masukkan Nilai Keberanian Mengambil
Resiko :");
    scanf("%d",&KMR);
    printf("Masukkan Nilai Pasar :");
    scanf("%d",&pasar);
    printf("Masukkan Nilai Teknis :");
    scanf("%d",&teknis);
    printf("Masukkan Nilai Organisasi :");
    scanf("%d",&organisasi);
    printf("Masukkan Nilai Keuangan :");
    scanf("%d",&keuangan);

    soft_skill= rata_softSkill(KD,MK,KMR);

```

```
    penguasaan_bisnis=
rata_penguasaan_bisnis(pasar,teknis,organisasi,keuan
gan);

    printf("Nilai akhir      = %.2f",  nilai_akhir
(soft_skill,penguasaan_bisnis));
}
```

6.7 Penutup

Kesimpulan

- Fungsi merupakan suatu bagian dari program yang dimaksudkan untuk mengerjakan suatu tugas tertentu dan letaknya terpisah dari program yang memanggilya.
- Keuntungan menggunakan fungsi : dapat dikerjakan oleh beberapa orang, alur program lebih mudah dimengerti, lebih mudah dimodifikasi tanpa mempengaruhi modul lain.
- Definisi fungsi terdiri dari tipe kembalian, nama fungsi, daftar parameter dan tubuh fungsi
- Tipe kembalian : sebuah fungsi mungkin akan memberikan nilai kembalian tertentu.
- Nama fungsi : merupakan nama dari suatu fungsi.
- Daftar Parameter : nilai-nilai yang dikirimkan oleh pemanggil fungsi terhadap suatu fungsi.
- Parameter bersifat opsional, sehingga suatu fungsi tidak harus memiliki parameter
- Tubuh fungsi : tubuh fungsi berisi sekumpulan perintah yang mendefinisikan apa yang dilakukan oleh suatu fungsi.
- Suatu fungsi harus dideklarasikan jika dipanggil dari file sumber lain
- Pemanggilan argumen fungsi dapat dilakukan dengan call by value atau call by reference
- Call by value melakukan penyalinan nilai parameter aktual ke dalam parameter formal yang dimiliki fungsi. Perubahan terhadap parameter yang dilakukan di dalam fungsi tidak akan memberikan perubahan pada nilai argumen setelah keluar dari fungsi.
- Call by reference melakukan penyalinan alamat pada sebuah argumen ke dalam parameter formal fungsi. Hal ini berarti setiap perubahan terhadap parameter akan berpengaruh pada argumen.
- Berdasarkan cakupannya, variabel di dalam Bahasa C ada tiga yaitu variabel lokal, variabel global, dan parameter formal

- Variabel lokal dideklarasikan di dalam blok fungsi atau blok. Cakupannya hanya di dalam fungsi atau blok dimana variabel tersebut dideklarasikan.
- Variabel global dideklarasikan di luar blok fungsi, biasanya di bagian atas program. Cakupannya adalah seluruh program
- Parameter formal dideklarasikan sebagai parameter fungsi. Cakupannya hanya di dalam fungsi itu saja.
- Urutan penggunaan variabel jika ada nama variabel yang sama dalam setiap cakupan adalah : variabel lokal, parameter formal dan variabel global
- Variabel lokal yang tidak diberikan nilai inisial oleh pemrogram akan mengambil nilai sebelumnya pada memorinya sedangkan variabel global akan diberi nilai inisial oleh sistem. Int, float dan double diberi nilai 0. Sedangkan char diberikan nilai '\0', dan pointer diberi nilai null
- Rekursi adalah suatu proses dari fungsi yang memanggil dirinya sendiri secara berulang-ulang.
- Dalam fungsi rekursi harus ada suatu kondisi yang mengakhiri prosesnya

Latihan

1. Buatlah program untuk melakukan penjumlahan bilangan dari 1 sampai dengan n, gunakanlah fungsi untuk melakukan penjumlahan tersebut!
2. Buatlah program menggunakan fungsi untuk mengelola sebuah tabungan dengan menu : Tabung, Tarik, Cek Saldo dan Keluar. Saldo awal dianggap nol. Menu akan selalu ditampilkan sampai pengguna memasukkan Pilihan Keluar. Jika menu pilih dipilih maka akan meminta input jumlah tabungan dan saldo akan bertambah. Jika menu Tarik dipilih maka akan diminta jumlah uang yang ditarik, kemudian akan dicek apakah saldo mencukup. Jika saldo cukup, kurangi saldo, sebaliknya berikan pesan tidak bisa. Cek saldo akan menampilkan saldo tabungan. Menu Tabung, Tarik dan Cek Saldo harus menggunakan fungsi.
3. Apakah keluaran dari program berikut ini :

```
#include <stdio.h>

void cetak(int n)
{
```

```

    if (n>0)
    {
        cetak(n-1);
        printf("%d\n", n);
    }
}

void main()
{
    int n=10;
    cetak(n);
}

```

4. Buatlah program yang memiliki fungsi untuk konversi bilangan desimal ke bilangan biner

Berikut adalah cara konversi desimal ke biner :

pembagi (basis)	nilai	sis
	5	
2	— 1	↑
	2	
2	— 0	
	1	
2	— 1	
	0	

pembagi (basis)	nilai	sis
	53	
2	— 1	↑
	26	
2	— 0	
	13	
2	— 1	
	6	
2	— 0	
	3	
2	— 1	
	1	
2	— 1	
	0	

Baca sisa bagi dari bawah ke atas, sehingga

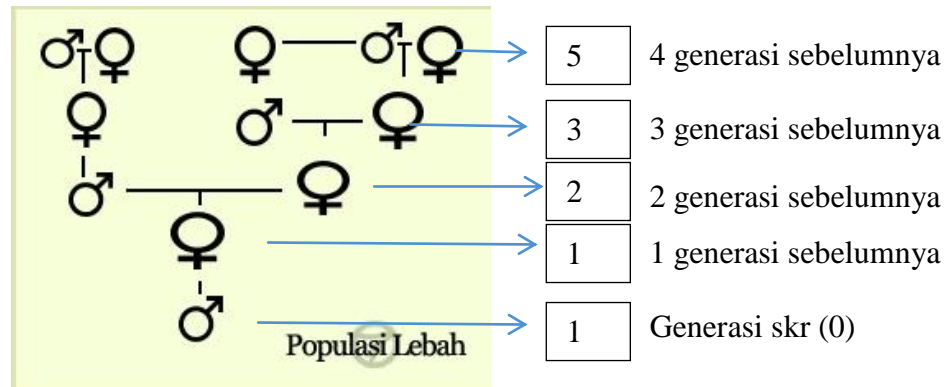
Biner dari 5 adalah 101

Biner dari 53 adalah 110101

5. Buatlah fungsi rekursi untuk kasus berikut :

Lebah adalah hewan yang hidup berkoloni. Ada fakta unik tentang lebah yang tidak diketahui semua orang. Lebah jantan diproduksi dari sel telur ratu yang tidak terbuahi (artinya lebah jantan tidak memiliki Ayah), sedangkan lebah betina dihasilkan dari sel telur yang sudah

dibuahi (lebah betina memiliki Ayah dan Ibu). Jika ditelusuri pohon nenek moyang dari seekor lebah jantan, adalah sebagai berikut :



Contoh Program :

```
Masukkan generasi penelusuran nenek moyang : 1
Jumlah lebah pada 1 generasi sebelumnya adalah : 1
```

```
Masukkan generasi penelusuran nenek moyang : 2
Jumlah lebah pada 2 generasi sebelumnya adalah : 2
```

```
Masukkan generasi penelusuran nenek moyang : 4
Jumlah lebah pada 4 generasi sebelumnya adalah : 5
```

```
Masukkan generasi penelusuran nenek moyang : 5
Jumlah lebah pada 5 generasi sebelumnya adalah : 8
```

Umpan Balik

Setelah menyelesaikan bab ini, mahasiswa diharapkan mampu menggunakan fungsi, baik untuk menyederhanakan program maupun untuk kebutuhan lainnya. Mahasiswa diharapkan mampu memberikan nilai kembalian yang tepat dan argumen yang sesuai dengan kasus yang diberikan. Mahasiswa diharapkan mampu menggunakan fungsi rekursi dan menentukan kapan fungsi tersebut akan berakhir. Pemahaman akan bab ini sangat penting untuk dapat melanjutkan pada bab berikutnya.

BAB VII

ARRAY

Standar Kompetensi

Mahasiswa diharapkan mampu menerjemahkan permasalahan dalam dunia nyata ke dalam algoritma dan mengimplementasikannya ke dalam Bahasa C setelah mempelajari mata kuliah ini.

Kompetensi Dasar

Setelah mempelajari materi ini, mahasiswa diharapkan mampu menggunakan variabel berindeks (*array*) dalam Bahasa C.

Indikator Pencapaian Kompetensi

Adapun indikator hasil belajar adalah sebagai berikut :

1. Mahasiswa mampu melakukan deklarasi array baik satu maupun dua dimensi
2. Mahasiswa mampu melakukan insialisasi array dengan berbagai cara baik satu maupun dua dimensi
3. Mahasiswa mampu melakukan pengaksesan array baik satu maupun dua dimensi
4. Mahasiswa mampu membuat program dengan melewati array ke dalam fungsi

MATERI

7.1 Definisi Array

Array atau larik merupakan struktur data yang menyimpan sekumpulan data dengan tipe yang sama dan ukuran tertentu yang fiks. Saat menyimpan 10 bilangan bulat, akan lebih efisien menyimpannya ke dalam array daripada memberinya nama dengan *bilangan1*, *bilangan2*, ... dan *bilangan10*. Dengan

penyimpanan ke dalam variabel array, maka hanya dibutuhkan satu variabel saja. Setiap elemen di dalam array akan diakses dengan menggunakan indeks tertentu. Semua elemen array akan menempati lokasi yang berurutan di dalam memori. Data paling pertama akan disimpan pada alamat paling rendah, dan demikian juga dengan data terakhir akan memiliki alamat memori terbesar.

7.2 Array Satu Dimensi

Bentuk array paling sederhana adalah array satu dimensi. Adapun cara mendeklarasikan array satu dimensi adalah sebagai berikut :

```
tipe_data nama_variabel[panjang];
```

Dimana :

- Tipe_data : tipe data dari variabel yang ingin diinput
- Nama_variabel : nama variabel array
- panjang : Panjang maksimal variabel array

Contoh :

```
int bilangan[4];
```

Source code di atas adalah source code untuk membuat sebuah array bernama bilangan dengan tipe integer dan panjang 4 elemen. Inisialisasi array dapat dilakukan dengan beberapa macam cara yaitu :

- Pemberian nilai langsung pada setiap elemen array sesuai dengan jumlah elemennya

```
int bilangan[4] = {3, 56, 45, 7};
```

Pendeklarasian di atas memesan empat alamat untuk menyimpan data, dimana data pertama akan diisi 3, data kedua 56 dan seterusnya. Jumlah bilangan yang dimasukkan ke dalam kurung kurawal memiliki jumlah yang sama dengan ukuran array bilangan.

- Pemberian nilai pada array langsung pada elemen array yang ukurannya tidak didefinisikan

```
int bilangan[] = {3, 56, 45, 7};
```

Pendeklarasian di atas tidak menentukan ukuran array secara spesifik, namun dengan meletakkan empat angka ke dalam kurung kurawal, berarti ukuran array yang dipesan adalah sepanjang empat. Data pertama akan diisi 3, data kedua 56 dan seterusnya.

- Pemberian nilai pada elemen tertentu

Pemberian nilai pada elemen tertentu adalah dengan cara menuliskan indeks di dalam kurung siku dan kemudian memberikan nilainya di kanan tanda sama dengan. Adapun pemberian nilai pada suatu elemen adalah sebagai berikut :

```
nama_variabel[index] =nilai;
```

Dimana

- index : urutan elemen dari array (dimulai dari 0)
- Nilai : nilai yang akan diberikan kepada elemen array ke-index+1

Contoh :

```
bilangan[0]=8;  
bilangan[3]=2;
```

Hal yang perlu diingat dalam array adalah indeksnya dimulai dari nol. Sehingga indeks array terbesar adalah jumlah elemen dikurangi satu. Contoh di atas memberikan nilai 8 kepada elemen array ke-1 dan memberikan nilai 2 pada elemen ke-4.

Pengaksesan elemen array adalah sebagai berikut :

```
nama_variabel[index];
```

Dimana :

- index : urutan elemen dari array (dimulai dari 0)

Contoh :

```
printf("%d",bilangan[0]);
```

Contoh di atas akan mencetak array bilangan pada elemen array pertama.

Sebuah array satu dimensi bisa digambarkan seperti berikut ini :

0	1	2	3
3	56	45	7

Elemen pertama memiliki indeks 0 dan diberi nilai 3, sedangkan elemen kedua berindeks 1 diberi nilai 56 dan seterusnya. Contoh program yang menggunakan array satu dimensi adalah sebagai berikut :

```
#include <stdio.h>

main()
{
    int bilangan[5],i;
    for(i=0;i<5;i++)
    {
        printf("Masukkan bilangan ke %d : ",i+1);
        scanf("%d",&bilangan[i]);
    }
    /*cetak terbalik*/
    for(i=4;i>=0;i--)
    {
        printf("Bilangan ke %d : %d\n",i+1,bilangan[i]);
    }
}
```

Program di atas menggunakan array satu dimensi bertipe integer dengan ukuran 5. Nama variabel array adalah bilangan. Perulangan pertama untuk melakukan input dari setiap elemen array mulai dari elemen pertama sampai elemen terakhir. Sedangkan perulangan kedua melakukan pencetakan elemen array mulai dari elemen terakhir sampai dengan elemen pertama. Adapun keluaran program tersebut adalah sebagai berikut :

```
Masukkan bilangan ke 1 : 3
Masukkan bilangan ke 2 : 13
Masukkan bilangan ke 3 : 24
Masukkan bilangan ke 4 : 2
Masukkan bilangan ke 5 : 6
Bilangan ke 5 : 6
Bilangan ke 4 : 2
Bilangan ke 3 : 24
Bilangan ke 2 : 13
Bilangan ke 1 : 3
```

Pada program di atas, misalnya pengguna melakukan input terhadap lima bilangan yang diminta, maka program akan menghasilkan keluaran berupa nilai yang diinput secara terbalik.

7.3 Array Dua Dimensi

Array 2 dimensi adalah bentuk array multi dimensi yang paling sederhana. Array 2 dimensi pada dasarnya adalah kumpulan dari array satu dimensi. Informasi yang terkandung di dalam array, diatur dalam baris dan kolom.

Pendeklarasian array 2 dimensi adalah sebagai berikut :

```
tipe_data nama_variabel[baris][kolom];
```

Dimana :

- Tipe_data : tipe data dari variabel yang ingin diinput
- Nama_variabel : nama variabel array
- Baris : Panjang maksimal baris variabel array
- Kolom : Panjang maksimal kolom dari setiap baris array

Contoh berikut adalah ilustrasi array dua dimensi dengan 3 baris dan 4 kolom. Nama array adalah a. Indeks array dipanggil mulai dari barisnya kemudian diikuti oleh kolomnya. Misalnya jika ingin mengetahui data pada baris ke dua, kolom ke tiga, maka digunakan cara a[1][2]. Indeks pada array dua dimensi juga dimulai dari nol.

	Kolom 1	Kolom 2	Kolom 3	Kolom 4
Baris 1	a[0][0]	a[0][1]	a[0][2]	a[0][[3]
Baris 2	a[1][0]	a[1][1]	a[1][[2]	a[1][[3]
Baris 3	a[2][0]	a[2][1]	a[2][[2]	a[2][[3]

Cara memberikan nilai pada array dua dimensi bisa dilakukan dengan cara berikut :

```
int a[3][4] = {  
    {0, 1, 2, 3} , /* inisialisasi pada indeks 0 */  
    {4, 5, 6, 7} , /* inisialisasi pada indeks 1 */  
    {8, 9, 10, 11} /* inisialisasi pada indeks 2 */  
};
```

Dengan cara di atas, maka setiap elemen array akan berisi nilai sesuai dengan yang telah didefinisikan. Array diberikan nilai per baris. Baris pertama akan terdiri dari 4 sub elemen yang diletakkan di dalam kurung kurawal { dan }. Sedangkan untuk mengisi elemen pada baris berikutnya hanya perlu dipisahkan dengan tanda koma dan kemudian dilanjutkan dengan pemberian nilai yang sama.

Cara lainnya bisa dilakukan dengan menuliskan kode berikut ini :

```
int a[3][4] = {0,1,2,3,4,5,6,7,8,9,10,11};
```

Cara di atas akan secara langsung memberikan empat elemen pertama pada setiap kolom dari baris pertama dan kemudian 4 nilai berikutnya pada setiap kolom baris kedua dan seterusnya.

Pemberian nilai pada elemen dengan indeks tertentu dilakukan dengan cara berikut :

```
int a[1][2] = 3;
```

Perintah di atas memberikan nilai 3 pada elemen array a pada baris ke dua dan kolom ketiga. Pengaksesan elemen array adalah sebagai berikut :

```
Nama_variabel[row_index][col_index];
```

Dimana :

- row_index : urutan baris dari array (dimulai dari 0)
- col_index : urutan kolom dari array elemen utama ke row_index

Contoh :

```
printf("%d",bilangan[0][1]);
```

Contoh di atas akan mencetak array bilangan pada baris pertama kolom ke 2.

Contoh program yang menggunakan array dua dimensi adalah sebagai berikut :

```
#include <stdio.h>

void main()
{
    int matrix1[2][3]={2,4,6},{1,0,4};
    int matrix2[2][3]={1,2,3},{2,1,0};
    int matrix3[2][3];

    int i,j;
    for(i=0;i<2;i++)
    {
        for(j=0;j<3;j++)
        {
            matrix3[i][j]=matrix1[i][j]+matrix2[i][j];
        }
    }

    /*cetak hasil penjumlahan*/
    for(i=0;i<2;i++)
    {
        for(j=0;j<3;j++)
        {
```

```

        printf("%3d ",matrix3[i][j]);
    }
    printf("\n");
}

```

Program di atas melakukan pendeklarasian tiga buah array dua dimensi. Ketiga array tersebut adalah matriks bernama matriks1, matriks2 dan matriks3. Matriks1 dan matriks2 diberi nilai inisial, sedangkan matriks3 diberikan merupakan hasil penjumlahan dari matriks1 dengan matriks2. Perulangan yang dibutuhkan adalah perulangan bersarang, dimana perulangan induk (i) mewakili baris, sedangkan perulangan di dalamnya (j) mewakili kolom.

7.4 Melewatkan Array pada Fungsi

Jika sebuah array satu dimensi dikirimkan sebagai argumen dalam sebuah fungsi, maka parameter formal harus dideklarasikan dengan salah satu dari tiga cara berikut ini :

- Parameter formal sebagai pointer

Cara pengiriman bisa dilakukan dengan cara berikut :

```

tipe_data nama_fungsi(int *param)
{
    //Tubuh fungsi ada di sini
}

```

- Parameter formal sebagai array dengan panjang tertentu

Cara pengiriman bisa dilakukan dengan cara berikut :

```

tipe_data nama_fungsi(int param[10])
{
    //Tubuh fungsi ada di sini
}

```

- Parameter formal sebagai array tanpa panjang

Cara pengiriman bisa dilakukan dengan cara berikut :

```

tipe_data nama_fungsi(int param[])
{
    //Tubuh fungsi ada di sini
}

```

Contoh berikut melewati array sebagai parameter ke dalam fungsi untuk menghitung rata-rata. Adapun array menyimpan sekumpulan bilangan bulat yang akan dirata-ratakan oleh fungsi.

```
#include <stdio.h>

float hitungRata(int arr[], int ukuran)
{
    int i;
    float rata;
    float total=0;
    for (i = 0; i < ukuran; ++i)
    {
        total += arr[i];
    }
    rata = total / ukuran;
    return rata;
}

void main()
{
    int data[5] = {1000, 2, 3, 17, 50};
    double rata;

    rata = hitungRata( data, 5 ) ;

    printf( "Nilai rata-rata : %.2f ", rata );
}
```

7.5 Mengembalikan Array pada Fungsi

Program C tidak mengizinkan pengembalian nilai dalam bentuk array sebagai sebuah argumen. Karena itu, maka bentuk pointer bisa digunakan untuk menspesifikasikan nama array tanpa indeks. Cara pendeklarasiannya adalah sebagai berikut :

```
tipe data * nama_fungsi
{
    //tubuh fungsi
}
```

Contohnya adalah sebuah fungsi yang melakukan pengacakan 10 bilangan random dan disimpan pada sebuah array. Maka kodenya adalah sebagai berikut :

```
#include <stdio.h>

int * acak( )
{
    static int bil[10];
```



```

int i;

srand( (unsigned)time( NULL ) );
for ( i = 0; i < 10; ++i)
{
    bil[i] = rand();
}
return bil;
}

void main()
{
    int *p;
    int i;
    p = acak();
    for ( i = 0; i < 10; i++ )
    {
        printf( "Data %d : %d\n", i+1, *(p + i));
    }
}

```

Satu hal yang perlu diingat ketika mengembalikan array adalah menyatakan variabel pointer sebagai static, karena program C tidak mengizinkan variabel lokal mengembalikan alamat keluar fungsi. Fungsi di atas melakukan pengacakan 10 buah bilangan dan disimpan kepada array bil. Nilai kembalian berupa pointer akan ditangkap oleh sebuah variabel pointer dalam program pemanggilnya. Cara mengaksesnya adalah dengan menuliskan *p untuk data pertama, kemudian dilanjutkan dengan *p+1 untuk data kedua dan seterusnya, sampai *p+9 untuk data ke 10.

7.6 Contoh Soal

Sesi berikut ini akan memberikan beberapa contoh kasus yang berkaitan dengan array dan pembahasannya.

1. Buatlah program untuk meminta input 10 buah data dari pengguna, dan kemudian menghitung perbedaan setiap data dengan rata-ratanya.

Pembahasan

Untuk membuat program di atas, dibutuhkan penyimpanan dalam bentuk array dan kemudian menghitung rata-rata semua data di dalam array. Setelah rata-rata dihitung, kemudian selisih setiap data dengan rata-rata mulai bisa dihitung. Berikut ini adalah kode program dari kasus di atas :

```

#include <stdio.h>

void main()
{
    int bil[10];
    int i;
    float total=0,rata;
    for(i=0;i<10;i++)
    {
        printf( "Masukkan data ke %d : ", i+1);
        scanf("%d",&bil[i]);
        total +=bil[i];
    }
    rata=total/10; ← Penghitungan rata-rata

    for ( i = 0; i < 10; i++ )
    {
        float selisih = fabs(bil[i]-rata);
        printf( "Selisih %d : %f\n", i+1, selisih);
    }
}

```

Pentotalan
bilangan yang
diinput

Penghitungan
selisih

Berikut ini adalah output yang dihasilkan oleh program di atas. Adapun fungsi fabs merupakan fungsi untuk mengembalikan nilai absolut atau nilai positif dari selisih setiap data dengan rata-rata.

```

"D:\Lecturer Materials\2015... -
Masukkan data ke 1 : 4
Masukkan data ke 2 : 6
Masukkan data ke 3 : 1
Masukkan data ke 4 : 8
Masukkan data ke 5 : 9
Masukkan data ke 6 : 3
Masukkan data ke 7 : 7
Masukkan data ke 8 : 3
Masukkan data ke 9 : 1
Masukkan data ke 10 : 8
Selisih 1 : 1.000000
Selisih 2 : 1.000000
Selisih 3 : 4.000000
Selisih 4 : 3.000000
Selisih 5 : 4.000000
Selisih 6 : 2.000000
Selisih 7 : 2.000000
Selisih 8 : 2.000000
Selisih 9 : 4.000000
Selisih 10 : 3.000000

```

Gambar 7.1. Keluaran Program Selisih

2. Apakah yang dilakukan oleh program berikut ini?

```
#include <stdio.h>

main()
{
    char kata[5]={'H','I','D','U','P'};
    int i;

    for(i=4;i>=0;i--)
    {
        printf("%c",kata[i]);
    }
}
```

Pembahasan

Program di atas melakukan pembentukan array bertipe char dengan ukuran 5. Array bernama kata akan diberikan nilai inisial yaitu HIDUP. Program di atas mencetak elemen array mulai dari elemen terakhir sampai elemen ke satu. Adapun keluaran program di atas adalah sebagai berikut :

```
PUDIH
```

3. Buatlah program untuk melakukan pencarian angka 5 pada n buah elemen array. Nilai n diinput oleh pengguna dan nilai data juga diinput oleh pengguna.

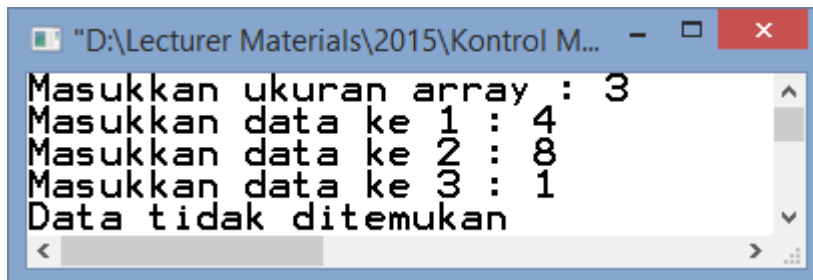
Pembahasan

Untuk melakukan hal di atas, maka perlu dilakukan pendeklarasian bilangan bulat n dan meminta input bilangan n. Kemudian mendeklarasikan array bilangan bulat dengan ukuran n. Kemudian masuk ke dalam perulangan untuk melakukan input data bilangan sebanyak n buah. Setelah semua data diinput, maka pencarian bisa dilakukan dengan melakukan perulangan dan mencocokkan semua data pada array satu-pesatu dengan bilangan yang dicari. Jika ada data yang sama, maka program akan memberikan pesan bahwa data ditemukan dan program akan berhenti melakukan perulangan dengan perintah break. Sebaliknya jika data tidak ditemukan, maka elemen array akan ditelusuri semua dan nilai pencacah pastinya akan lebih besar daripada indeks array yang terakhir. Oleh sebab itu di bagian bawah perulangan harus dilakukan

pengecekan apakah indeks array mencapai n, jika demikian, maka berarti data tidak ditemukan. Berikut ini adalah kode programnya :

```
#include <stdio.h>
void main()
{
    int i,n,cari=5;
    printf("Masukkan ukuran array : ");
    scanf("%d",&n);
    int bil[n];
    for(i=0;i<n;i++)
    {
        printf("Masukkan data ke %d : ", i+1);
        scanf("%d",&bil[i]);
    }
    for ( i = 0; i < n; i++ )
    {
        if(bil[i]==cari)
        {
            printf("Data pada urutan %d\n",i+1);
            break;
        }
    }
    if(i>=n)
        printf("Data tidak ditemukan");
}
```

Berikut adalah contoh keluarannya :



```
"D:\Lecturer Materials\2015\Kontrol M... - [X]
Masukkan ukuran array : 3
Masukkan data ke 1 : 4
Masukkan data ke 2 : 8
Masukkan data ke 3 : 1
Data tidak ditemukan
```

Pertanyaan

Apakah yang terjadi jika break pada program dihilangkan? Apakah terdapat perbedaan pada output? Jika ada, jelaskan! Jika tidak, jelaskan untuk apa break tersebut ditulis di sana!

7.7 Penutup

Kesimpulan

- Array atau larik merupakan struktur data yang menyimpan sekumpulan data dengan tipe yang sama dan ukuran tertentu yang fiks.
- Semua elemen array akan menempati lokasi yang berurutan di dalam memori. Data paling pertama akan disimpan pada alamat paling rendah, dan demikian juga dengan data terakhir akan memiliki alamat memori terbesar.
- Cara pendeklarasian array satu dimensi adalah `tipe_data nama_variabel[panjang];`
- Cara mengakses elemen array adalah dengan menuliskan nama variabel array-nya dan kemudian dilanjutkan dengan indeksnya di dalam kurung siku [dan]
- Indeks array dimulai dari nol dan maksimal jumlah data dikurangi satu
- Array dua dimensi adalah kumpulan dari array satu dimensi
- Cara pendeklarasian array dua dimensi adalah `tipe_data nama_variabel[baris][kolom];`
- Sebuah array bisa dilewatkan sebagai parameter fungsi dengan tiga cara yaitu : mengirimnya sebagai pointer, mengirimnya sebagai array dengan ukuran atau mengirimnya dengan array tanpa ukuran
- Sebuah array bisa menjadi kembalian dari fungsi dengan cara menyatakan tipe kembalian dari suatu fungsi sebagai sebuah pointer dan membentuk variabel lokal array dalam fungsi sebagai variabel static.
- Cara pengaksesan variabel array yang disimpan dalam sebuah pointer adalah dengan memanggil pointer tersebut dengan simbol bintang di depannya untuk data pertama, dan ditambah 1 untuk data berikutnya.

Latihan

1. Buatlah program untuk menginput elemen array dan menentukan elemen terbesar di dalam array!

2. Buatlah program untuk meminta n buah bilangan dari pengguna dan kemudian mengurutkan bilangan dari kecil ke besar!
3. Buatlah program untuk melakukan input array 1 dimensi bertipe integer. Kemudian mintalah user untuk menginput sebuah threshold. Lalu tampilkanlah berapa jumlah data yang nilainya lebih besar daripada threshold!

Contoh :

Input Panjang Data : 3

Data ke-1 : 23

Data ke-2 : 4

Data ke-3 : 7

Input Threshold : 7

Jumlah data di atas threshold : 1

4. Buatlah program untuk membalik nilai suatu array. Pastikan nilai array memang terbalik, bukan hanya tampilan saja!

Contoh :

Input Panjang Data : 4

Data ke-1 : 23

Data ke-2 : 4

Data ke-3 : 7

Data ke-4 : 1

Data setelah dibalik

Data ke-1 : 1

Data ke-2 : 7

Data ke-3 : 4

Data ke-4 : 23

5. Buatlah program untuk melakukan pencarian di dalam array, dimana program akan menampilkan indeks array dimana data ditemukan!

Contoh :

Input Panjang Data : 4

Data ke-1 : 23

Data ke-2 : 4

Data ke-3 : 7

Data ke-4 : 4

Input data yang dicari : 4

Data ditemukan pada index : 2, 4

6. Buatlah program untuk melakukan transpose matriks!
7. Buatlah program untuk mengalikan matriks! Baris dan kolom dari masing-masing matriks diinput oleh user. Ingat syarat perkalian matriks adalah kolom dari matriks pertama harus sama dengan jumlah kolom dari matriks kedua.
8. Buatlah program untuk mengubah sebuah array 2 dimensi menjadi array 1 dimensi, dengan cara sebagai berikut :

Diinputkan array 2 dimensi dengan elemen sebagai berikut :

3	5	2	1
7	4	1	9
8	12	0	3

Cara mengubahnya adalah baca ke samping lalu lanjutkan ke bawah, sehingga bentuk array 1 dimensi adalah :

3	5	2	1	7	4	1	9	8	12	0	3
Baris 1				Baris 2				Baris 3			

Jumlah baris dan kolom untuk array adalah sesuai dengan input pengguna

Umpan Balik

Setelah menyelesaikan bab ini, mahasiswa diharapkan mampu menggunakan struktur data array guna menyimpan sekumpulan data. Mahasiswa diharapkan mampu menentukan apakah suatu data dapat disimpan ke dalam array atau tidak. Mahasiswa diharapkan mampu menggunakan array satu dimensi atau dua dimensi untuk menyelesaikan kasus yang diberikan. Mahasiswa juga diharapkan mampu menggunakan fungsi dengan parameter sebuah array. Pemahaman akan bab ini sangat dibutuhkan agar mahasiswa dapat melanjutkan mempelajari bab berikutnya.

BAB VIII

STRING

Standar Kompetensi

Mahasiswa diharapkan mampu menerjemahkan permasalahan dalam dunia nyata ke dalam algoritma dan mengimplementasikannya ke dalam Bahasa C setelah mempelajari mata kuliah ini.

Kompetensi Dasar

Setelah mempelajari materi ini, mahasiswa diharapkan mampu menggunakan string dan memanipulasinya dalam Bahasa C.

Indikator Pencapaian Kompetensi

Adapun indikator hasil belajar adalah sebagai berikut :

1. Mahasiswa mampu melakukan deklarasi dan input string
2. Mahasiswa mampu menggunakan fungsi-fungsi string yang disediakan oleh bahasa C

MATERI

8.1 String

String merupakan untaian karakter. Dalam pemrograman C, tidak terdapat tipe data string. Di dalam C, string dianggap sebagai sebuah array dengan tipe karakter. String sebenarnya adalah array berdimensi satu bertipe karakter yang diakhiri oleh karakter null '\0'. Cara mendeklarasikan sebuah string adalah sebagai berikut :

```
char nama_variabel[panjang];
```

Adapun cara pemberian nilai inisial pada suatu string adalah sebagai berikut :

```
//cara 1  
char salam[6] = {'H', 'e', 'l', 'l', 'o', '\0'};
```

```
//cara 2
char salam[] = "Hello";
```

Cara melakukan pencetakan string adalah sebagai berikut

```
char salam[6] = {'H', 'e', 'l', 'l', 'o', '\0'};
printf("Salam : %s\n", salam);
```

Keluaran dari perintah di atas adalah “Hello”, string kosong di belakang tidak akan tampil.

Adapun cara untuk melakukan input string sedikit berbeda dengan cara input variabel lainnya. Contoh berikut ini adalah cara melakukan input variabel string

```
#include <stdio.h>

void main()
{
    char nama[30];
    printf("Masukkan nama : ");
    scanf("%s", &nama);

    printf("%s", nama);
}
```

Pertanyaan

Coba input sebuah string tanpa spasi, apa hasilnya? Kemudian inputkanlah string dengan spasi dan lihatlah apa hasilnya?

Jika `scanf` digunakan untuk melakukan input string, maka string yang diinputkan tidak akan bisa menampilkan input berspasi. Oleh sebab itu bisa digunakan perintah `gets`. Adapun contohnya adalah sebagai berikut :

```
#include <stdio.h>

void main()
{
    char nama[30];
    printf("Masukkan nama : ");
    gets(nama);

    printf("%s", nama);
}
```

8.2 Fungsi String

Program C menyediakan beberapa fungsi khusus untuk memanipulasi string. Semua fungsi khusus tersebut berada di dalam library `string.h`, sehingga untuk menggunakan fungsi-fungsi tersebut, maka pemrogram harus menambahkan library `string.h` ke dalam programnya. Beberapa fungsi string yang umum digunakan ditunjukkan pada Tabel 8.1

Tabel 8.1. Daftar Fungsi String Umum

Fungsi	Kegunaan
<code>strcpy(dest,src)</code>	Mengkopi string <code>src</code> ke dalam string <code>dest</code>
<code>strncpy(dest,src,n)</code>	Mengkopi string <code>src</code> sepanjang <code>n</code> karakter ke dalam string <code>dest</code>
<code>strcat(s1,s2)</code>	Menggabungkan string <code>s1</code> dan <code>s2</code> dengan hasil <code>s1</code> kemudian dilanjutkan dengan <code>s2</code> . Hasil disimpan pada <code>s1</code> .
<code>strncat(s1,s2,n)</code>	Menggabungkan string <code>s1</code> dan <code>s2</code> sepanjang <code>n</code> karakter dengan hasil <code>s1</code> (full) kemudian dilanjutkan dengan <code>s2</code> sepanjang <code>n</code> karakter. Hasil disimpan pada <code>s1</code> .
<code>strcmp(s1,s2)</code>	Membandingkan string <code>s1</code> dengan <code>s2</code> . Hasilnya 0 jika sama, dan bukan nol, jika berbeda
<code>strlen(s)</code>	Menghitung panjang karakter dari string <code>s</code>

Contoh program yang menggunakan fungsi string adalah sebagai berikut :

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

void main()
{
    char namaDepan[30],namaBelakang[30], nama[60];
    printf("Masukkan nama depan : ");
    gets(namaDepan);
    printf("Masukkan nama belakang : ");
    gets(namaBelakang);

    strcpy(nama,namaDepan);
    strcat(nama," ");
    strcat(nama,namaBelakang);

    printf("Nama : %s, panjang : %d",nama,strlen(nama));
}
```

Program di atas menggunakan tiga buah string yaitu namaDepan, namaBelakang dan nama. Variabel namaDepan dan namaBelakang diinput, sedangkan variabel nama akan menggabungkan keduanya. Langkah pertama adalah menggunakan fungsi strcpy untuk menyalin nilai namaDepan kepada variabel nama. Selanjutnya strcpy tidak bisa digunakan karena jika digunakan maka string yang sebelumnya disimpan pada nama akan hilang. Langkah paling tepat adalah menggabungkan nama dengan nama belakang yaitu dengan menggunakan fungsi strcat. Langkah terakhir adalah menghitung panjang string nama dengan fungsi strlen.

8.3 Contoh Soal

Sesi ini akan memberikan contoh beberapa kasus yang berkaitan dengan string, sekaligus membahasnya.

1. Buatlah program untuk melakukan pencetakan string secara terbalik! String akan diinput oleh pengguna.

Pembahasan

Langkah pertama untuk program di atas adalah melakukan deklarasi string, kemudian melakukan perulangan mulai dari panjang string dikurangi satu (karena indeks mulai dari nol) hingga nol, dan kemudian melakukan pencetakan setiap karakternya di dalam perulangan sesuai dengan indeks yang ditunjuk oleh pencacah. Berikut ini adalah kode programnya :

```
void main()
{
    char nama[30];
    int i;
    printf("Masukkan nama : ");
    gets(nama);

    for(i=strlen(nama)-1;i>=0;i--)
    {
        printf("%c",nama[i]);
    }
}
```

2. Buatlah program untuk melakukan input umur dan nama seseorang!

Pembahasan

Pada dasarnya program di atas sederhana, namun ada hal khusus yang terjadi jika melakukan input string setelah tipe data non string. Oleh sebab itu, maka

ditambahkan perintah `stdin(fflush)` sebelum input string. Berikut ini adalah kode programnya :

```
#include <stdio.h>

void main()
{
    char nama[30];
    int umur;

    printf("Masukkan umur : ");
    scanf("%d", &umur);
    fflush(stdin);
    printf("Masukkan nama : ");
    gets(nama);
}
```

Pertanyaan

Apa hal khusus yang terjadi jika `fflush(stdin)` dihilangkan?

8.4 Penutup

Kesimpulan

- Dalam Bahasa C, string sebenarnya adalah array berdimensi satu bertipe karakter yang diakhiri oleh karakter null `'\0'`
- String dideklarasikan dengan cara membuat array bertipe karakter
- Input tipe data string dapat dilakukan dengan perintah `gets`
- Ketika sebuah string diinput setelah input non-string variabel, maka buffer harus dibersihkan dahulu dengan memanggil perintah `fflush(stdin)` sebelum dilakukan input string
- Beberapa perintah untuk memanipulasi string disediakan dalam library `string.h`

Latihan

1. Buatlah program untuk mengecek password yang diinputkan. Keluarkan pesan berhasil apabila password yang dimasukkan adalah kata oyi.
2. Buatlah program untuk menghitung panjang sebuah string, namun tidak termasuk spasi di dalamnya.
Contoh :
String : aku, Panjang : 3
String : aku senang, Panjang : 9
String : aku kamu dan dia, Panjang : 13
3. Buatlah program dengan menggunakan fungsi untuk mengecek apakah suatu substring ditemukan di dalam string lainnya.
Contoh :
String : Aku anak Indonesia
Substring : esia
Substring “esia” ada ada pada string “Aku anak Indonesia”
Atau
String : Aku anak Indonesia
Substring : ramai
Substring “ramai” tidak ada ada pada string “Aku anak Indonesia”
4. Buatlah program untuk meminta input n buah string. Nilai n juga diinput. Kemudian program akan memeberikan keluaran string terpanjang.
Contoh :
Masukkan jumlah string : 3
String 1 : Aku belajar
String 2 : Dimakah Ibu berada?
String 3 : Hello World
String terpanjang : Dimanakah Ibu berada?

Umpan Balik

Setelah menyelesaikan bab ini, mahasiswa diharapkan mampu memahami string, meliputi cara pendeklarasiannya, pengaksesannya dan cara inputnya. Mahasiswa diharapkan mampu menggunakan fungsi string yang telah disediakan di dalam file string.h. Pemahaman akan bab ini sangat penting karena hampir pada setiap kasus akan ditemukan tipe data string, selain itu juga penting untuk dapat melanjutkan mempelajari bab berikutnya.

BAB IX

POINTER

Standar Kompetensi

Mahasiswa diharapkan mampu menerjemahkan permasalahan dalam dunia nyata ke dalam algoritma dan mengimplementasikannya ke dalam Bahasa C setelah mempelajari mata kuliah ini.

Kompetensi Dasar

Setelah mempelajari materi ini, mahasiswa diharapkan mampu memahami dan menggunakan pointer dalam Bahasa C.

Indikator Pencapaian Kompetensi

Adapun indikator hasil belajar adalah sebagai berikut :

1. Mahasiswa mampu menggunakan pointer dalam Bahasa C
2. Mahasiswa mampu memahami null pointer
3. Mahasiswa mampu memahami aritmatik pointer
4. Mahasiswa mampu memahami perbandingan pointer

MATERI

9.1 Pointer

Beberapa program dalam Bahasa C akan menjadi lebih mudah dengan menggunakan pointer, karena dengan pointer dimungkinkan melakukan alokasi memori secara dinamis. Sebuah pointer merupakan variabel yang nilainya berupa alamat dari variabel lain. Pointer harus dideklarasikan terlebih dahulu sebelum digunakan.

Setiap variabel akan tersimpan pada alamat tertentu di memori. Dalam Bahasa C digunakan tanda & untuk menampilkan alamat memori dari suatu variabel. Contohnya bisa dilihat pada program berikut ini :


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

```
void main()
```

```
{
```

```
    int a=5;
```

```
    printf("Nilai a : %d\n",a);
```

```
    printf("Variabel a berada pada alamat : %x",&a);
```

```
}
```

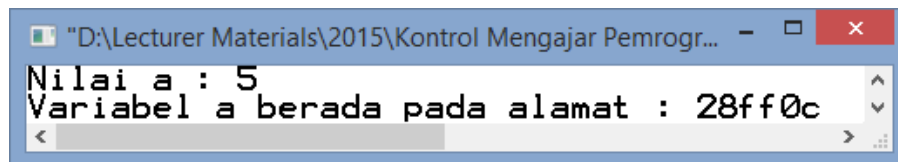
Nilai a



Alamat a



Adapun keluaran dari program di atas adalah nilai variabel a dan alamat penyimpanan variabel a. Nilai variabel a sesuai dengan yang didefinisikan yaitu 5, sedangkan alamatnya sesuai dengan pemesanan pada saat deklarasi. Pemesanan alamat dilakukan otomatis oleh sistem. Pencetakan alamat menggunakan string kontrol %x, agar bisa mencetak nilai dalam format heksadesimal.



Gambar 9.1. Keluaran Program Cetak Alamat

Cara untuk mendeklarasikan sebuah pointer adalah sebagai berikut :

```
tipe_data *nama_pointer;
```

Pada pendeklarasian di atas tipe_data adalah tipe data dalam Bahasa C, sedangkan nama pointer adalah nama pointer yang akan dideklarasikan. Cara pemberian namanya seperti pemberian nama identifier. Pendeklarasian pointer hampir sama dengan pendeklarasian variabel biasa, hanya saja pada pointer terdapat tanda * di depan nama pointer. Tipe data yang dideklarasikan saat membuat pointer adalah sesuai dengan data yang akan ditunjuknya. Nilai dari suatu pointer dengan tipe data apapun adalah sama, yaitu sebuah bilangan heksadesimal yang menyatakan alamat, sedangkan yang berbeda adalah tipe data dari nilai yang ditunjuknya. Adapun contoh pendeklarasian pointer dapat dilihat pada program berikut ini :

```
int *ip /*pointer yang menunjuk data integer*/  
double *dp /*pointer yang menunjuk data double*/  
float *fp /*pointer yang menunjuk data float*/  
char *cp /*pointer yang menunjuk data char*/
```

9.2 Penggunaan Pointer

Terdapat beberapa operasi penting ketika menggunakan pointer, yaitu :

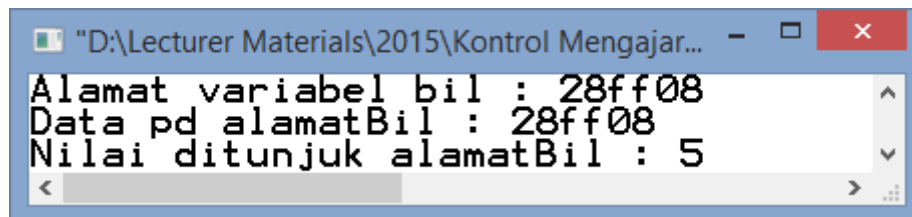
1. Mendefinisikan variabel pointer
2. Memberikan alamat kepada pointer
3. Mengakses nilai pada alamat yang tersedia di dalam variabel pointer

Contoh penggunaan pointer dapat dilihat pada program di bawah ini :

```
#include <stdio.h>

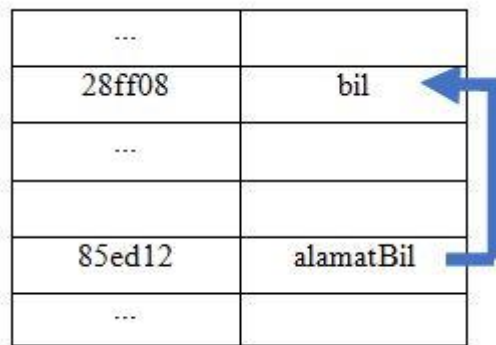
void main()
{
    int bil=5;
    int *alamatBil; /*deklarasi pointer*/
    alamatBil=&bil; /*pemberian nilai pada pointer*/
    printf("Alamat variabel bil : %x\n",&bil);
    printf("Data pd alamatBil : %x\n",alamatBil);
    printf("Nilai ditunjuk alamatBil : %d\n",*alamatBil);
}
```

Program di atas melakukan pendeklarasian pointer dan kemudian memberikan nilai berupa alamat variabel bil. Pencetakan suatu pointer akan langsung mencetak alamat yang ditunjuknya tanpa menggunakan tanda & di depan nama pointer-nya. Untuk mencetak nilai yang ditunjuk pointer maka digunakan tanda * di depan pointer-nya. Adapun keluaran dari program di atas adalah



Gambar 9.2. Keluaran Program Pointer

Ilustrasi dari penggunaan pointer seperti program di atas ditunjukkan pada Gambar 9.3. Ada dua variabel yang disimpan di dalam memori, yaitu bil disimpan pada alamat 28ff08 dan alamatBil disimpan pada alamat 85ed12. Pointer alamatBil menunjuk pada variabel bil, sehingga nilai alamatBil adalah 28ff08, sedangkan untuk alamat dari alamatBil adalah di tempat yang berbeda yaitu 85ed12.



Gambar 9.3. Ilustasi Pointer

Pertanyaan

Bagaimana jika program di atas ditambah `printf("%x",&alamatBil)`, berapakah keluaran yang dihasilkan?

9.3 Pointer null

Pemberian nilai inisial pada sebuah pointer dapat diberikan null. Null berarti pointer tersebut tidak menunjuk kepada variabel manapun. Nilai alamat yang ditunjuk oleh pointer null adalah 0. Berikut ini adalah contohnya :

```
#include <stdio.h>
int main ()
{
    int *ptr = NULL;
    printf("Nilai ptr : %x\n", ptr );

    return 0;
}
```

Keluaran dari program di atas adalah sebagai berikut :

Nilai ptr : 0

Untuk mengecek apakah sebuah pointer null atau tidak, maka bisa digunakan perintah if, seperti potongan perintah berikut :

```
int *ptr=null;
if(ptr) printf("%x",ptr); /*tidak dilakukan karena false*/
if(!ptr) printf("Pointer kosong"); /*dilakukan karena true*/
```

Keluaran program di atas adalah :

Pointer kosong

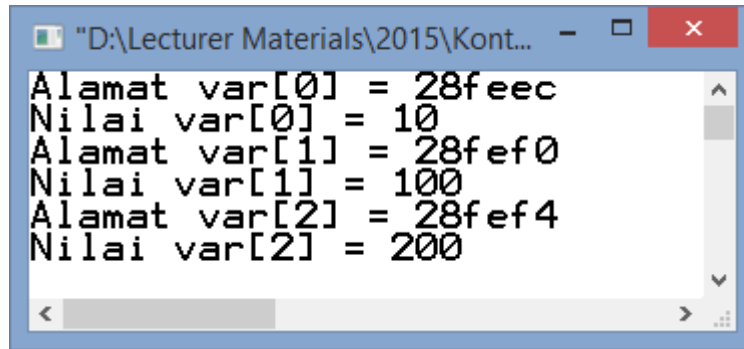
9.4 Aritmatik Pointer

Pointer merupakan alamat dari suatu variabel, sehingga bentuknya adalah bilangan. Oleh sebab itu sebuah pointer mendukung operasi aritmatik. Adapun operato aritmatik yang didukung adalah : ++, --, + dan -. Penggunaan operator ++ misalnya akan menambah pointer sejumlah satu blok alamat. Nilai dari satu blok alamat akan menyesuaikan dengan banyaknya memori yang dihabiskan oleh tipe data pointer. Misalnya jika sebuah tipe data pointer ptr menunjuk alamat memori 100000, maka jika dilakukan ptr++, ptr akan bertambah sejumlah satu blok memori untuk integer yaitu 4 byte, sehingga menjadi 100004. Sedangkan jika ptr adalah sebuah char yang menunjuk alamat 100009, dan dilakukan ptr--, maka akan dikurangi sejumlah 1 blok alamat char yaitu 1 byte, sehingga menjadi 100008.

Berikut ini adalah contoh dari penggunaan aritmatik pointer :

```
#include <stdio.h>
const int MAX = 3;
void main ()
{
    int  var[] = {10, 100, 200}; /*sebuah array*/
    int  i, *ptr;
    ptr = var; /*memberikan array pada pointer*/
    for ( i = 0; i < MAX; i++)
    {
        printf("Alamat var[%d] = %x\n", i, ptr );
        printf("Nilai var[%d] = %d\n", i, *ptr );
        /*menunjuk elemen array berikutnya*/
        ptr++;
    }
}
```

Pada program di atas terdapat array bernama var dengan elemen sebanyak 3 data. Pointer ptr diberikan array var, sehingga pointer ptr akan menunjuk elemen pertama dari array. Perulangan dilakukan untuk mencetak nilai pointer dan nilai yang ditunjuknya. Untuk berpindah ke elemen berikutnya, maka dilakukan operasi penambahan pointer ptr sebesar satu. Adapun keluaran dari program di atas adalah seperti ditunjukkan pada Gambar 9.4



```
"D:\Lecturer Materials\2015\Kont... - [X]
Alamat var[0] = 28feec
Nilai var[0] = 10
Alamat var[1] = 28fef0
Nilai var[1] = 100
Alamat var[2] = 28fef4
Nilai var[2] = 200
```

Gambar 9.4. Keluaran dari Program Aritmatik Pointer

9.5 Perbandingan Pointer

Dua buah pointer bisa dibandingkan dengan cara menggunakan operator perbandingan seperti pada tipe data bilangan. Misalnya ($\text{ptr1} \leq \text{ptr2}$) akan mengembalikan nilai true jika alamat yang ditunjuk ptr1 lebih kecil daripada ptr2. Contoh program di bawah ini melakukan pencetakan elemen array dengan operator perbandingan. Perulangan dilakukan dengan menggunakan while dan meningkatkan nilai ptr sebesar satu blok setiap kali perulangan berjalan. Perulangan akan selalu dilakukan selama ptr masih belum melebihi blok memori dari array yang terakhir. Keluran dari program berikut ini sama dengan Gambar 9.4.

```
#include <stdio.h>
const int MAX = 3;
void main ()
{
    int var[] = {10, 100, 200}; /*sebuah array*/
    int i, *ptr;
    ptr = var; /*memberikan array pada pointer*/
    i=1;
    while( ptr<= &var[MAX-1])
    {
        printf("Alamat var[%d] = %x\n", i, ptr );
        printf("Nilai var[%d] = %d\n", i, *ptr );
        /*menunjuk elemen array berikutnya*/
        ptr++;
        i++;
    }
}
```

9.6 Contoh Soal

Sesi berikut akan membahas beberapa contoh soal yang berkaitan dengan pointer dan pembahasannya

1. Diberikan potongan program berikut. Apakah nilai variabel akan berubah setelah program dieksekusi? Jika berubah, berapakah nilainya?

```
int a=10, *ptr;  
ptr=&a;  
*ptr=20;
```

Pembahasan

Nilai a awalnya 10, ptr menunjuk a dan kemudian pemanggilan *ptr=20 akan mengubah nilai dari variabel yang ditunjuk ptr, yaitu a. Dengan demikian nilai a akan menjadi 20

2. Berikut ini adalah program untuk meminta input sebuah kata dan menampilkannya secara terbalik dengan pointer. Namun masih ada kesalahan di dalamnya. Coba cek kesalahannya dan perbaikilah!

```
#include <stdio.h>  
#include <string.h>  
  
void main ()  
{  
    char kata[10];  
    int *ptr;  
    printf("Masukkan kata : "); gets(kata);  
    ptr=&kata[strlen(kata)-1];  
    while( ptr>= &kata[0])  
    {  
        printf("%c", *ptr);  
        ptr--;  
    }  
}
```

9.7 Penutup

Kesimpulan

- Pointer merupakan variabel yang nilainya berupa alamat dari variabel lain.
- Dalam Bahasa C digunakan tanda & untuk menampilkan alamat memori dari suatu variabel
- Pointer harus dideklarasikan terlebih dahulu sebelum digunakan.
- Pendeklarasian pointer dilakukan dengan menambahkan tanda * di depan nama pointer.
- Tipe data pada saat pendeklarasian pointer harus sama tipe data variabel yang ditunjukkannya.
- Nilai dari suatu pointer adalah sama yaitu alamat memori dalam bentuk heksadesimal, tanpa mempedulikan tipe data variabel yang diacunya.
- Sebuah pointer null tidak menunjuk pada memori manapun, sehingga nilainya adalah nol
- Pointer mendukung operasi aritmatik ++, --, +, -.
- Operasi aritmatik akan mengubah blok memori sesuai dengan operator dan operannya
- Perbandingan pointer dilakukan dengan menggunakan operator perbandingan

Latihan

1. Diberikan kode program berikut ini. Apakah keluaran a=100? Jelaskan jawaban Anda!

```
#include <stdio.h>

void main ()
{
    int *a, *b, c=100;
    b=&c;
    a=&b;

    printf("%d", *a);
}
```

2. Buatlah program untuk menginput array bilangan berukuran n, dengan n input pengguna. Kemudian cetaklah alamat setiap elemen array!
3. Diberikan variabel dan alamatnya seperti pada tabel berikut ini!

Variabel	Tipe Data	Alamat	Nilai
var1	int	000100	56
var2	char	001000	A
var3	float	010000	5.6
var4	double	100000	8.7

Berapakah hasil dari potongan program berikut ini :

```
int *ip=&var1;
char *cp=&var2;
float *fp=&var3;
double *dp=&var4;

printf("ip : %d\n",*ip);
printf("cp : %c\n",*cp);
printf("fp : %f\n",*fp);
printf("dp : %lf\n",*dp);

ip +=3;
cp--;
fp = fp +5;
dp -=2;

printf("ip = %x\n",ip);
printf("cp = %x\n",cp);
printf("fp = %x\n",fp);
printf("dp = %x\n",dp);

if(cp>=dp)
    printf("Yes");
```

Umpan Balik

Setelah menyelesaikan bab ini, mahasiswa diharapkan mampu memahami pointer dalam C dan mengetahui hasil operasi aritmatik dan perbandingan pointer. Pemahaman akan bab ini diperlukan untuk dapat melanjutkan bab berikutnya.

BAB X

STRUKTUR

Standar Kompetensi

Mahasiswa diharapkan mampu menerjemahkan permasalahan dalam dunia nyata ke dalam algoritma dan mengimplementasikannya ke dalam Bahasa C setelah mempelajari mata kuliah ini.

Kompetensi Dasar

Setelah mempelajari materi ini, mahasiswa diharapkan mampu membuat program dengan melibatkan struktur dalam pemrograman C.

Indikator Pencapaian Kompetensi

Adapun indikator hasil belajar adalah sebagai berikut :

1. Mahasiswa mampu melakukan deklarasi struktur Bahasa C dan memberinya nilai
2. Mahasiswa mampu melakukan pengaksesan sebuah struktur
3. Mahasiswa mampu membuat program yang berisi fungsi dengan argumen berupa struktur
4. Mahasiswa mampu membuat program yang berisi pointer dari struktur dalam Bahasa C
5. Mahasiswa mampu membuat program yang berisi array dari struktur dalam Bahasa C

MATERI

10.1 Struktur

Struktur merupakan tipe bentukan dari pembuat program yang dapat menyimpan sekumpulan data dengan tipe data berbeda. Jika array menyimpan sekumpulan data dengan tipe yang sama, berbeda halnya dengan struktur yang memungkinkan penyimpanan sekumpulan data dengan berbagai tipe. Misalnya

saja jika ingin menyimpan data matakuliah berupa kode, nama dan sks. Kode dan nama bertipe string, sedangkan sks bertipe int.

10.2 Pendefinisian Struktur

Sebuah struktur bisa didefinisikan dengan cara menggunakan kata struct. Struktur akan membentuk sebuah tipe baru yang nantinya dapat dipanggil oleh pengguna seperti pendeklarasian variabel lainnya dalam C. Adapun cara pendeklarasian struktur adalah sebagai berikut :

```
struct nama_struktur
{
    Tipe_data data1;
    Tipe_data data2;
    ...
    Tipe_data data3;
};
```

Kata kunci struct selalu digunakan untuk mendahului deklarasi struktur, kemudian nama_struktur mengikuti aturan penamaan identifier. Tipe data di dalam struktur merupakan tipe data yang ada di dalam C atau bisa tipe data bentukan pemrogram. Contoh pendefinisian struktur adalah seperti berikut :

```
struct Matakuliah
{
    char kode[6];
    char nama[40];
    int sks;
};
```

Cara pemanggilan struktur di dalam suatu fungsi adalah seperti kode berikut, dengan nama_variabel adalah nama variabel yang digunakan untuk mengakses struktur.

```
struct nama_struktur nama_variabel
```

Pengaksesan anggota suatu struktur bisa dilakukan dengan tanda titik (.). Cara melakukan akses anggota variabel struktur bernama matakuliah dapat dilakukan pada contoh berikut :

```
printf("%s", matakuliah.kode);
```

Contoh program penggunaan struktur adalah seperti dicontohkan pada program di bawah ini :

```
#include <stdio.h>

struct Matakuliah
{
    char   kode[6];
    char   nama[40];
    int    sks;
};

int main( )
{
    /* Deklarasi variabel struktur bertipe Matakuliah */
    struct Matakuliah matkul1;

    /* input matkul1 */
    printf("Masukkan kode : "); gets(matkul1.kode);
    printf("Masukkan nama : "); gets(matkul1.nama);
    printf("Masukkan sks : "); scanf("%d",&matkul1.sks);

    /* menampilkan matkul1 */
    printf("Kode   : %s",matkul1.kode);
    printf(", Nama   : %s",matkul1.nama);
    printf(", SKS    : %d\n",matkul1.sks);
}
```

10.3 Pengiriman Struktur Sebagai Argumen

Sebuah struktur dapat dikirim sebagai argumen fungsi dengan cara menuliskan kata kunci struct sebelum menuliskan parameter dalam fungsi. Adapun cara pengiriman struktur ke dalam fungsi adalah sebagai berikut :

```
tipe_kembalian nama_fungsi(struct nama_struct nama_parameter)
```

Apabila program pencetakan data matakuliah pada sub bab sebelumnya ingin dituangkan ke dalam fungsi, maka struktur Matakuliah bisa dikirimkan sebagai argumen. Adapun program pengelolaan matakuliah sebelumnya akan menjadi sebagai berikut :

```
#include <stdio.h>

struct Matakuliah
{
    char   kode[6];
    char   nama[40];
    int    sks;
};
```

```

int main( )
{
    struct Matakuliah matkul1;

    printf("Masukkan kode : "); gets(matkul1.kode);
    printf("Masukkan nama : "); gets(matkul1.nama);
    printf("Masukkan sks : "); scanf("%d", &matkul1.sks);

    cetak(matkul1);
}

void cetak(struct Matakuliah mt)
{
    printf("Kode   : %s", mt.kode);
    printf(", Nama   : %s", mt.nama);
    printf(", SKS    : %d\n", mt.sks);
}

```

10.4 Pointer dari Struktur

Pointer selain dapat digunakan untuk menunjuk sebuah variabel biasa dan array, juga bisa digunakan untuk menunjuk sebuah struktur. Adapun caranya hampir sama seperti pendeklarasian sebuah struktur, hanya saja diisi tanda * di depan pointernya. Berikut ini adalah cara mendeklarasikannya :

```
struct nama_struktur *nama_pointer;
```

Adapun contohnya adalah sebagai berikut :

```
struct Matakuliah *mt_ptr;
```

Pemberian nilai pada suatu pointer dapat dilihat pada contoh berikut :

```

struct Matakuliah *mt_ptr;
struct Matakuliah matkul1;
mt_ptr = &matkul1;

```

Pengaksesan elemen dari sebuah pointer yang menunjuk struktur adalah seperti berikut :

```
printf("%s", mt_ptr->kode);
```

Contoh program yang menggunakan pointer penunjuk struktur dapat dilihat pada contoh berikut :

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

```

struct Matakuliah
{
    char   kode[6];
    char   nama[40];
    int    sks;
};

int main( )
{
    struct Matakuliah matkull;

    input(&matkull);
    cetak(matkull);
}

void cetak(struct Matakuliah matkull)
{
    printf("Kode   : %s",matkull.kode);
    printf(", Nama   : %s",matkull.nama);
    printf(", SKS    : %d\n",matkull.sks);
}

void input(struct Matakuliah *matkull)
{
    printf("Masukkan kode : ");gets(matkull->kode);
    printf("Masukkan nama : ");gets(matkull->nama);
    printf("Masukkan sks : ");scanf("%d",&matkull->sks);
}

```

Pada contoh di atas, input data dituangkan ke dalam sebuah fungsi bernama input. Input data struktur mengubah nilai data dari tidak diberi nilai menjadi sesuai nilai yang diinputkan. Dengan demikian, maka digunakan *call by reference* dengan menggunakan pointer. Cara pengaksesan anggota struktur menjadi berubah dari tanda titik (.) menjadi tanda panah (→).

10.5 Array dari Struktur

Array dapat digunakan bersama dengan struktur jika pemrogram ingin menyimpan sekumpulan struktur. Misalnya jika pemrogram ingin menyimpan data tiga buah mata kuliah, maka bisa digunakan array bertipe matakuliah dengan ukuran 3. Adapun cara deklarasi array dari struktur adalah sebagai berikut :

```

struct nama_struktur variabel[ukuran]

```

Adapun contoh pengimplementasian array dari struktur adalah seperti contoh kode di bawah ini :

```
#include <stdio.h>
#define MAKS 3

struct Matakuliah
{
    char   kode[6];
    char   nama[40];
    int    sks;
};

int main( )
{
    struct Matakuliah matkul[MAKS];
    int i;
    for(i=0;i<MAKS;i++)
    {
        input(&matkul[i]);
    }
    for(i=0;i<MAKS;i++)
    {
        cetak(matkul[i]);
    }
}

void cetak(struct Matakuliah matkul1)
{
    printf("Kode   : %s",matkul1.kode);
    printf(", Nama   : %s",matkul1.nama);
    printf(", SKS    : %d\n",matkul1.sks);
}

void input(struct Matakuliah *matkul1)
{
    printf("Masukkan kode : ");gets(matkul1->kode);
    printf("Masukkan nama : ");gets(matkul1->nama);
    printf("Masukkan sks : ");scanf("%d",&matkul1->sks);
    fflush(stdin);
}
```

Pada contoh di atas terdapat penggunaan array untuk menyimpan tiga buah mata kuliah. Setelah pendeklarasian array struktur matakuliah, dilakukan pemanggilan fungsi input dengan perulangan. Untuk setiap nilai i, dilakukan input data struktur ke-i, dengan pemanggilan `input(&matkul[i]);`. Langkah berikutnya adalah mencetak matakuliah dengan perulangan.

10.5 Contoh Soal

Pada sesi berikut ini diberikan contoh soal yang berkaitan dengan struktur dan pembahasannya.

1. Buatlah sebuah program dengan array struct untuk menerima input 5 orang pegawai, dimana akan disimpan NIP, Nama, Gaji dan Jabatan pegawai!

Pembahasan

Untuk membuat program di atas, maka digunakan array dari struktur, karena data yang disimpan ada 5 buah. Struktur yang dibentuk adalah pegawai dengan anggota nip, nama, gaji dan jabatan. Berikut ini adalah programnya :

```
#include <stdio.h>
#define MAKS 5

struct Pegawai
{
    char   NIP[18];
    char   nama[40];
    int    gaji;
    char   jabatan[30];
};

int main( )
{
    struct Pegawai peg[MAKS];
    int i;
    for(i=0;i<MAKS;i++)
    {
        input(&peg[i]);
    }
    for(i=0;i<MAKS;i++)
    {
        cetak(peg[i]);
    }
}

void cetak(struct Pegawai peg)
{
    printf("NIP   : %s",peg.NIP);
    printf(", Nama   : %s",peg.nama);
    printf(", Gaji   : %d",peg.gaji);
    printf(", Jabatan  : %s\n",peg.jabatan);
}

void input(struct Pegawai *peg)
{
    printf("Masukkan NIP : ");gets(peg->NIP);
    printf("Masukkan nama : ");gets(peg->nama);
```

```

printf("Masukkan gaji : ");scanf("%d",&peg->gaji);
fflush(stdin);
printf("Masukkan jabatan : ");gets(peg->jabatan);
}

```

2. Buatlah program untuk menyimpan 3 data mahasiswa, dimana data yang disimpan adalah nama, nilai quiz, nilai uts, nilai uas, nilai akhir. Nilai akhir dihitung dari $30\% \text{ quiz} + 35\% \text{ uts} + 35\% \text{ uas}$.

Pembahasan

Program di atas melibatkan array dan struktur seperti pada soal sebelumnya, namun pada soal ini terdapat operasi aritmatika dalam penghitungan nilai akhir. Input ke dalam program hanya nama, nilai quiz, nilai uts, dan nilai uas.

Kode dari kasus di atas adalah sebagai berikut :

```

#include <stdio.h>
#define MAKS 3

struct Mahasiswa
{
    char  nama[40];
    int   n_Quiz,n_UTS,n_UAS;
    float n_Akhir;
};

int main( )
{
    struct Mahasiswa mhs[MAKS];
    int i;
    for(i=0;i<MAKS;i++)
    {
        input(&mhs[i]);
    }
    for(i=0;i<MAKS;i++)
    {
        cetak(mhs[i]);
    }
}

void cetak(struct Mahasiswa mhs)
{
    printf(" nama   : %s",mhs.nama);
    printf("\n nilai quiz   : %d",mhs.n_Quiz);
    printf("\n nilai UTS   : %d",mhs.n_UTS);
    printf("\n nilai UAS   : %d",mhs.n_UAS);
    printf("\n nilai Akhir   : %.2f\n",mhs.n_Akhir);
}

```



```

void input(struct Mahasiswa *mhs)
{
    printf("Masukkan nama : "); gets(mhs->nama);
    printf("Masukkan nilai quiz : ");
    scanf("%d", &mhs->n_Quiz);
    printf("Masukkan nilai UTS : ");
    scanf("%d", &mhs->n_UTS);
    printf("Masukkan nilai UAS : ");
    scanf("%d", &mhs->n_UAS);
    fflush(stdin);
    mhs->n_Akhir=0.3*mhs->n_Quiz+
                0.35*mhs->n_UTS+0.35*mhs->n_UAS;
}

```

3. Buatlah program untuk menyimpan data maksimal 20 proposal PMW. Setiap proposal PMW memiliki detail sebagai berikut :

- Judul
- Besaran Dana
- Pengusul yaitu seluruh mahasiswa pengusul (maks 5 orang) dengan detail :
 - Nama pengusul
 - Jurusan
 - Peran (Ketua/Anggota)
- Nilai dari dua orang juri, sesuai dengan kasus pada Bab III Latihan 3.
- Status dapat ditentukan dengan aturan sebagai berikut :
 - Jika nilai akhir ≥ 5 maka lulus
 - Jika nilai akhir < 5 maka tidak lulus

Pembahasan

Dalam kasus ini akan digunakan perpaduan antara struktur dan array. Struktur yang digunakan adalah struktur bertingkat. Pengusul yang memiliki beberapa detail akan dijadikan struktur sendiri, demikian juga dengan nilai. Baik pengusul maupun nilai memiliki jumlah lebih dari satu, sehingga harus dijadikan array. Struktur utama yang digunakan adalah proposal, dimana di dalamnya terdapat detail proposal termasuk di dalamnya array dari struktur pengusul dan nilai. Agar program lebih mudah dipelihara jika jumlah proposal, pengusul dan juri berubah, maka sebaiknya dibuatkan konstanta untuk maksimal data yang disimpan.

```

#include <stdio.h>
#include <string.h>
#define MAKS_prop 20
#define MAKS_pengusul 5
#define MAKS_juri 2

struct Pengusul
{
    char  nama[40],jurusan[30],peran[10];
};

struct Nilai
{
    char  juri[40];
    int KD, MK, KMR, pasar, teknis, organisasi, keuangan;
    float soft_skill,penguasaan_bisnis,nilai_akhir;
};

struct Proposal
{
    char  judul[100];
    int  dana;
    int  jumlah_pengusul;
    struct Pengusul pengusul[MAKS_pengusul];
    struct Nilai nilai[MAKS_juri];
    float nilai_rata;
    char status[15];
};

void cetak(struct Proposal proposal)
{
    printf("\n Judul   : %s",proposal.judul);
    printf("\n Dana    : %d",proposal.dana);
    printf("\n Pengusul   : ");
    int i;
    for(i=0;i<proposal.jumlah_pengusul;i++)
    {
        printf("\n          Nama                : %s",proposal.pengusul[i].nama);
        printf("\n          Jurusan              : %s",proposal.pengusul[i].jurusan);
        printf("\n          Peran                 : %s",proposal.pengusul[i].peran);
    }
    for(i=0;i<MAKS_juri;i++)
    {
        printf("\n          Nama      Juri      %d      : %s",i+1,proposal.nilai[i].juri);
        printf("\n ASPEK SOFT SKILL :");
        printf("\n          Nilai      Kesesuaian      Data      : %d",proposal.nilai[i].KD);
    }
}

```

```

        printf("\n Nilai Motivasi & Kepercayaan Diri      :
%d",proposal.nilai[i].MK);
        printf("\n Nilai Keberanian Mengambil Resiko      :
%d",proposal.nilai[i].KMR);
        printf("\n          Rerata          Soft          Skill      :
%.2f",proposal.nilai[i].soft_skill);
        printf("\n ASPEK PENGUASAAN BISNIS :");
        printf("\n          Nilai          Pasar              :
%d",proposal.nilai[i].pasar);
        printf("\n          Nilai          Teknis              :
%d",proposal.nilai[i].teknis);
        printf("\n          Nilai          Organisasi          :
%d",proposal.nilai[i].organisasi);
        printf("\n          Nilai          Keuangan          :
%d",proposal.nilai[i].keuangan);
        printf("\n          Rerata          Penguasaan          Bisnis      :
%.2f",proposal.nilai[i].penguasaan_bisnis);
        printf("\n          Nilai          akhir          juri          %d      :
%.2f",i+1,proposal.nilai[i].nilai_akhir);
    }
    printf("\n          nilai          Total              :
%.2f\n",proposal.nilai_rata);
    printf("\n Status      : %s",proposal.status);
}

void input(struct Proposal *proposal)
{
    printf("\nMasukkan Judul : ");gets(proposal->judul);
    printf("Masukkan Dana      :      ");scanf("%d",&proposal-
>dana);

    printf("\n Pengusul      : ");
    int i;
    printf("\n          Masukkan          Jumlah          Pengusul      :
");scanf("%d",&proposal->jumlah_pengusul);
    fflush(stdin);
    for(i=0;i<proposal->jumlah_pengusul;i++)
    {
        printf("Masukkan          Nama          :      ");gets(proposal-
>pengusul[i].nama);
        printf("Masukkan          Jurusan          :      ");gets(proposal-
>pengusul[i].jurusan);
        printf("Masukkan          Peran          :      ");gets(proposal-
>pengusul[i].peran);
    }
    printf("\n Penilaian      : ");
    float total=0;
    for(i=0;i<MAKS_juri;i++)
    {
        printf("\n Masukkan Nama Juri %d      : ",i+1);
        gets(proposal->nilai[i].juri);
        printf("\n ASPEK SOFT SKILL :");
    }
}

```

```

        printf("\n Masukkan Nilai Kesesuaian Data : ");
        scanf("%d",&proposal->nilai[i].KD);
        printf("\n Masukkan Nilai Motivasi & Kepercayaan
Diri : ");
        scanf("%d",&proposal->nilai[i].MK);
        printf("\n Masukkan Nilai Keberanian Mengambil
Resiko : ");
        scanf("%d",&proposal->nilai[i].KMR);

        proposal->nilai[i].soft_skill=(proposal-
>nilai[i].KD+
                                proposal-
>nilai[i].MK+
                                proposal-
>nilai[i].KMR)/3;

        printf("\n ASPEK PENGUASAAN BISNIS :");
        printf("\n Masukkan Nilai Pasar : ");
        scanf("%d",&proposal->nilai[i].pasar);
        printf("\n Masukkan Nilai teknis : ");
        scanf("%d",&proposal->nilai[i].teknis);
        printf("\n Masukkan Nilai organisasi : ");
        scanf("%d",&proposal->nilai[i].organisasi);
        printf("\n Masukkan Nilai keuangan : ");
        scanf("%d",&proposal->nilai[i].keuangan);

        proposal->nilai[i].penguasaan_bisnis=(proposal-
>nilai[i].pasar+
                                proposal-
>nilai[i].teknis+
                                proposal-
>nilai[i].organisasi+
                                proposal-
>nilai[i].keuangan)/4;

        proposal->nilai[i].nilai_akhir=(proposal-
>nilai[i].soft_skill+
                                proposal-
>nilai[i].penguasaan_bisnis)/2;

        total +=proposal->nilai[i].nilai_akhir;
        fflush(stdin);
    }
    proposal->nilai_rata=total/MAKS_juri;

    if(proposal->nilai_rata>=5)
        strcpy(proposal->status,"Lulus");
    else
        strcpy(proposal->status,"Tidak Lulus");
}

int main( )
{

```

```

struct Proposal proposal[MAKS_prop];
int i,jum_proposal;
do{
printf("Masukkan jumlah proposal terdaftar : ");
scanf("%d",&jum_proposal);
fflush(stdin);
}
while(jum_proposal<0 && jum_proposal>20);
for(i=0;i<jum_proposal;i++)
{
    printf("\nDATA PROPOSAL KE %d",i+1);
    input(&proposal[i]);
}
for(i=0;i<jum_proposal;i++)
{
    printf("\nPROPOSAL KE %d",i+1);
    cetak(proposal[i]);
}
}

```

10.6 Penutup

Kesimpulan

- Struktur merupakan tipe bentukan dari pembuat program yang dapat menyimpan sekumpulan data dengan tipe data berbeda.
- Sebuah struktur bisa didefinisikan dengan cara menggunakan kata struct.
- Pengaksesan anggota suatu struktur bisa dilakukan dengan tanda titik (.) untuk variabel non pointer dan panah (→) untuk data pointer
- Sebuah struktur dapat dikirim sebagai argumen fungsi dengan cara menuliskan kata kunci struct sebelum menuliskan parameter dalam fungsi.
- Pointer juga bisa digunakan untuk menunjuk sebuah struktur. Caranya hampir sama seperti pendeklarasian sebuah struktur, hanya saja diisi tanda * di depan pointer-nya.
- Array dapat digunakan bersama dengan struktur jika pemrogram ingin menyimpan sekumpulan struktur.

Latihan

1. Buatlah program untuk mengolah data tabungan dari maksimal 10 data. Data tabungan berupa nomor rekening, pemilik, saldo. Program memiliki menu Menabung, Menarik, Transfer, Cek Saldo dan Keluar. Ketika memilih menabung dan menarik, maka pengguna harus menginput nomor rekening dahulu kemudian baru diminta menginput jumlahnya. Ketika memilih transfer, pengguna harus menginput nomor rekening sumber dan tujuan, kemudian jumlahnya. Ketika memilih cek saldo, pengguna harus menginput nomor rekening. Transaksi penarikan dan transfer harus melalui validasi apakah saldonya mencukupi atau tidak.
2. Buatlah program dengan struktur untuk melakukan pengelolaan transaksi perbelanjaan. Sebuah transaksi terdiri dari n buah item yang dibeli. Nilai n akan diinput oleh pengguna. Sebuah item akan terdiri dari nama barang, harga, jumlah dan subtotal. Subtotal adalah perkalian antara harga dan jumlah.
3. Buatlah program untuk melakukan input 10 buah data buku. Data buku terdiri atas judul, harga dan pengarang. Setelah pengguna melakukan input data, maka lakukan pengurutan berdasarkan harga, mulai dari terendah sampai harga tertinggi.

Umpan Balik

Setelah menyelesaikan bab ini, mahasiswa diharapkan mampu memahami konsep struktur dalam Bahasa C. Mahasiswa diharapkan mampu menentukan apakah suatu kasus dapat diselesaikan dengan struktur atau tidak. Mahasiswa diharapkan mampu membuat program yang melibatkan fungsi dengan struktur sebagai argumennya, memadukan struktur dengan pointer, memadukan struktur dengan array untuk mengolah sekelompok data sesuai dengan kasus yang diberikan. Pemahaman akan bab ini sangat penting, karena bab ini akan membentuk konsep dasar dari matakuliah yang akan didapatkan pada semester berikutnya.

TES SUMATIF

A. Tes Objektif

Pilihlah salah satu jawaban yang menurut Anda paling tepat!

1. Tipe data yang paling tepat untuk menyimpan tipe data bilangan bulat adalah
 - a. char
 - b. int
 - c. float
 - d. string
 - e. boolean
2. File yang dihasilkan setelah melakukan build program berbahasa C memiliki ekstensi
 - a. cpp dan c
 - b. h dan c
 - c. o dan exe
 - d. c dan o
 - e. c dan exe
3. Berikut ini adalah nama variabel yang salah, kecuali
 - a. _Nilai2
 - b. nilai 2
 - c. nilai-2
 - d. int
 - e. 2nilai
4. Untuk menampilkan suatu bilangan bulat dalam simbol oktalesimal akan digunakan string kontrol
 - a. %d
 - b. %D
 - c. %X
 - d. %i
 - e. %H
5. Diketahui potongan program sebagai berikut :

```
a=5,b=7;  
b -= ++a;
```

Berapakah nilai b setelah perintah di atas dijalankan?
 - a. 6
 - b. 5
 - c. 2
 - d. 4
 - e. 1
6. Apakah keluaran program berikut :

```
A=6;  
if (A=7) printf("Yes");  
if (A=6) printf("No");  
else printf("Hi");
```

 - a. Yes
 - b. No
 - c. YesNo
 - d. YesHi
 - e. Tidak ada output
7. Diketahui perintah sebagai berikut :

```
if (!(A>7 || B<=8)) printf("Oyi");
```

```
else printf("Iyo");
```

Manakah dari nilai A dan B berikut yang menyebabkan output program adalah “Oyi”

- a. A=6; B=8 c. A=8;B=7 e. A=7;B=7
- b. A=7;B=10 d. A=8;B=8

8. Diketahui potongan program seperti berikut

```
float i=3;
while(.....)
{
    printf("Ujian\n");
    i=i-1;
}
```

Apakah yang harus diisi pada bagian titik-titik agar program tidak menghasilkan keluaran apapun

- a. i=4 c. i<=3 e. i<4
- b. i!=3 d. i=2

9. Apakah keluaran dari program berikut ini?

```
#include <stdio.h>

int a=4;

void cetak()
{
    printf("%d",++a);
}

main()
{
    int a=2;
    cetak();
}
```

- a. 3 c. 2 e. 5
- b. 4 d. 0

10. Apakah fungsi yang digunakan untuk memberikan nilai kepada sebuah variabel bertipe string?

- a. = c. strcpy e. strlen

- b. strcmp d. strcat

11. Cara untuk melakukan deklarasi array dua dimensi yang benar pada Bahasa C adalah

- a. string nama[10][20] c. char nama{10,20} e. char nama[10,20]
b. char nama[10][20] d. char nama{10}{20}

12. Diketahui perintah seperti berikut :

```
int a,*b;
```

```
b=&a;
```

Maka apakah yang akan tersimpan pada variabel b?

- a. Nilai variabel a c. Alamat variabel b e. Tidak ada jawaban
b. Alamat variabel a d. NULL

13. Diketahui program seperti berikut :

```
void X(int a)
{
    a = a++ *2;
}
main()
{
    int a=3;
    X(a);
    printf("%d",a);
}
```

Berapakah output program?

- a. 3 c. 6 e. 4
b. 8 d. 0

14. Berapakah output program berikut?

```
float X(int a, int b)
{
    if(a>=2)
        return 0.5;
    else if(b<=5)
        return 0.1;
    else
        return X(a+2,b-3);
}
main()
{
    int x=-3,y=13;
```

```

        printf("%f",X(x,y));
    }

```

- a. 0 c. 0.4 d. 0.5
 b. 0.6 d. 0.1

15. Diketahui program seperti berikut, berapakah outputnya?

```

float X(int a, int b)
{
    if(a>=2)
        return 0.5;
    else if(b<=5)
        return 0.1;
    else
        return a+b-X(a+2,b-3);
}

main()
{
    int x=-3,y=13;
    printf("%f",X(x,y));
}

```

- a. 10 c. 10.5 e. 10.1
 b. 9.5 d. 8.5

B. Essay

- Modifikasilah program pada soal latihan no.3, sehingga program mampu menampilkan proposal secara urut berdasarkan nilai akhir dari besar menuju kecil. Selain itu program harus mampu meminta kuota proposal yang lulus kepada user, dan status kelulusan akan ditentukan oleh kuota. Misalnya, jika kuota adalah lima, maka proposal yang statusnya lulus adalah lima proposal dengan nilai akhir terbesar.
- Periksalah dan jelaskan kesalahan pada program berikut ini!

1	#include <studio.h>
2	main
3	{
4	printf("Cek kesalahan saya")
5	int x,y=8,z=9;
6	printf("%d",X);
7	x+;
8	if(y<4 z>7)

9	y=y-1;
10	z=z+1;
11	else
12	y=y+1;
13	z=z-1;
14	}

3. Lakukan tracing pada program berikut, sehingga outputnya dapat diketahui!

```
#include <stdio.h>

float A(float x, float y)
{
    if (x<=0)
        return 0.5;
    else if (y>=0)
        return -0.1;
    else
        return A(A(x-0.2,0),A(0,y+0.1))+x+y;
}

main()
{
    printf("%f",A(1,-1));
}
```










DAFTAR PUSTAKA

- Anharku. 2009. Flowchart. www.ilmukomputer.org. Tanggal akses : 11 Novemver 2015
- Banahan, M, Brady, D, Doran, M. 1991. The C Book. GBDirect
- Burgess, Mark. 1999. C Programming Tutorial. Oslo College
- Deitel, Paul dan Deitel, Harvey. 2010. C : How to Program. Pearson Education : New Jearsey
- Kernighan, B.W., Ritchie, D.M. 1988. The C Programming Language. Prentice Hall : New Jearseys.
- Parlante, Nick. 2003. Essential C. Stanford CS Education : California
- Tutorials Point. 2014. Learn C Programming. www.tutorialspoint.com. Tanggal akses : 10 Oktober 2015
- www.asciitable.com. ASCII Table and Description. Tanggal akses : 11 Novemver 2015

LAMPIRAN

Lampiran 1 : Simbol Flowchart









Sumber : Anharku, 2009

SIMBOL	NAMA	FUNGSI
	TERMINATOR	Permulaan/akhir program
	GARIS ALIR (FLOW LINE)	Arah aliran program
	PREPARATION	Proses inisialisasi/ pemberian harga awal
	PROSES	Proses perhitungan/ proses pengolahan data
	INPUT/OUTPUT DATA	Proses input/output data, parameter, informasi
	PREDEFINED PROCESS (SUB PROGRAM)	Permulaan sub program/ proses menjalankan sub program
	DECISION	Perbandingan pernyataan, penyeleksian data yang memberikan pilihan untuk langkah selanjutnya
	ON PAGE CONNECTOR	Penghubung bagian-bagian flowchart yang berada pada satu halaman
	OFF PAGE CONNECTOR	Penghubung bagian-bagian flowchart yang berada pada halaman berbeda

Lampiran 2 : Tabel ASCII

Sumber : www.asciitable.com

Dec	Hx	Oct	Char	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr
0	0	000	NUL (null)	32	20	040	 Space	64	40	100	@ @	96	60	140	` `			
1	1	001	SOH (start of heading)	33	21	041	! !	65	41	101	A A	97	61	141	a a			
2	2	002	STX (start of text)	34	22	042	" "	66	42	102	B B	98	62	142	b b			
3	3	003	ETX (end of text)	35	23	043	# #	67	43	103	C C	99	63	143	c c			
4	4	004	EOT (end of transmission)	36	24	044	$ \$	68	44	104	D D	100	64	144	d d			
5	5	005	ENQ (enquiry)	37	25	045	% %	69	45	105	E E	101	65	145	e e			
6	6	006	ACK (acknowledge)	38	26	046	& &	70	46	106	F F	102	66	146	f f			
7	7	007	BEL (bell)	39	27	047	' '	71	47	107	G G	103	67	147	g g			
8	8	010	BS (backspace)	40	28	050	((72	48	110	H H	104	68	150	h h			
9	9	011	TAB (horizontal tab)	41	29	051))	73	49	111	I I	105	69	151	i i			
10	A	012	LF (NL line feed, new line)	42	2A	052	* *	74	4A	112	J J	106	6A	152	j j			
11	B	013	VT (vertical tab)	43	2B	053	+ +	75	4B	113	K K	107	6B	153	k k			
12	C	014	FF (NP form feed, new page)	44	2C	054	, ,	76	4C	114	L L	108	6C	154	l l			
13	D	015	CR (carriage return)	45	2D	055	- -	77	4D	115	M M	109	6D	155	m m			
14	E	016	SO (shift out)	46	2E	056	. .	78	4E	116	N N	110	6E	156	n n			
15	F	017	SI (shift in)	47	2F	057	/ /	79	4F	117	O O	111	6F	157	o o			
16	10	020	DLE (data link escape)	48	30	060	0 0	80	50	120	P P	112	70	160	p p			
17	11	021	DC1 (device control 1)	49	31	061	1 1	81	51	121	Q Q	113	71	161	q q			
18	12	022	DC2 (device control 2)	50	32	062	2 2	82	52	122	R R	114	72	162	r r			
19	13	023	DC3 (device control 3)	51	33	063	3 3	83	53	123	S S	115	73	163	s s			
20	14	024	DC4 (device control 4)	52	34	064	4 4	84	54	124	T T	116	74	164	t t			
21	15	025	NAK (negative acknowledge)	53	35	065	5 5	85	55	125	U U	117	75	165	u u			
22	16	026	SYN (synchronous idle)	54	36	066	6 6	86	56	126	V V	118	76	166	v v			
23	17	027	ETB (end of trans. block)	55	37	067	7 7	87	57	127	W W	119	77	167	w w			
24	18	030	CAN (cancel)	56	38	070	8 8	88	58	130	X X	120	78	170	x x			
25	19	031	EM (end of medium)	57	39	071	9 9	89	59	131	Y Y	121	79	171	y y			
26	1A	032	SUB (substitute)	58	3A	072	: :	90	5A	132	Z Z	122	7A	172	z z			
27	1B	033	ESC (escape)	59	3B	073	; ;	91	5B	133	[[123	7B	173	{ {			
28	1C	034	FS (file separator)	60	3C	074	< <	92	5C	134	\ \	124	7C	174	|			
29	1D	035	GS (group separator)	61	3D	075	= =	93	5D	135]]	125	7D	175	} }			
30	1E	036	RS (record separator)	62	3E	076	> >	94	5E	136	^ ^	126	7E	176	~ ~			
31	1F	037	US (unit separator)	63	3F	077	? ?	95	5F	137	_ _	127	7F	177	 DEL			

128	Č	144	É	160	á	176		192	Ł	208	⌌	224	α	240	≡
129	û	145	æ	161	í	177		193	⌊	209	≠	225	β	241	±
130	é	146	Æ	162	ó	178		194	⌋	210	≡	226	Γ	242	≥
131	â	147	ô	163	ú	179		195	⌈	211	⌌	227	π	243	≤
132	ä	148	ö	164	ñ	180	⌋	196	—	212	≡	228	Σ	244	∫
133	à	149	ò	165	Ñ	181	≠	197	+	213	≡	229	σ	245	∫
134	â	150	û	166	▪	182	≡	198	⌋	214	≡	230	μ	246	÷
135	ç	151	ù	167	°	183	≡	199	⌋	215	≡	231	τ	247	≈
136	ê	152	ÿ	168	¿	184	≡	200	⌌	216	≡	232	Φ	248	°
137	ë	153	Ö	169	┐	185	≡	201	≡	217	┐	233	⊕	249	·
138	è	154	Ü	170	┐	186	≡	202	≡	218	┐	234	Ω	250	·
139	ï	155	◊	171	½	187	≡	203	≡	219		235	δ	251	√
140	î	156	£	172	¼	188	≡	204	⌋	220		236	∞	252	∞
141	ï	157	¥	173	┐	189	≡	205	≡	221		237	φ	253	z
142	Ä	158	£	174	«	190	≡	206	≡	222		238	ε	254	■
143	Å	159	f	175	»	191	┐	207	≡	223		239	∩	255	...

INDEX

- argumen, 30, 85
- aritmatik pointer, 124
- array *lihat larik*,
- ASCII, 27
- bit, 26, 37
- break, 52, 67
- Cakupan variabel, 87
- call by reference, 86
- call by value, 85
- case sensitive, 13, 20
- char, 26, 27, 89, 114
- compiler, 9
- continue, 69, 70
- define, 16, 17
- double, 26
- do-while, 62, 66
- file header, 16
- float, 16, 26
- floating point, 26
- for, 62, 63
- fungsi, 14, 15, 81, 82, 104, 105, 131
- gets, 115
- goto, 71
- identifier, 20
- if, 46, 47
- if-else, 47
- if-else if, 48
- include, 14, 16
- infinite loop*, 73
- integer, 15, 26
- karakter khusus, 26, 41
- kata kunci, 20, 21
- kode, 9
- komentar, 14, 17
- kompilasi, 9
- konstanta, 25, 28
- label, 71
- larik, 98, 133
- library, 116, 118
- long, 26
- looping. *lihat perulangan*
- matriks, 104
- nilai kembalian, 82
- null, 95, 114
- operand*, 32
- operator, 31, 32, 33, 35
- operator binary, 31
- operator unary, 32
- ordinal, 51
- parameter, 82, 83
- parameter formal, 87
- pencacah, 63
- perulangan, 62, 63, 66, 72
- perulangan pak perhingga. *lihat infinite loop*
- pointer, 89, 104, 120
- post-increment*, 34
- pre-increment*, 34
- preprosesor, 9, 14, 16
- rekursi, 89
- reserved word. *lihat kata kunci*
- short, 26
- signatur fungsi, 82
- stdin, 118
- string kontrol, 15
- struktur, 129, 130, 131, 132, 133
- switch-case, 51
- tipe data, 25
- unsigned, 27
- variabel, 27
- variabel global, 87
- variabel lokal, 87
- while, 62, 65

GLOSSARY

- ASCII : Kepanjangannya *American Standard Code for Information Interchange*. Merupakan kode yang merepresentasikan karakter latin dalam bentuk angka, 0 sampai 127 untuk huruf, angka, tanda baca dan 128 sampa 255 untuk karakter khusus. Misalnya kode ASCII huruf A (kapital) adalah 65 dan a kecil adalah 97. Nilai ASCII berguna dalam pertukaran data antar komputer.
- Bit : Kepanjangan dari *binary digit*, merupakan ukuran unit terkecil dari informasi pada suatu mesin. Sebuah bit hanya dapat bernilai 1 atau 0. Sebuah data dalam komputer biasanya merupakan kombinasi dari nilai beberapa bit. Misalnya 1000001 merupakan representasi bit dari angka desimal 65, atau dalam ASCII yaitu huruf A. Istilah ini pertama kali digunakan pada 1946 oleh John Tukey.
- Boolean : Sistem logika yang dikembangkan oleh George Boole. Boolean sering digunakan pada ilmu komputer. Sebuah data bertipe boolean dapat bernilai 0 (false) atau 1 (true). Boolean terdiri dari operator AND, OR, NOT dan XOR.
- Byte : Merupakan satuan penyimpanan untuk sebuah karakter. Satu byte sama dengan 8 bit. Ukuran memori saat ini dinyatakan dalam bentuk byte, misalnya 1 KB= 1 kilo byte, atau 1 MB = 1 mega byte. $1\text{ KB} = 2^{10}\text{ byte}$ atau 1.024 byte.
- Compiler : Sebuah program yang mampu melakukan penerjemahan dari kode program menjadi kode objek, sehingga sebuah progra bisa dijalan oleh pengguna.
- floating point : Representasi formula yang memperkirakan sebuah bilangan riil. Sebuah floating point akan memiliki ketelitian untuk merepresentasikan bilangan riil

tergantung pada besarnya memori dari tipe data yang digunakan. Misalnya double akan memiliki ketelitian yang lebih besar dibandingkan dengan float.

- garbage value : Nilai lama yang tertinggal di suatu memori. Setiap memori yang pernah digunakan oleh suatu variabel akan menyimpan nilai dari variabel tersebut. Nilai tersebut akhirnya tidak digunakan lagi, namun belum dibersihkan dan tetap berada di memori.
- file header : Penanda atau nama file yang diletakkan di atas sebuah file (file kode program), sehingga sistem operasi atau perangkat lunak akan tahu apa yang harus dilakukan pada konten file tersebut.
- karakter khusus : Karakter yang bukan berupa angka, huruf atau tanda baca. Karakter khusus misalnya string kontrol, yaitu suatu karakter untuk melakukan pemformatan pada penampilan data.
- kompilasi : Proses penerjemahan kode program menjadi kode objek, yang dilakukan oleh compiler
- konstanta : Sebuah nilai yang tidak pernah berubah sejak diberi nilai pertama kali. Dalam pemrograman, konstanta dideklarasikan dan langsung diberi nilai, pemberian nilai setelah deklarasi akan menimbulkan kesalahan sintaks.
- label : Penanda pada suatu lokasi tertentu di dalam sebuah program, biadanya menunjuk baris tertentu pada program.
- library : Merupakan sekumpulan modul yang tersimpan dalam format objek dan dapat dipergunakan oleh suatu program lainnya. Library biasanya digunakan untuk menyimpan modul-modul yang sering digunakan dan linker akan otomatis menghubungkan program dengan library ketika terjadi fase linking. Library memiliki ekstensi .dll
- linker : Disebut juga *link editor* atau *binder*, merupakan sebuah program yang mengkombinasikan modul objek untuk

	menghasilkan program yang dapat dieksekusi (.exe)
matriks	: Merupakan angka, simbol atau ekspresi yang disusun dalam baris dan kolom (segiempat). Matrik memiliki ordo, misalnya 2x3 berarti, matriks tersusun atas 2 baris dan 3 kolom
modulo	: Sisa hasil pembagian suatu bilangan dengan bilangan lainnya. Misalnya 5 modulo 3 adalah 2.
nilai kembalian	: Variabel atau informasi lainnya yang dikembalikan oleh suatu fungsi, setelah fungsi tersebut dijalankan.
null	: Tidak memiliki nilai atau dalam konteks pointer, pointer yang tidak menunjuk memori manapun. Suatu variabel yang bernilai null berbeda dengan nol. Variabel bernilai nol berarti memiliki sebuah nilai yaitu bilangan 0, sementara null berarti tidak bernilai.
operand	: Data yang dioperasikan atau dimanipulasi dalam suatu operasi. Misalnya dalam operasi $b = 2 + 3$, operand-nya adalah b, 2 dan 3, sedangkan tanda = dan + adalah operator
reserved word	: Istilah atau frase yang digunakan secara khusus oleh program untuk melakukan fungsi atau aksi tertentu.
signatur fungsi	: Informasi umum tentang suatu fungsi di dalam bahasa pemrograman, seperti nama fungsi, parameter dan cakupannya.
statement	: Instruksi dalam program yang meminta komputer untuk melakukan suatu aksi tertentu
unsigned	: Istilah yang digunakan untuk suatu bilangan, unsigned berarti tipe data yang tidak mendukung tanda dalam bilangannya. Tipe data unsigned hanya dapat memuat bilangan positif saja.

CURRICULUM VITAE PENGARANG



Putu Indah Ciptayani merupakan salah satu dosen di Program Studi Manajemen Informatika, Jurusan Teknik Elektro Politeknik Negeri Bali. Penulis mendapatkan gelar sarjananya pada Program Studi Ilmu Komputer Universitas Brawijaya dan masternya pada Program Studi Ilmu Komputer Universitas Gadjah Mada. Penulis telah menjadi tenaga pengajar sejak tahun 2009. Adapun matakuliah yang biasa diajarkan adalah Pemrograman Dasar, Struktur Data, Basis Data, Logika Informatika dan Pemrograman Berorientasi Objek. Sejak tahun 2014, penulis menjadi dosen tetap di Politeknik Negeri Bali dan diberikan kepercayaan untuk mengajarkan matakuliah Pemrograman Dasar dan Pemrograman Beorientasi Objek 2.