MODUL PERKULIAHAN

ALGORITMA DAN PEMROGRAMAN

PENULIS:
DIAN PRAWIRA

JURUSAN SISTEM INFORMASI
UNIVERSITAS TANJUNGPURA
PONTIANAK
2015

Daftar Isi

BAGIAN 1
PENGANTAR ALGORITMA DAN PEMROGRAMAN

Capaian Pembelajaran:
1. Mahasiswa mampu menjelaskan secara singkat pengertian dari algoritma

1.1 Pendahuluan

Program yang berjalan pada komputer baik itu pada perangkat PC, Laptop, Smartphone dan lainnya tidak dibuat begitu saja, akan tetapi tentunya melalui suatu proses analisis dan perancangan. Suatu program pada dasarnya merupakan implementasi dari suatu algoritma yang sudah dirancang sebelumnya. Jadi, algoritma merupakan ide dasar dibalik adanya program yang berjalan pada perangkat komputer.

1.2 Pengertian Algoritma

Tujuan pembuatan suatu program tentunya adalah untuk membantu mengurai dan mengatasi suatu masalah. Masalah dapat di atasi tentunya dengan urutan langkah-langkah tertentu. Untuk masalah dengan instansiasi yang relatif kecil kita dapat menemukan solusinya dengan mudah dan cepat. Namun bila masalah yang relatif lebih besar tentunya akan sulit untuk menemukan solusiya dengan cepat tanpa bantuan suatu metode yang menguraikan solusi-solusi dari suatu masalah menjadi langkah-langkah tertentu. Langkah-langkah penyelesaian masalah tersebut yang kemudian disebut sebagai algoritma.

Pengertian dasar Algoritma dapat diartikan sebagai urutan langkah-langkah untuk memecahkan suatu masalah. Langkah-langkah tersebut harus dapat dikerjakan dan mempunyai dampak tertentu. Terdapat beberapa definisi lain algoritma yang bersumber dari beberapa literatur diantaranya:

Algoritma adalah deretan langkah-langkah komputasi yang mentransformasikan data masukan menjadi keluaran.[Cormen, 1990]

Algoritma adalah deretan istruksi yang jelas untuk memecahkan masalah yaitu untuk memperoleh keluaran yang diinginkan.

Algoritma adalah prosedur komputasi yang terdefinisi dengan baik, yang menggunakan beberapa nilai sebagai masukan dan menghasilkan beberapa nilai yang disebut keluaran. Jadi, algoritma adalah deretan langkah komputasi yang mentransformasikan masukan menjadi keluaran.

Dalam kehidupan sehari-hari banyak kita temukan beberapa implementasi dari penggunaan algoritma. Contohnya adalah pada resep masakan. Pada resep masakan umumnya ada bagian "cara pembuatan". Sebagai contoh resep membuat roti cane berikut:

RESEP ROTI CANE

1. Campur tepung, garam dengan air hangat.
2. Setelah rata masukkan mentega cair.
3. Remas-remas sampai tidak lengket, bagi menjadi 20 bagian, masing-masing dibulatkan, diamkan ± selama 15 menit, tipiskan setebal ± 1/2 cm.
4. Panaskan dan olesi dengan sedikit minyak wajan datar untuk membuat martabak, panggang adonan yang sudah ditipiskan di atas, balik-balik sampai matang (ada bagianbagian yang jadi lebih cokelat). Angkat.
5. Lakukan sampai adonan habis.

Walaupun resep tersebut berjudul resep pembuatan roti cane namun dalam konteks pembahasan algoritma maka langkah-langkah tersebut dapat disebut sebagai suatu algoritma.

Contoh lainnya adalah pada sebuah kasus yang dikenal dengan istilah water jug problem. Sebagai contoh anda diminta untuk mendapatkan air dari sebuah danau sebanyak 4 liter dengan menggunakan bantuan dua buah ember. Kedua ember tersebut mempunyai volume 5-liter dan 3-liter. Tidak ada peralatan lain yang tersedia, hanya kedua ember itu saja. Cara yang digunakan bebas untuk mengambil air tersebut. Maka akan ada banyak sekali kemungkinan cara yang akan digunakan oleh orang. Salah satunya tertuang pada algoritma berikut:

ALGORITMA mendapatkan air dengan volume 4 liter

1. Isi penuh ember 3 liter dengan air

2. Tuangkan air dari ember 3 liter ke dalam ember 5 liter.

3. Isi penuh ember 3 liter dengan air.

4. Tuangkan air dari ember 3 liter ke dalam ember 5 liter hingga penuh

5. Buang seluruh air ember 5 liter ke danau

6. Tuangkan air dari ember 3 liter ke dalam ember 5 liter

7. Isi penuh ember 3 liter dengan air

8. Tuangkan air dari ember 3 liter ke dalam ember 5 liter

Serta banyak lagi contoh lainnya dari algoritma seperti penggunaannya pada:

- Panduan registrasi mahasiswa
- Panduan pembukaan rekening listrik
- Panduan penggunaan software
- Panduan pemasangan atap
- dan lain sebagainya

1.3 Sejarah algoritma

Asal kata algoritma tersebut yang berasal dari nama penulis buku arab yang terkenal yaitu Abu Ja'far Muhammad Ibnu Musa Al-Khuwarizmi. Al-Khuwarizmi dibaca orang barat menjadi Algorism. Al-Khuwarizmi menulis buku yang berjudul "Kitab Al Jabar Wal-Muqabala" yang artinya "Buku pemugaran dan pengurangan" (The book of restoration and reduction). Dari judul buku itulah diperoleh akar kata "Aljabar" (Algebra).

Perubahan kata dari algorism menjadi algorithm muncul karena kata algorism sering dikelirukan dengan arithmetic, sehingga akhiran usm berubah menjadi uthm. Karena perhitungan dengan angka Arab sudah menjadi hal yang biasa, maka lambat laun kata algorithm berangsur-angsur dipakai sebagai metode perhitungan (komputasi) secara umum, sehingga kehilangan makna kata aslinya.

Pada 1950, algoritma pertama kali digunakan pada Algoritma Eucliden (Euclid Algorithm). Euclid sendiri merupakan seorang matemaitkawan Yunani yang lahir sekitar 350 SM. Euclid menulis buku yang berjudul Element.

Di dalam buku tersebut, dijelaskan langkah-langkah untuk menemukan pembagi bersama terbesar (common greatest divisor) dari dua bilangan bulat, yakni m dan n. Namun, Eucliden pada saat itu tidak menyebutkan bahwa cara yang digunakannya adalah metode algoritma. Hal tersebut baru disebut sebagai algoritma pada abad-abad modern.

1.4 Kriteria Algoritma

Menurut Donald E. Knuth dalam bukunya The Art Of Computer Programming mendefinisikan 5 kriteria pemrograman yang menjadi ciri penting dari suatu algoritma

1. Input: algoritma dapat memiliki nol atau lebih inputan dari luar.

2. Output: algoritma harus memiliki minimal satu buah output keluaran.

3. Definite (pasti): algoritma memiliki instruksi-instruksi yang jelas dan tidak ambigu.

4. Finite (ada batas): algoritma harus memiliki titik berhenti (stopping role).

5. Effective (tepat dan efisien): algoritma sebisa mungkin harus dapat dilaksanakan dan efektif. Contoh instruksi yang tidak efektif adalah: A = A + 0 atau A = A * 1

Namun ada beberapa program yang memang dirancang untuk unterminatable, contoh Sistem Operasi

BAGIAN 2
DASAR BAHASA PEMROGRAMAN

Capaian Pembelajaran
1. Mahasiswa mampu menjelaskan jenis-jenis bahasa pemrograman
2. Mahasiswa mampu membedakan antara kode sumber program dan file program

2.1 Program dan Pemrograman

Suatu algoritma dikatakan efektif bila dijalankan oleh sebuah pemroses (processor). Pemroses tersebut bisa saja berupa manusia, robot, mesin, komputer dan lain sebagainya. Instruksi yang terdapat pada algoritma dibaca dan kemudian dikerjakan oleh processor. Suatu pemroses harus:

1. Mengerti setiap langkah dalam algoritma

2. Mengerjakan operasi yang bersesuaian dengan langkah tersebut

Karena cakupan pembahasan kita persempit pada algoritma dan pemrograman maka yang bertugas untuk memproses algoritma tersebut (processor) adalah komputer. Bila kita bicara dengan sesama manusia tentunya bahasa yang kita gunakan adalah bahasa yang dimengerti oleh manusia tersebut. Tentu ini berkaitan lagi dengan daerah masing-masing manusia tersebut berada dan tingkat pemahaman manusia terhadap suatu bahasa. Bagitu pula dengan komputer. Bila ingin berinteraksi dengan komputer maka harus menggunakan bahasa yang dimengerti oleh komputer, yaitu bahasa komputer. Hal tersebut agar apa yang diperintahkan pada komputer dapat segera diproses. Algoritma yang ditulis dalam bahasa komputer disebut dengan program. Beberapa istilah lainnya yang berkaitan dengan program ini yaitu:

- Bahasa Pemrograman: Bahasa yang digunakan dalam penulisan program adalah bahasa pemrograman.
- *Programmer* : orang yang membuat program komputer.
- Pemrograman: Kegiatan/pekerjaan merancang dan menulis program

Secara garis besar komputer tersusun dari empat komponen utama:

a) Perangkat Masukan

Merupakan perangkat yang digunakan untuk berkomunikasi dengan komputer.

Perangkat masukan juga berfungsi untuk memasukkan data atau program ke dalam memori. Contohnya adalah keyboard, mouse, joystick, dan lain-lain.

b) Perangkat Keluaran

Merupakan perangkat yang digunakan untuk menampilkan atau menghasilkan keluaran hasil pemrosesan dari Unit Pemrosesan Utama. Contohnya adalah monitor, printer, dan lain-lain.

c) Unit Pemrosesan Utama

Dikenal juga dengan CPU (Central Processing Unit) yang merupakan otak dari komputer. Fungsi dari CPU adalah mengerjakan operasi-operasi dasar seperti operasi perbandingan, perhitungan, membaca dan menulis

d) Memori

Merupakan komponen yang berfungsi sebagai tempat penyimpanan sementara selama program dijalankan. Termasuk juga menyimpan program(yang diproses oleh CPU) dan data (yang diolah oleh proses-proses pada program)

Adapun mekanisme kerja keempat elemen tersebut:

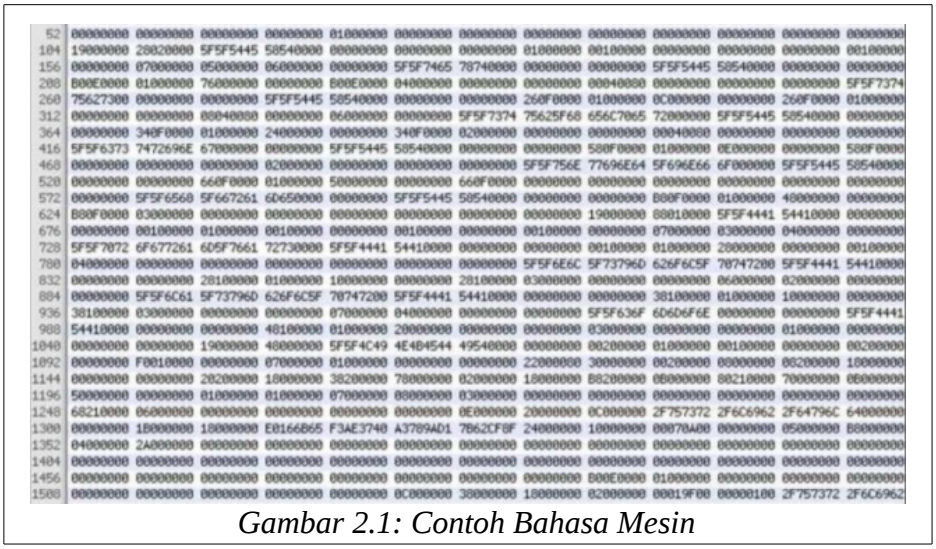
1. Program dimasukkan ke dalam memori
2. Ketika program di-eksekusi setiap perintah di dalam program yang telah tersimpan di dalam memori dikirim ke CPU.
3. CPU mengerjakan operasi-operasi yang bersesuaian dengan perintah tersebut
4. Bila suatu perintah dari program meminta data masukan, maka data dibaca dari perangkat masukan, kemudian dikirim ke CPU untuk operasi yang memerlukannya
5. Bila program menghasilkan suatu keluaran, maka keluaran tersebut ditulis ke perangkat keluaran

2.2 Perkembangan Bahasa Pemrograman

Hingga saat ini terdapat puluhan bahasa pemrograman yang berkembang. Jika dikelompokkan berdasarkan level bahasa maka bahasa pemrograman dapat dikelompokkan menjadi 4 yaitu: bahasa mesin, bahasa tingkat rendah, bahasa tingkat tinggi, dan bahasa yang berorientasi pada permasalahan spesifik.

1) Bahasa Mesin

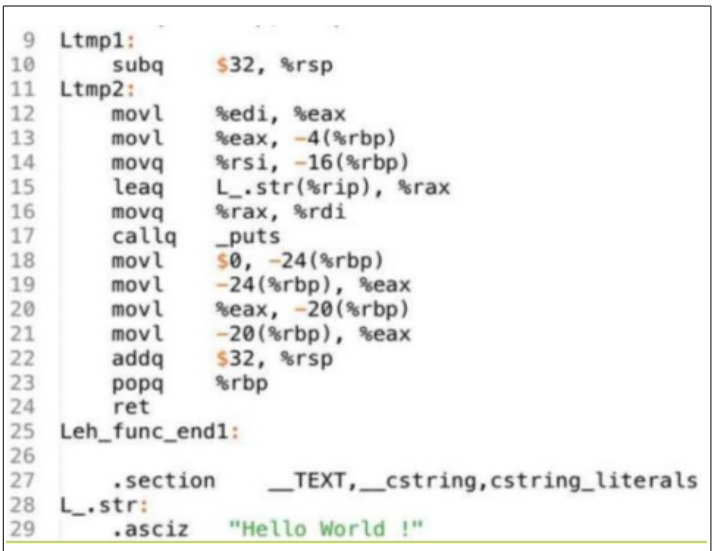
Bahasa mesin merupakan bahasa dengan level terendah . Bahasa mesin berisi kode-kode mesin yang hanya dapat diinterpretasikan secara langsung oleh mesin komputer , sebab berupa kode numerik, biner, dan hexadesimal seperti terlihat pada Gambar 2.1. Kelebihan dari bahasa mesin ini adalah eksekusinya yang cepat , namun memiliki kelemahan sulit dipelajari manusia.



Gambar 2.1: Contoh Bahasa Mesin

2) Bahasa Assembly

Bahasa assembly merupakan bahasa simbol dari bahasa mesin . Contoh instruksi seperti ADD, MUL, SUB, DIV . Kelebihan bahasa assembly adalah eksekusi yang cepat, dan masih dapat dipelajari oleh manusia daripada bahasa mesin seperti terlihat pada Gambar 2.2. Selain itu bahasa assembly memiliki file yang kecil . Namun dibalik kelebihan tersebut bahaasa assembly tetap sulit dipelajari, dikarenakan program yang sangat panjang . Umumnya bahasa assembly digunakan untuk pembuatan driver, firmware, kernel (inti dari suatu sistem operasi).



```
Ltmp1:
    subq    $32, %rsp
Ltmp2:
    movl    %edi, %eax
    movl    %eax, -4(%rbp)
    movq    %rsi, -16(%rbp)
    leaq    L_.str(%rip), %rax
    movq    %rax, %rdi
    callq   _puts
    movl    $0, -24(%rbp)
    movl    -24(%rbp), %eax
    movl    %eax, -20(%rbp)
    movl    -20(%rbp), %eax
    addq    $32, %rsp
    popq    %rbp
    ret
Leh_func_end1:

    .section    __TEXT,__cstring,cstring_literals
L_.str:
    .asciz   "Hello World !"
```

Gambar 2.2: Contoh Bahasa Assembly

3) Bahasa Tingkat Tinggi

Bahasa tingkat tinggi dikenal juga dengan "The 3rd Generation Programming Language ". Bahasa tingkat tinggi ini lebih dekat dengan bahasa manusia karena menggunakan perintah-perintah yang umumnya menggunakan bahasa inggris sehingga bisa lebih dimengerti oleh programmer. Bahasa tingkat tinggi ini memberi banyak fasilitas kemudahan dalam pembuatan program, misalnya, variabel, tipe data, konstanta, struktur kontrol, loop, fungsi, prosedur, dan lain sebagainya. Contoh dari bahasa tingkat tinggi ini yaitu, Pascal, Basic, C, Java, PHP , dan lain-lain. Kelebihan dari bahasa tingkat tinggi ini yaitu Mudah dipelajari; mendekati permasalahan yang akan dipecahkan; kode program pendek . Namun bahasa tingkat tinggi juga memiliki kelemahan yaitu eksekusinya yang lambat.

4) Specific Problem Oriented

Specific Problem Oriented atau lebih dikenal juga dengan "The 4 th Generation Programming Language". Merupakan bahasa pemrograman yang digunakan langsung untuk memecahkan suatu masalah tertentu. Contohnya adalah SQL untuk database, GUI Programming (Visual Basic.NET, Delphi, Qt).

2.3 Langkah Pembuatan Program

Dalam pembuatan program ada beberapa langkah yang ditempuh, diantaranya:

a) Mendefinisikan masalah

Menurut hukum Murphy (oleh Henry Ledgard), "Semakin cepat menulis program, akan semakin lama kita dapat menyelesaikannya". Langkah pendefinisian masalah ini memang terkadang sering dilupakan oleh seorang programmer. Pendefinisian masalah ini berlaku untuk permasalahan yang kompleks. Secara garis besar, langkah pendefinisian masalah yaitu penentian masalah; apa saja yang semestinya dipecahkan dengan menggunakan komputer; penentuan input; penentuan output.

b) Menemukan solusi

Setelah masalah didefinisikan, maka langkah berikutnya adalah menentukan solusi. Jika masalah terlalu kompleks, maka ada baiknya masalah tersebut dipecah menjadi modul-modul kecil agar lebih mudah diselesaikan. Dengan penggunaan modul tersebut program utama akan menjadi lebih singkat dan mudah dilihat.

c) Memilih algoritma

Pilihlah algoritma yang benar-benar sesuai dan efisien untuk permasalahan tersebut.

d) Menulis program

Pilihlah bahasa yang mudah dipelajari, mudah digunakan, dan lebih baik lagi jika sudah dikuasai, memiliki tingkat kompatibilitas tinggi dengan perangkat keras dan platform lainnya

e) Menguji Program

Setelah program jadi, silahkan uji program tersebut dengan segala macam kemungkinan yang ada, termasuk error-handlingnya sehingga program tersebut akan benar-benar handal dan layak digunakan.

f) Menulis dokumentasi

Menulis dokumentasi sangat penting agar pada suatu saat jika kita akan melakukan perubahan atau membaca source code. Caranya adalah dengan menuliskan komentar-komentar kecil pada kode sumber program.

g) Mendistribusikan aplikasi

Dalam mendistribusikan aplikasi dapat ditempuh dengan berbagai cara, diantaranya adalah: melakukan kompresi file, membuat installer, maupun paket yang memudahkan pengguna untuk memasang program pada perangkat yang dimiliki.

h) Merawat program

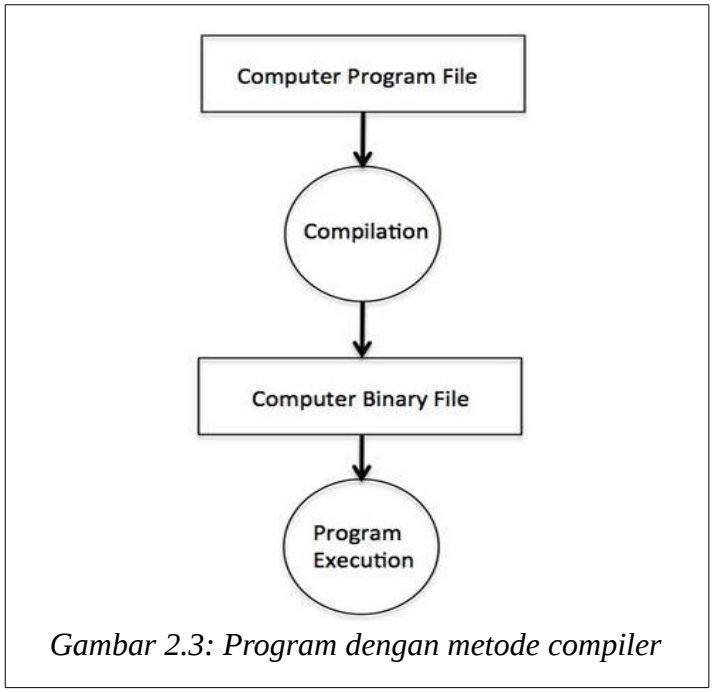
Program yang sudah jadi perlu dirawat untuk mencegah munculnya bug yang sebelumnya tidak terdeteksi. Perawatan program juga diperlukan untuk mendefinisikan bahwa pengguna membutuhkan fasilitas baru yang dulu tidak ada dengan cara penambahan fitur-fitur tertentu.

2.4 Metode Translasi

Pada bahasa pemrograman tingkat tinggi, bahasa tersebut tidak dapat langsung dimengerti oleh komputer, oleh karena itu dibutuhkan suatu cara atau metode untuk merubahnya ke dalam bahasa mesin. Terdapat dua metode untuk melakukan hal tersebut, diantaranya:

a) Compiler

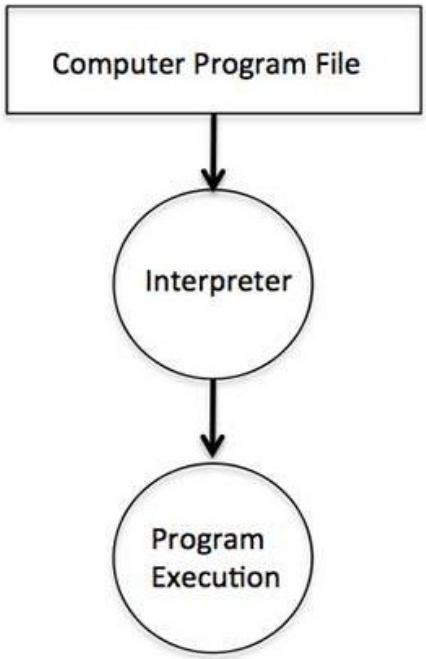
Dengan metode compiler kode sumber program akan dikompilasi terlebih dahulu ke dalam program dalam format binary sehingga akan menghasilkan file baru yang bila di windows berekstensi .exe. Perhatikan Gambar 2.3 untuk skema metode compiler ini. Contoh program yang menggunakan metode compiler ini adalah bahasa C, C++, Pascal, dan lain-lain. Kelebihan dari program yang menggunakan metode ini adalah jalannya program lebih cepat, kode sumber yang tidak terpisah dengan program, lebih fleksibel ketika akan dijalankan pada perangkat-perangkat lain yang mempunyai platform yang sama.



Gambar 2.3: Program dengan metode compiler

b) Interpreter

Beda halnya dengan compiler, maka program dengan metode interpreter ini tidak memerlukan proses kompilasi terlebih dahulu. Kode sumber program langsung dapat dijalankan dengan proses interpretasi dari bahasa yang menggunakan interpretasi itu. Perhatikan untuk membayangkan metode interpreter ini. Contoh dari program dengan metode interpreter ini adalah bahasa Python, Ruby, Javascript, dan lain-lain. Kelebihan program dengan metode translasi interpreter ini yaitu mudah bagi user, dan debugging yang cepat. Hanya saja metode ini juga memiliki kelemahan yaitu eksekusi program lambat dikarenakan tidak langsung menjadi program executable, selain itu kode sumber yang sangat transparan.



Gambar 2.4: Skema proses program dengan metode interpreter

Sebenarnya masih terdapat 1 jenis metode lagi yaitu yang menggabungkan antara interpreter dan compiler, dan yang menerapkan metode tersebut adalah bahasa Java. Namun dari beberapa literatur menjadikan Java sebagai bahasa program dengan metode compiler.

2.5 Pemrograman Prosedural

Algoritma berisi urutan langkah-langkah penyelesaian masalah. Ini berarti langkah-langkah di dalam algoritma menyatakan proses yang prosedural. Pada pemrograman prosedural, program dibedakan antara bagian data dan bagian instruksi. Bagian instruksi terdiri atas runtunan (sequence) instruksi yang dilaksanakan satu per satu secara berurutan oleh sebuah pemroses. Alur pelaksanaan instruksi dapat berubah karena adanya pencabangan kondisional. Data yang disimpan didalam memori dimanipulasi oleh instruksi secara beruntun atau procedural. Paradigma pemrograman seperti ini dinamakan pemrograman procedural.

2.6 Bahasa Pemrograman C

C adalah Bahasa pemrograman tingkat tinggi yang terkait erat dengan sistem operasi UNIX yang dikembangkan. Hal tersebut dikarenakan sistem UNIX dan sebagian besar program yang dijalankan ditulis dalam C. Dalam perkembangannya bahasa C juga digunakan dalam berbagai sistem operasi selain UNIX. Bahasa program C dirancang oleh Dennis M Ritchie di

Bell Laboratories pada tahun 1972. Dan pada tahun 1978Dennis dan Brian W. Kernighan mempublikasikan bahasa C melalui buku "The C Programming Language" . Pada perkembangan selanjutnya Bahasa C distandarisasi ANSI (The American National Standard Institute) dan Standar ISO/IEC 9899:1990.

Bahasa C dikatakan sebagai bahasa pemrograman terstruktur, prosedural karena strukturnya menggunakan fungsi-fungsi sebagai bagian program-program (subroutine / module). Fungsi-fungsi selain fungsi utama disebut subroutine/ module dan ditulis setelah fungsi utama (main) atau diletakkan pada file pustaka (library). Kode sumber bahasa C berekstensi .c dan bila dilakukan compile pada sistem operasi windows akan menghasilkan program dengan ekstensi .exe sedangkan pada sistem operasi linux tidak perlu untuk membuat ekstensi.

Untuk melakukan compile bahasa mempunyai banyak compiler diantaranya adalah:

- GCC (GNU C Compiler)
- MinGW
- OpenWatcom compiler
- Tiny C Compiler
- Borland C
- Intel C++
- Visual C/C++
- dan lain-lain

Struktur Program Bahasa C minimal harus memiliki function main(), tanpa function itu maka program C tidak dapat dieksekusi tapi bisa dikompilasi. Perhatikan format berikut:

```
<preprocessor directive>
int main() {
      <statement>
      <statement>
return 0
}
```

Statement adalah suatu baris instruksi/perintah tertentu. Statement menyebabkan suatu tindakan akan dilakukan oleh komputer. Diakhiri dengan titik koma (;).

Preprosessor Directive adalah bagian yang berisi pengikutsertaan file atau berkas-berkas fungsi, pendefinisian konstanta, atau fungsi makro tertentu. Berikut ini adalah contoh program sederhana untuk penjumlahan.

```
#include <stdio.h>
int main()
{
   int a,b,c;
   printf("Isi bilangan pertama:");
   scanf("%d",&a);
   printf("Isi bilangan kedua:");
   scanf("%d",&b);
   c = a + b;
   printf("Hasil %d + %d = %d\n",a,b,c);
   return 0
}
```

BAGIAN 3
PSEUDOCODE DAN FLOWCHART

Capaian Pembelajaran
1. Mahasiswa mampu merancang alur algoritma dengan Pseudocode dan Flowchart

3.1 Pseudocode

Pseudocode adalah notasi yang mirip dengan notasi bahasa pemrograman tingkat tinggi seperti pascal, C, ataupun python. Pseudocode dapat pula didefinisikan sebagai campuran antara bahasa alami manusia dengan bahasa pemrograman. Dalam menulis pseudocode tidak ada aturan yang benar-benar baku, oleh karena itu sembarang versi pseudocode dapat diterima asalkan perintahnya tidak membingungkan pembaca. Contoh pseudocode seperti terlihat di bawah ini:

```
PROGRAM Bilangan Terbesar
Program ini berfungsi mencari bilangan terbesar dari dua bilangan yang
diinputkan

ALGORITMA
INIT a,b : integer  //bilangan bulat yang akan dijadikan variabel input
READ a
READ b
IF a > b THEN PRINT a
ELSE PRINT b
```

Walaupun tidak terdapat standar baku mengenai penulisan pseudocode ini, namun untuk mempermudah dalam penulisan pseudocode selama perkuliahan maka digunakan aturan berikut:

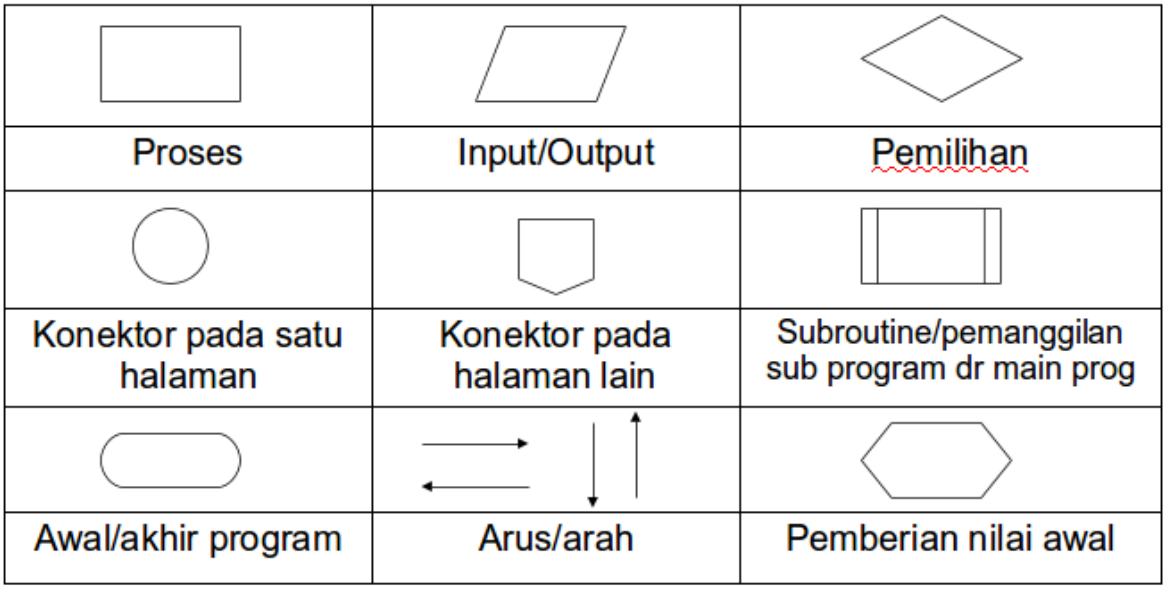
- Menerima input: READ, GET
- Menampilkan output: DISPLAY, PRINT, WRITE, SHOW
- Aritmatika: +,-,*,/,%,div,sub,mul,min
- Pemberian nilai: ←
- Inisialisasi: INIT, SET
- Memilih: IF ... THEN ... ELSE ... , CASE ...

- pengulangan: FOR, WHILE

Pada contoh pseudocode sebelumnya terdapat tanda // yang merupakan simbol dari komentar yang hanya digunakan untuk memberi catatan pada program maupun pseudocode.

3.2 Flowchart

Pada awal perkembangan teknologi komputer, para peneliti menspesifikasikan algoritma sebagai bagan alir *(flowchart)*, yang mengekspresikan algoritma sebagai sekumpulan bentuk-bentuk geometri yang berisi langkah-langkah komputasi. Gambar 5 memperlihatkan simbol-simbol umum dari flowchart.



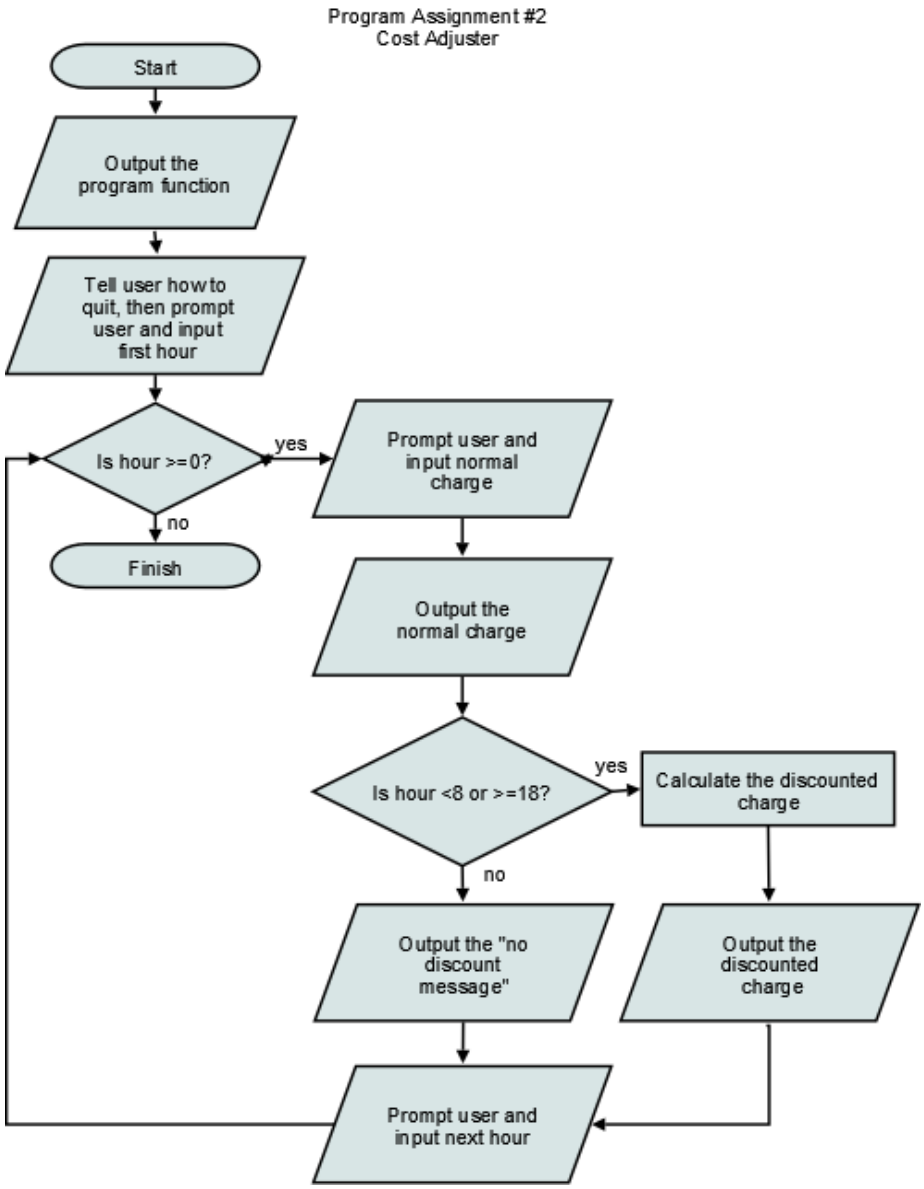
Gambar 3.1: Simbol-simbol pada flowchart beserta artinya

Pada pembuatan flowchart ada beberapa langkah yang dapat ditempuh yaitu:

1. Penggambaran dari atas ke bawah
2. Kegiatan-kegiatan harus ditunjukkan dengan jelas
3. Harus menunjukkan mulai kegiatan dan akhir dari kegiatan
4. Masing-masing kegiatan harus di dalam urutan yang semestinya
5. Kegiatan yang terpotong ditunjukkan dengan simbol penghubung
6. Menggunakan simbol-simbol standar yang sudah umum digunakan

Bagan alir (flowchart) tidak banyak digunakan lagi pada masa sekarang karena flowchart dianggap tidak praktis bila dikonversi ke dalam bahasa pemrograman jika algoritma yang dibuat sudah cukup kompleks. Namun untuk algoritma yang sederhana flowchart masih berguna.

Pada terlihat pada Gambar 3.2 suatu contoh flowchart.



Gambar 3.2: Contoh gambar dari flowchart
(Sumber: https://commons.lbl.gov)

BAGIAN 4
TIPE DATA, VARIABEL, DAN I/O

Capaian Pembelajaran
1. *Mahasiswa mampu membedakan tipe data yang terdapat pada bahasa pemrograman C*
2. *Mahasiswa mampu memahami cara menggunakan fungsi input dan output standar dalam bahasa C*
3. *Mahasiswa mampu membuat program dengan menggunakan Variabel dengan tipe data yang tepat*

4.1 Tipe Data

Tipe data merupakan bagian program yang paling penting karena tipe data mempengaruhi setiap instruksi yang akan dilaksanakan oleh computer. Misalnya saja 5 dibagi 2 bisa saja menghasilkan hasil yang berbeda tergantung tipe datanya. Jika 5 dan 2 bertipe integer maka akan menghasilkan nilai 2, namun jika keduanya bertipe float maka akan menghasilkan nilai 2.5000000. Pemilihan tipe data yang tepat akan membuat proses operasi data menjadi lebih efisien dan efektif.

Dalam bahasa C terdapat lima tipe data dasar, seperti terlihat pada Tabel 4.1

Tabel 4.1: Tipe data dasar pada bahasa C

No	Tipe Data	Ukuran	Range (Jangkauan)	Format	Keterangan
1	Char	1 byte	-128 s.d. 127	%c	Karakter/String
2	Int	4 byte	-2,147,483,648 s.d. 2,147,483,647	%i, %d	Integer/Bilangan Bulat
3	Float	4 byte	1.2E-38 s.d. 3.4E+38	%f	Float/Bilangan Pecahan (6 desimal)
4	Double	8 byte	2.3E-308 s.d. 1.7E+308	%lf	Pecahan Presisi Ganda
5	Void	0 byte	-	-	Tidak Bertipe

4.2 Konstanta

Konstanta Konstanta merupakan suatu nilai yang tidak dapat diubah selama proses program berlangsung. Konstanta nilainya selalu tetap. Konstanta harus didefinisikan terlebih dahulu di awal program. Konstanta dapat bernilai integer, pecahan, karakter dan string. Contoh

konstanta : 50; 13; 3.14; 4.50005; 'A'; 'Bahasa C'.

Selain itu, bahasa C juga menyediakan beberapa karakter khusus yang disebut karakter escape yang dapat dilihat pada Tabel 4.2

Tabel 4.2: Karakter Khusus/ Karakter Escape

No	Karakter Escape	Keterangan
1	\a	untuk bunyi bell (alert)
2	\b	mundur satu spasi (backspace)
3	\n	ganti baris baru (new line)
4	\r	ke kolom pertama, baris yang sama (carriage return)
5	\t	tabulasi vertikal
6	\'	karakter petik tunggal
7	\"	karakter petik ganda
8	\\	karakter garis miring

Perhatikan contoh kode program berikut

```
#include <stdio.h>

int main () {
   printf ("\a") ; // bunyi bip
   printf ("\n \n");// turun kebawah 2 kali
   printf ("\tPenggunaan Tab"); // penggunaan tab
   printf ("\n \"SAYA BELAJAR C \" " ); // penggunaan tanda petik
   printf ("\nTulisan Tertimpa Dengan Tulisan Berikutnya");
   printf ("\r Mulai dari awal"); // tulisan ini mulai dari awal
   return 0;
}
```

4.3 Variabel

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

Nama dari suatu variable dapat ditentukan sendirioleh pemrogram dengan aturan sebagai berikut :

- Terdiri dari gabungan huruf dan angka dengan karakter pertama harus berupa huruf.

Bahasa C bersifat case-sensitive artinya huruf besar dan kecil dianggap berbeda. Jadi antara nim, NIM dan Nim dianggap berbeda.

- Tidak boleh mengandung spasi. Tidak boleh mengandung symbol-simbol khusus, kecuali garis bawah (underscore).
- Yang termasuk symbol khusus yang tidak diperbolehkan antara lain : $, ?, %, #, !, &, *, (,), -, +, = dsb.
- Panjangnya bebas, tetapi hanya 32 karakter pertama yang terpakai.

Perhatikan Contoh Kode Program berikut[1]:

```
# include <stdio.h>

int main () {
   int x , a ;
   float y ;
   char z ;
   double w ;
   a =17;
   x = 10; /* variable x diisi dengan 10 */
   y = 9.45; /* variable y diisi dengan 9.45 */
   z = 'C '; /* variable z diisi dengan karakter " C " */
   w = 3.45786; /* variable w diisi dengan 3.45786 */
   printf("Nilai dari x adalah : % i \n",x); /* Menampilkan isi
   ¬ variable x */
   printf("Nilai dari y adalah : % f \n",y); /* Menampilkan isi
   ¬ variable y */
   printf("Nilai dari z adalah : % c \n", z); /* Menampilkan isi
   ¬ variable z */
   printf("Nilai dari w adalah : % lf \ n",w); /* Menampilkan
¬ isi variable w */
   printf("%i %i Nilai dari x adalah % c : \n",x,a,z);
   return 0;
}
```

1: dalam contoh program ini dan beberapa contoh program lain pada modul ini terdapat simbol ¬ yang mempunyai arti bahwa baris tersebut merupakan sambungan dari baris sebelumnya. Simbol tersebut sebenarnya tidaklah terdapat pada program yang sebenarnya, hanya sebagai simbol pembantu dikarekan keterbatasan area penulisan pada kertas.

4.4 Deklarasi

Deklarasi diperlukan bila kita akan menggunakan pengenal (identifier) dalam program. Identifier dapat berupa variable, konstanta dan fungsi.

4.4.1 Deklarasi Preprocessor

Pada bahasa C, program diawali dengan deklarasi preprocessor. Preprocessor yang umum dideklarasikan adalah library C yang digunakan dalam program. Seperti pada contoh penggalan program di bawah ini:

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

4.4.2 Deklarasi Fungsi

Pada bahasa C harus mendeklarasikan minimal satu fungsi utama yaitu adalah fungsi main(). Standarisasi dari penulisan program main yang baik adalah dengan memberi nilai kembalian int. Perhatikan contoh penggalan program berikut

```
int main() {
   return 0;
}
```

4.4.3 Deklarasi Variabel

Bentuk umum pendeklarasian suatu variable adalah :

```
Nama_tipe nama_variabel;
```

Contoh :

```
int x; // Deklarasi x bertipe integer
char y, huruf, nim[10]; // Deklarasi variable bertipe char
float nilai; // Deklarasi variable bertipe float
double beta; // Deklarasi variable bertipe double
int array[5][4]; // Deklarasi array bertipe integer
```

4.4.4 Deklarasi Konstanta

Dalam bahasa C konstanta dideklarasikan menggunakan preprocessor #define. Contohnya :

```
#define PHI 3.14
```

```
#define Grafitasi 9.8
#define Sepuluh 10
```

Contoh Kode Program:

```
# include <stdio.h>

int main(){
   # define PHI 3.14
   # define Grafitasi 9.8
   # define Sepuluh 10
   printf("Phi : % f",PHI); // memanggil konstanta
   printf("\n Gravitasi : %f",Grafitasi); // memanggil
¬ konstanta
   printf("\n %i = Sepuluh : ",Sepuluh); // memanggil konstanta
   return 0;
}
```

4.5 Komentar

Komentar program hanya diperlukan untuk memudahkan pembacaan dan pemahaman suatu program (untuk keperluan dokumentasi program). Dengan kata lain, komentar program hanya merupakan keterangan atau penjelasan program. Untuk memberikan komentar atau penjelasan dalam bahasa C digunakan pembatas /* dan */ atau menggunakan tanda // untuk komentar yang hanya terdiri dari satu baris. Komentar program tidak akan ikut diproses dalam program (akan diabaikan)

Contoh Kode Program

```
# include <stdio.h>

int main () {
   printf("Contoh Penggunaan Komentar");// komentar tidak ikut diproses
   return 0;
}
```

4.6 Input dan Output (I/O)

Dalam pemrograman dua fungsi yang penting untuk digunakan adalah funsgi input dan output. Kedua fungsi tersebut umumnya memberdayakan perangkat input dan perangkat output pada komputer.

4.6.1 Input

Dalam bahasa C proses memasukkan suatu data bisa menggunakan beberapa fungsi pustaka yang telah tersedia. Beberapa fungsi pustaka yang bisa digunakan adalah : scanf() & getchar().

4.6.1.1 Scanf()

Fungsi pustaka scanf() digunakan untuk menginput data berupa data numerik, karakter dan string secara terformat. Hal-hal yang perlu diperhatikan dalam pemakaian fungsi scanf() :

1. Fungsi scanf() memakai penentu format
2. Fungsi scanf() memberi pergantian baris secara otomatis
3. Fungsi scanf() tidak memerlukan penentu lebar field
4. Variabelnya harus menggunakan operator alamat &

Kode penentu format :

- **`%c`** : Membaca sebuah karakter
- **`%i, %d`** : Membaca sebuah bilangan bulat (integer)
- **`%f, %1d`** : Membaca sebuah bilangan pecahan (real)
- **`%o`** : membaca sebuah bilangan octal
- `%x` : Membaca sebuah bilangan heksadesimal

Perhatikan contoh program di bawah ini:

```
# include <stdio.h>
int main () {
   int ujian1 , ujian2 ;
   printf("Masukan Nilai ujian 1 =");
   scanf("%d" ,&ujian1);
   printf("Masukan Nilai ujian 2 =");
   scanf("%d",&ujian2);
   printf("Nilai ujian 1 = % i dan ujian 2 = % i",ujian1,ujian2);
   return 0;
}
```

Perhatikan pula contoh program berikut:

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

```
int main () {
   float rataRataUjian;
   printf("Masukan Rata - rata Ujian = ");
   scanf("%f" ,&rataRataUjian);
   printf("Rata - rata Ujian =%.3f",rataRataUjian);
   return 0;
}
```

4.6.1.2 Getchar()

Fungsi getchar() digunakan untuk membaca data yang bertipe karakter . Ada beberapa kondisi berkaitan dengan getchar() diantaranya:

- Harus diakhiri dengan penekanan tombol enter
- Karakter yang dimasukkan terlihat pada layar
- Pergantian baris secara otomatis

Perhatikan contoh program berikut

```
# include <stdio.h>
int main () {
   char karakter ;
   printf("Masukkan Karakter:");
   karakter = getchar();
   printf("Karakter yang dimasukkan = % c \ n",karakter);
   return 0;
}
```

4.6.2 Ouput

Bentuk umum dari output adalah :

```
printf("%m.nf", argument);
```

Yang memiliki arti:

- m : menyatakan panjang range
- n : menyatakan jumlah digit di belakang koma.
- argument : nilai atau variable yang akan ditampilkan.

Contoh :

```
printf("%5.2f", nilai);
```

Artinya variable nilai akan ditampilkan sebanyak 5 digit dengan 2 digit di belakang koma.

Perhatikan contoh program berikut:

```
# include <stdio . h>
int main () {
   printf("Contoh 1:%6d \n" ,9876) ;
   printf("Contoh 2:%4d \n" ,987689) ;
   printf( Contoh 3:%10.2f \n" ,11987.6543) ;
   printf("Contoh 4:%.2f \n" ,987.6543) ;
   printf("Contoh 5:%e \n" ,987.6543) ;
   printf("Contoh 1:%8 d \n" ,129876) ;
   printf("Contoh 2:%8d  \n" ,9876) ;
   return 0;
}
```

BAGIAN 5
OPERATOR DAN EKSPRESI

Capaian Pembelajaran
1. *Mahasiswa mampu menjelaskan macam-macam Operator dalam bahasa C*
2. *Mahaiswa mampu menjelaskan macam-macam ekspresi dalam bahasa C*
3. *Mahasiswa mampu membuat program sederhana dengan menggunakan operator dan ekspresi dalam bahasa C*

5.1 Operator

Operator merupakan simbol-simbol yang digunakan untuk tujuan tertentu. Operator dalam bahasa C dikelompokan berdasarkan keperluan atau kegunaannya.

5.1.1 Operator Penugasan

Operator Penugasan (Assignment operator) dalam bahasa C berupa tanda sama dengan("="). Contoh : nilai = 80; A = x * y; Artinya : variable "nilai" diisi dengan 80 dan variable "A" diisi dengan hasil perkalian antara x dan y.

5.1.2 Operator Aritmatika

Bahasa C menyediakan lima operator aritmatika, yaitu :

* : untuk perkalian

/ : untuk pembagian

% : untuk sisa pembagian (modulo) [2]

+ : untuk pertambahan

- : untuk pengurangan

Perhatikan contoh program berikut:

```
# include <stdio .h>
int main (){
int a,b;
a=80;
b=15;
printf (" Nilai dari 10 + 4 = %i\n", 10 + 4);
```

2 operator % digunakan untuk mencari sisa pembagian antara dua bilangan. Misalnya : 9 % 2 = 1 atau 9 % 3 = 0.

```
printf (" Nilai dari 10 - 4 = %i\n", 10 - 4);
printf (" Nilai dari 10 * 4 = %i\n", 10 * 4);
printf (" Nilai dari 10 / 4 = %i\n", 10 / 4);
printf (" Nilai dari 10 %% 4 = %i\n", 10 % 4);
printf (" Nilai dari %i + %i = %i\n",a,b, a + b);
 printf (" Nilai dari %i - %i = %i\n",a,b, a – b);
return 0;
}
```

5.1.3 Operator Hubungan

Operator Hubungan digunakan untuk membandingkan hubungan antara dua buah operand (sebuah nilai atau variabel). Operator hubungan dalam bahasa C dapat dilihat pada Tabel 5.1.

Tabel 5.1: Tabel Macam-Macam Operator Hubungan Dalam Bahasa C

Operator	Arti	Contoh	
<	Kurang dari	x<y	Apakah x kurang dari y
<=	Kurang dari atau sama dengan	x<=y	Apakah x kurang dari atau sama dengan y
>	Lebih dari	x>y	Apakah x lebih dari y
>=	Lebih dari atau sama dengan	x>=y	Apakah x lebih dari atau sama dengan y
==	Sama dengan	x==y	Apakah x sama dengan y
!=	Tidak sama dengan	x!=y	Apakah x tidak sama dengan y

Perhatikan contoh program berikut

```
# include " stdio .h"
int main (){
   int a,b;
   a =80;
   b =15;
   printf (" Perbandingan 10 <4 = %i\n", 10 <4);
   printf (" Nilai dari 7>4 = %i\n", 7 >4);
   printf (" Nilai dari 8 <=8 = %i\n", 8 <=8);
   printf (" Nilai dari 9 >=7 = %i\n", 9 >=7);
   printf (" Nilai dari 6==4 = %i\n", 6==4) ;
   printf (" Nilai dari 5!=4 = %i\n", 5!=4) ;
   printf (" Nilai dari %i >=% i = %i\n",a,b, a >=b);
   a=b;
```

```
   printf (" Nilai dari a!=b = %i\n", a!=b);
   return 0
}
```

5.1.4 Operator Logika

Jika operator hubungan membandingkan hubungan antara dua buah operand, maka operator logika digunakan untuk membandingkan logika hasil dari operator-operator hubungan.

Operator logika ada tiga macam, yaitu :

&& : Logika AND (DAN)

|| : Logika OR (ATAU)

! : Logika NOT (INGKARAN)

Perhatikan contoh program berikut

```
# include <stdio .h>
int main (){
   printf (" Logika Dan \n");
   printf (" (10 <4) &&(7 >8) = %i\n", (10 <4) &&(7 >8) );
   printf (" (10 >4) &&(7 <8) = %i\n", (10 >4) &&(7 <8) );
   printf ("\ nLogika Or\n");
   printf ("(9 >4) ||(7 >8) = %i\n", (9 >4) ||(7 >8) );
   printf ("(9 <4) ||(7 >8) = %i\n", (9 <4) ||(7 >8) );
   printf ("\ nLogika Not \n");
   printf (" !(6 <4) = %i\n", !(6 <4) );
   printf (" !(6 >4) = %i\n", !(6 >4) );
return 0;
}
```

5.1.5 Operator Bitwise

Operator bitwise digunakan untuk memanipulasi bit-bit dari nilai data yang ada di memori.

Operator bitwise dalam bahasa C :

<< : Pergeseran bit ke kiri

>> : Pergeseran bit ke kanan

& : Bitwise AND

^ : Bitwise XOR (exclusive OR)

| : Bitwise OR

~ : Bitwise NOT

Perhatikan contoh program berikut

```
#include <stdio.h>

int main()
{

   int a = 60;
   int b = 13;
   int c = 0;

   c = a & b;
   printf("Rumus 1 - Nilai dari c adalah %d\n", c );

   c = a | b;
   printf("Rumus 2 - Nilai dari c adalah %d\n", c );

   c = a ^ b;
   printf("Rumus 3 - Nilai dari c adalah %d\n", c );

   c = ~a;
   printf("Rumus 4 - Nilai dari c adalah %d\n", c );

   c = a << 2;
   printf("Rumus 5 - Nilai dari c adalah %d\n", c );

   c = a >> 2;
   printf("Rumus 6 - Nilai dari c adalah %d\n", c );
return 0;
}
```

5.1.6 Operator Unary

Operator Unary merupakan operator yang hanya membutuhkan satu operand saja. Operator Unary dalam bahasa C antara lain:

- - : Mengurangi 1 dari nilai variabel. Contoh penggunaan a- - atau - -a

++ : Menambahkan 1 dari nilai variabel. Contoh penggunaan a++ atau ++a

Perhatikan contoh program berikut

```
# include <stdio .h>
int main (){
   int a;
   a =8;
   a ++;
   printf ("a = %i\n", a);
   a =12;
   ++a;
   printf ("a = %i\n", a);
   a =10;
   a --;
   printf ("a = %i\n", a);
   a =7;
   --a;
   printf ("a = %i\n", a);
return 0;
}
```

```
# include <stdio .h>
int main (){
   int a;
   a =8;
   a +=3;
   printf ("a = %i\n", a);
   a =12;
   a +=5;
   printf ("a = %i\n", a);
   a =10;
   a -=5;
   printf ("a = %i\n", a);
   a =7;
   a -=4;
   printf ("a = %i\n", a);
return 0;
}
```

```
# include <stdio .h>
int main (){
   int a;
   a =9;
```

```
   printf ("a = %i\n", a ++) ;
   printf ("a = %i\n", a);
   a =17;
   printf ("a = %i\n", ++a);
   printf ("a = %i\n", a);
   a =10;
   printf ("a = %i\n", a --);
   printf ("a = %i\n", a);
   a =15;
   printf ("a = %i\n", --a);
   printf ("a = %i\n", a);
return 0;
}
```

5.2 Ekspresi

Suatu nilai dipakai untuk proses transformasi menjadi keluaran yang diinginkan. Transformasi nilai menjadi keluaran dilakukan melalui suatu perhitungan atau komputas. Cara perhitungan itu dinyatakan dalam suatu ekspresi. Ekspresi terdiri atas operand dan operator. Operand adalah nilai yang dioperasikan dengan nilai tertentu. Operang dapat berupa konstanta, nama peubah, nama konstanta, atau hasil dari suatu fungsi. Hasil evaluasi dari sebuah ekspresi adalah nilai di dalam ranah yang sesuai dengan tipe operand yang dipakai. Dikenal tida macam ekspresi yaitu ekspresi aritmetik, ekspresi relasional, dan ekspresi string.

5.2.1 Ekspresi Aritmatik

Ekspresi numerik merupakan ekspresi yang operand dan hasilnya bertipe numerik. Perhatikan contoh program berikut

```
#include <stdio.h>

int main() {
    int a,b,c;
    float d,i,j,k;
    a = 1;
    b = 2;
    c = 3;
    d=a*c;
    printf("%f",d);

return 0;
}
```

5.2.2 Ekspresi Relasional

Ekspresi relasional merupakan ekspresi dengan operator-operator <,<=,>,>=, dan !=; NOT, AND, OR, XOR. Perhatikan contoh program berikut:

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

int main() {
    bool x,y;
    x = true;
    y = false;
    printf("%d", y&&x);

return 0;
}
```

5.2.3 Ekspresi String

Ekspresi string adalah ekspresi yang menggunakan operator yang mendukung pengolahan string. In syaa Allah pembahasan mengenai ekspresi string akan dibahas pada pembahasan mengenai string.

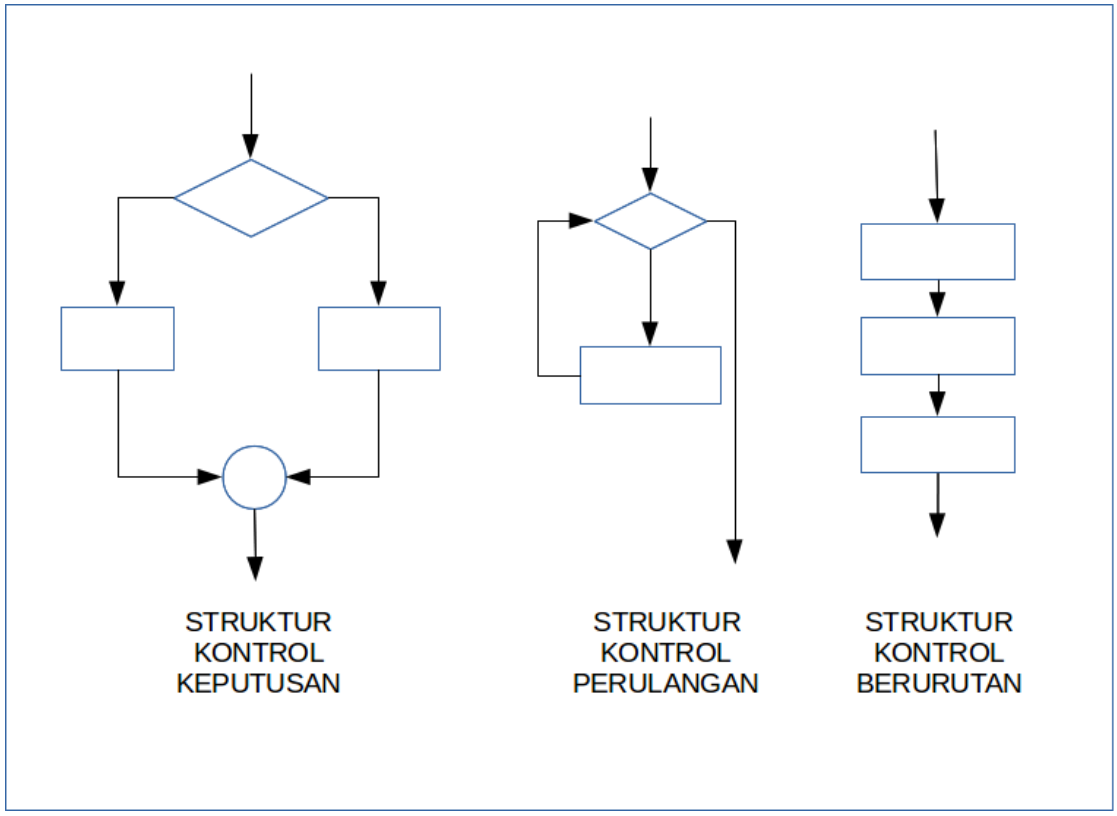
BAGIAN 6
STRUKTUR KONTROL DASAR

Capaian Pembelajaran
1. Mahasiswa mampu menjelaskan konsep Struktur kontrol dasar
2. Mahasiswa mampu membuat program sederhana dengan menggunakan struktur kontrol dasar pada bahasa C

Struktur kontrol di dalam pemrograman adalah kumpulan perintah dengan bentuk tertentu yang digunakan untuk melakukan pegontrolan terhadap jalannya program. Pada dasarnya struktur kontrol terdapat beberapa macam diantaranya adalah:

1. Struktur kontrol berurutan (*Sequence Control Structures*)
2. Struktur kontrol keputusan/percabangan/kondisi (*Selection Control Structures*)
3. Struktur kontrol pengulangan (*Iteration Control Structures*)

Bila struktur kontrol digambarkan dalam bentuk flowchart maka dapat dilihat pada Gambar 6.1



Gambar 6.1: Macam-macam struktur kontrol

Struktur kontrol berurutan merupakan struktur kontrol yang paling sederhana dari sebuah program. Pada struktur kontrol berurutan program berjalan berdasarkan perintah-perintah berurutan yang diberikan oleh programmer dari awal kode sumber program hingga akhir kode sumber program. Pada umumnya contoh-contoh program yang telah dibahas pada bagian-bagian sebelumnya, dan memiliki pola sebagai berikut:

```
int main() {
   statement 1;
   statement 2:
   statement 3;
}
```

Perhatikan contoh program berikut

```
#include <stdio.h>

int main() {
```

```
    int a,b,c;
    float d,i,j,k;
    a = 1;
    b = 2;
    c = 3;
    d=a*c;
    printf("%f",d);

return 0;
}
```

Pada contoh program tersebut, program akan menjalankan perintah mulai dari penugasan/pemberian nilai pada variabel a kemudian memberi nilai pada variabel b, dilanjutkan pemberian nilai pada variabel c, dan diakhiri dengan pemberian nilai d yang merupakan hasil dari perkalian a dan c untuk kemudian ditampilkan di layar monitor.

Dalam suatu program dikenal pula istilah blok kode program yang merupakan kumpulan perintah yang diait oleh tanda kurung kurawal {} . Kumpulan perintah yang diapit oleh tanda tersebut akan dijalankan sesuai dengan fungsi dari blok tersebut. Pada contoh program sebelumnya blok program yang mengandung kurung kurawal merupakan penjelas bahwa kumpulan perintah tersebut merupakan fungsi main yang merupakan fungsi utama dari sebuah program yang ditulis dengan bahasa c.

BAGIAN 7
STRUKTUR KONTROL PERCABANGAN

Capaian Pembelajaran
1. Mahasiswa mampu menjelaskan konsep Struktur kontrol percabangan
2. Mahasiswa mampu membuat program sederhana dengan menggunakan struktur kontrol percabangan pada bahasa C

Banyak istilah dari struktur kontrol percabangan. Pada literatur-literatur lain didapatkan istilah percabangan dengan nama seleksi, ada pula yang menyebutnya sebagai struktur kontrol keputusan. Namun dari semua istilah mengandung makna yang sama, sebab pada beberapa hal berubahnya nama tidaklah dapat mengubah makna dari sesuatu. Begitu pula dengan percabangan dari beberapa istilah yang telah disebutkan sebelumya merujuk pada makna sebuah struktur program yang digunakan untuk mengarahkan perjalanan suatu proses.

Percabangan dapat diibaratkan sebagai katup atau kran yang mengatur jalannya air. Bila katup terbuka maka air akan mengalir dan sebaliknya bila katup tertutup air tidak akan mengalir atau akan mengalir melalui tempat lain. Fungsi penyeleksian kondisi penting artinya dalam penyusunan bahasa C, terutama untuk program yang kompleks.

7.1 Struktur Kondisi If...

Struktur if dibentuk dari pernyataan if dan sering digunakan untuk menyeleksi suatu kondisi tunggal. Bila proses yang diseleksi terpenuhi atau bernilai benar, maka pernyataan yang ada di dalam blok if akan diproses dan dikerjakan. Bentuk umum struktur kondisi if adalah :

```
if( kondisi ){
   pernyataan
}
```

Perhatikan contoh program berikut:

```
# include " stdio .h"
int main (){
   float nilai ;
   printf (" Masukan nilai yang didapat : ");
   scanf ("%f", & nilai );
   if( nilai > 65 && nilai <=100) {
      printf ("\n ANDA LULUS !!\ n");
   }
return 0;
}
```

7.2 Struktur Kondisi if... else...

Dalam struktur kondisi if.....else minimal terdapat dua pernyataan. Jika kondisi yang diperiksa bernilai benar atau terpenuhi maka pernyataan pertama yang dilaksanakan dan jika kondisi yang diperiksa bernilai salah maka pernyataan yang kedua yang dilaksanakan. Bentuk umumnya adalah sebagai berikut :

```
if( kondisi ){
   pernyataan 1
} else {
   pernyataan 2
}
```

Perhatikan contoh program berikut:

```
# include <stdio.h>
int main (){
   float nilai ;
   printf (" Masukan nilai yang didapat : ");
   scanf ("%f", & nilai );
   if( nilai > 65) {
      printf ("\n ANDA LULUS !!\ n");
   } else {
      printf ("\n ANDA TIDAK LULUS !!\ n");
   }
}
```

7.3 Struktur Kondisi if ... else if...else...

Dalam struktur kondisi if.....elseif...else minimal terdapat 3 pernyataan. Pada struktur ini setiap kondisi diperiksa, jika kondisi tersebut benar maka pernyataan yang berada di ruang lingkupnya yang dijalankan, namun bila semua kondisi tidak benar maka pernyataan pada bagian terakhir yang dijalankan.Bentuk umumnya adalah sebagai berikut :

```
if( kondisi 1){
   pernyataan 1
} else if( kondisi 2){
   pernyataan 2
} else if( kondisi 3){
   pernyataan 3
} else {
   pernyataan 4
}
```

Perhatikan contoh kode program berikut:

```
# include <stdio .h>
int main (){
   float nilai ;
   printf (" Masukan nilai yang didapat : ");
   scanf ("%f", & nilai );
   if( nilai > 80) {
      printf (" Anda Mendapat Nilai A");
   } else if( nilai > 70) {
      printf (" Anda Mendapat Nilai B");
   } else if( nilai > 60) {
      printf (" Anda Mendapat Nilai C");
   } else if( nilai > 50) {
      printf (" Anda Mendapat Nilai D");
   } else {
      printf (" Anda Mendapat Nilai E");
   }
   return 0;
}
```

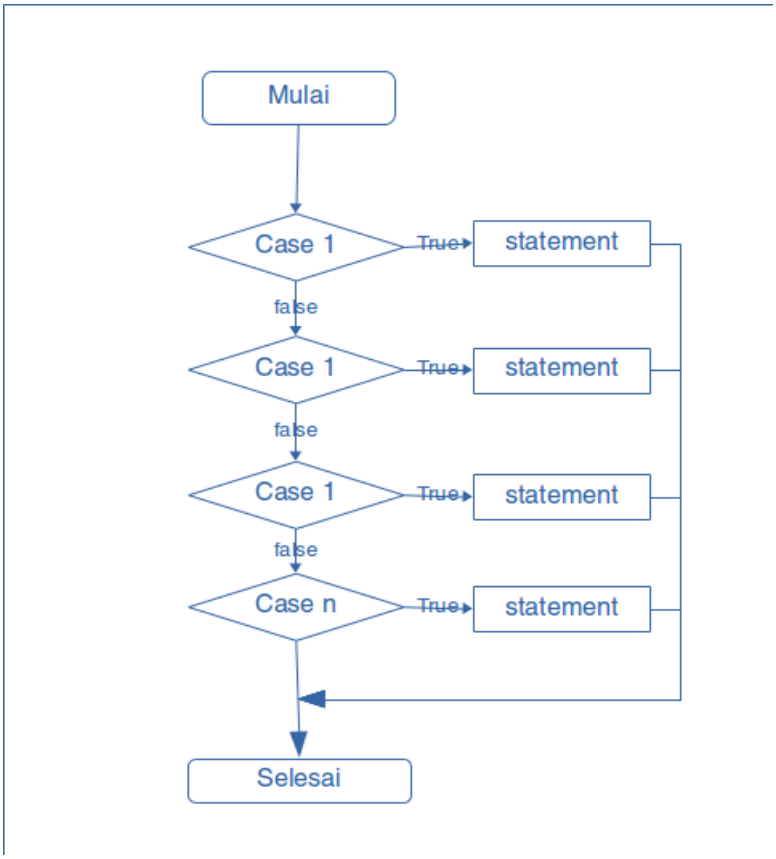
7.4 Struktur switch... case...

Suatu struktur kontrol switch case berguna untuk variabel yang akan diuji kesamaannya terhadap daftar nilai. Setiap dari nilai-nilai tersebut disebut sebagai case, dan pemilihan dari suatu aksi akan bergantung dari pensyaratan yang ada pada switch. Bentuk umumnya adalah sebagai berikut:

```
switch(ekspresi){
    case ekspresi_konstan  :
       statement;
       break;
case ekspresi_konstan  :
       statement;
       break;
   ...

    default : /* opsional */
       statement;
}
```

Bila digambarkan dalam bentuk flowchart dapat dilihat pada Gambar 7.1



Gambar 7.1: Flowchart statement switch case

BAGIAN 8
STRUKTUR KONTROL PENGULANGAN

Capaian Pembelajaran
1. Mahasiswa mampu menjelaskan konsep Struktur kontrol perulangan
2. Mahasiswa mampu membuat program sederhana dengan menggunakan struktur kontrol perulangan pada bahasa C

8.1 Pendahuluan

Pengulangan(looping) adalah suatu bagian yang bertugas melakukan kegiatan mengulang suatu proses sesuai dengan yang diinginkan. Banyak dari aplikasi perangkat lunak yang melakukan pekerjaan berulang sampai sebuah kondisi yang diinginkan, oleh karena itu pengulangan merupakan bagian yang penting dalam pemrograman karena dengan adanya pengulangan pembuat program tidak perlu menulis kode program sebanyak pengulangan yang diinginkan.

Pengulangan mempunyai beberapa bagian yang harus dipenuhi yaitu :

1. **Inisialisasi** adalah tahap persiapan membuat kondisi awal sel melakukan pengulangan, misalnya mengisi variabel dengan nilai awal. Tahap ini dilakukan sebelum memasuki bagian pengulangan.
2. **Proses** terjadi di dalam bagian pengulangan dimana berisi semua proses yang perlu dilakukan secara berulang-ulang.
3. **Iterasi** terjadi di dalam pengulangan di mana merupakan kondisi pertambahan agar pengulangan dapat terus berjalan.
4. **Terminasi** adalah kondisi berhenti dari pengulangan, kondisi berhenti sangat penting dalam pengulangan agar pengulangan dapat berhenti, tidak menjadi pengulangan yang tanpa henti. Kondisi pengulangan adalah kondisi yang dipenuhi oleh kondisi jalannya algoritma untuk masuk ke dalam blok pengulangan.

Pengulangan merupakan salah satu inti dari analisis kasus pada pembuatan algoritma, sebuah kasus harus dipikirkan penyelesaiannya dengan pemikiran ada proses atau aksi yang harus dikerjakan secara berulang agar sebuah kasus terselesaikan.

8.2 Pengulangan "for"

Struktur pengulangan for biasa digunakan untuk mengulang suatu proses yang telah diketahui jumlah pengulangannya. Dari segi penulisannya, struktur pengulangan for tampaknya lebih efisien karena susunannya lebih simpel dan sederhana. Bentuk umum pengulangan for adalah sebagai berikut :

```
for(Inisialisasi; Terminasi; Iterasi){
   Proses 1;
   Proses 2;
   Proses n;
}
```

Contoh Program Menampilkan Tulisan:

```
/* Program pengulangan menggunakan for */
# include <stdio .h>
int main (){
   int x;
   for(x =1; x <=10; x ++) {
      printf ("%d. Belajar Pengulangan for \n", x);
   }
   return 0;
}
```

Contoh Program Menampilkan Bilangan Ganjil:

```
/* Program Menampilkan Bilangan Ganjil 1 -10 */
# include <stdio .h>
int main (){
   int x;
   for(x =1;x <=10; x ++) {
      if(x %2==1) {
         printf ("%d ", x);
      }
   }
   return 0;
}
```

8.3 Pengulangan "while"

Pada pengulangan while, pengecekan terhadap pengulangan dilakukan di bagian awal (sebelum tubuh loop). Lebih jelasnya, bentuk struktur pengulangan while adalah sebagai berikut:

```
Inisialisasi
while(Terminasi){
   Proses 1;
   Proses 2;
   Proses n;
}
```

Contoh Program Menampilkan Tulisan:

```
# include <stdio.h>
int main (){
   int x;
   x =1;
   while (x <=10) {
      printf ("%d. Belajar Pengulangan While \n", x);
      x ++;
   }
}
```

Contoh program menampilkan bilangan genap

```
# include <stdio.h>
int main (){
   int x;
   x =1;
   while (x <=100) {
      if(x %2!=1) {
         printf ("%d ", x);
      }
      x ++;
   }
   return 0;
}
```

8.4 Pengulangan Repeat

Perbedaan mendasar dengan pengulangan yang lain adalah pada pengulangan repeat, proses pasti akan dilakukan minimal 1 kali. Hal ini terjadi karena terminasi diletakan setelah proses. Dalam bahasa C dan C++ pengulangan repeat menggunakan statement do-while. Berikut ini struktur dari pengulangan do-while:

```
Inisialisasi
do{
```

```
   Proses
   Iterasi
}while(Terminasi)
```

Setelah proses dijalankan 1 kali maka barulah terjadi pengecekkan apakah proses diulang lagi atau tidak.

Contoh Program Menampilkan Tulisan

```
# include <stdio.h>
int main (){
   int x;
   x =1;
   do{
      printf ("%d. Belajar Pengulangan do - While \n", x);
      x ++;
   } while (x <=10);
   return 0;
}
```

Contoh Program Menampilkan Bilangan Kelipatan 3:

```
# include <stdio.h>
int main (){
   int x;
   x =1;
   while (x <=10) {
      if(x %3==0) {
         printf ("%d ", x);
      }
      x ++;
   }
   return 0;
}
```

Contoh Program Mengulang Menu

```
# include <stdio.h>
# include <stdlib.h>
int main (){
   int x, pilihan ;
   do{
      system (" cls ");
      printf (" Pilihlah Menu Berikut Ini :\n");
```

```
      printf ("\t1. Cetak \n");
      printf ("\t2. Lihat \n");
      printf ("\t3. Keluar \n");
      printf (" Pilihan Anda : ");
      scanf ("%d" ,& pilihan );
      if( pilihan ==1) {
         printf ("\ nCetak Laporan . Siapkan Printer ");
      } else if( pilihan ==2) {
      printf ("\ nTampilkan Laporan ");
      }
   } while ( pilihan !=3) ;
}
```

BAGIAN 9
PERULANGAN BERSARANG

Capaian Pembelajaran
1. Mahasiswa mampu menjelaskan konsep Struktur kontrol perulangan bersarang
2. Mahasiswa mampu membuat program sederhana dengan menggunakan struktur kontrol perulangan bersarang pada bahasa C

9.1 Pendahuluan

Sebuah program mengizinkan blok pengulangan di dalam blok pengulangan lainnya, dan tidak membatasai jenis pengulangan apa yang boleh berada di dalam pengulangan lainnya, misalnya di dalam blok pengulangan for terdapat pengulangan while, atau didalam pengulangan while terdapat pengulangan for.

9.2 Pengulangan Bersarang "For"

Bentuk umum pengulangan bersarang for adalah sebagai berikut :

```
for(Inisialisasi; Terminasi; Iterasi){
   Proses;
   for(Inisialisasi; Terminasi; Iterasi){
      Proses;
   }
}
```

Contoh Program Menampilkan Baris Kolom Bilangan:

```
# include <stdio.h>
int main (){
   int i,j;
   printf ("\n Pengulangan Bersarang For \n");
   for(i =1;i <=5; i ++) {
      printf (" %i ",i);
      for (j =1;j <=3; j ++) {
         printf (" %i ",j);
      }
      printf ("\n");
   }
   return 0;
}
```

9.3 Pengulangan Bersarang "While

Bentuk umum pengulangan bersarang for adalah sebagai berikut :

```
Inisialisasi
while(Terminasi){
   Proses 1;
   Proses 2;
   Proses n;
   Inisialisasi
   while(Terminasi){
      Proses 1;
      Proses 2;
      Proses n;
   }
}
```

Contoh Program Menampilkan Baris Kolom Bilangan

```
# include <stdio .h>

int main (){
   int i,j;
   printf ("\ nPengulangan Bersarang While \n");
   i=0;
   while (i <=10) {
      printf ("%d ",i);
      j =0;
      while (j <=5) {
         printf ("%d ",j);
         j ++;
      }
      i ++;
      printf ("\n");
   }
   return 0;
}
```

BAGIAN 10
LARIK (ARRAY)

Capaian Pembelajaran
1. Mahasiswa mampu menjelaskan konsep dasar array
2. Mahasiswa mampu menjelaskan alokasi memory pada penggunaan array
3. Mahasiswa mampu membuat program sederhana dengan menggunakan array pada bahasa C

10.1 Larik 1 Dimensi

Array (larik) ialah penampung sejumlah data sejenis (homogen) yang menggunakan satu identifier (pengenal). Masing-masing elemen larik diakses menggunakan indeks (subscript) dari nol sampai n-1 (n menyatakan jumlah elemen larik). Pengolahan data larik harus per elemen. Elemen Larik dapat diakses langsung (acak), maksudnya untuk memanipulasi elemen ke-4 tidak harus melalui elemen ke-1, ke-2 dan ke-3. Berdasarkan banyaknya indeks larik dibagi menjadi satu dimensi dan multi dimensi (duadimensi, tiga dimensi).

10.1.1 Deklarasi

Pendeklarasian larik 1 dimensi mempunyai pola tipeVariabel namaVariabel[ukuran]. Contohnya:

```
int panjangPadi[5];
float nilaiUas[5];
```

10.1.2 Pengisian Larik

Ada 2 metode dalam pengisian Larik, yaitu:

• Pengisian langsung saat deklarasi

```
int my_array[] = {1,23,17,4,-5,100};
```

• Pengisian setelah deklarasi

```
int my_array[5];
my_array[0]=10;
my_array[1]=20;
my_array[2]=30;
my_array[3]=40;
my_array[4]=50;
```

10.1.3 Menampilkan Larik

Saat akan menampilkan Larik yang perlu diingat adalah indeks dari setiap Larik adalah

mulai dari 0, misalkan telah di deklarasikan int my_array[] = {1,23,17,4,-5,100}. Maka untuk menampilkan data Larik ke-2 yaitu 23 dari variabel my_array adalah my_array[1]. Berikut ini contoh kode lengkapnya dari program menampilkan larik satu dimensi tanpa perulangan.

```
include <stdio .h>

int main (){
   int my_array [6] = {1 ,23 ,17 ,4 , -5 ,100};
   printf (" Data ke -1 = %d\n",my_array [0]) ;
   printf (" Data ke -2 = %d\n",my_array [1]) ;
   printf (" Data ke -3 = %d\n",my_array [2]) ;
   printf (" Data ke -4 = %d\n",my_array [3]) ;
   printf (" Data ke -5 = %d\n",my_array [4]) ;
   printf (" Data ke -6 = %d\n",my_array [5]) ;
   return 0;
}
```

Berikut ini adalah kode program untuk menampilkan larik satu dimensi dengan perulangan

```
# include <stdio .h>

int main (){
   int my_array [6] = {1 ,23 ,17 ,4 , -5 ,100};
   int i;
   printf (" Data Larik Dari Depan \n");
   // Menggunakan Looping
   for(i =0;i <6; i ++) {
      printf (" Data ke -%d = %d\n",i+1, my_array [i]);
   }

   printf ("\ nData Larik Dari Belakang \n");
   // Menggunakan Looping Urut Terbalik
   for(i =5;i >=0;i --){
      printf (" Data ke -%d = %d\n",i+1, my_array [i]);
   }
   return 0;
}
```

Mengelola larik 1 dimensi

```
# include <stdio .h>
int main (){
   int index , nilai [10];

   /* input nilai mahasiswa */
```

```
   printf (" Input nilai 10 mahasiswa : \n");
   for( index =0; index < 10; index ++) {
      printf (" Mahasiswa %d : ", index +1) ;
      scanf ("%d", & nilai [ index ]);
   }
   /* tampilkan nilai mahasiswa */
   printf ("\ nNilai mahasiswa yang telah diinput \n");
   for( index =0; index < 10; index ++) {
      printf (" Mahasiswa %d : %d \n",index +1, nilai [ index ]);
   }
   return 0;
}
```

Mengelola larik 1 dimensi dengan kondisi

```
# include <stdio .h>
int main (){
   int index , nilai [10];

   /* input nilai mahasiswa */
   printf (" Input nilai 10 mahasiswa : \n");
   for( index =0; index < 10; index ++) {
      printf (" Mahasiswa %d : ", index +1) ;
      scanf ("%d", & nilai [ index ]);
   }
   /* tampilkan nilai mahasiswa diatas 60 */
   printf ("\ nNilai mahasiswa yang telah diinput \n");
   for( index =0; index < 10; index ++) {
      if( nilai [ index ] >=60) {
         printf (" Mahasiswa %d Lulus : %d \n",index +1, nilai [ index
]);
      } else {
         printf (" Mahasiswa %d Tidak Lulus : %d \n",index +1, nilai [
index ]) ;
      }
   }
}
```

10.2 Larik Multi Dimensi

Array multi-dimensi merupakan array yang mempunyai ukuran lebih dari dua. Bentuk pendeklarasian array sama saja dengan array dimensi satu maupun array dimensi dua.

10.2.1 Deklarasi

Pendeklarasian larik 2 dimensi mempunyai pola tipeVariabel namaVariabel[ukuran1]

[ukuran2].

Contohnya:

```
int matriksA[3][4];
float matrikB[4][4];
```

Pendeklarasian larik 3 dimensi mempunyai pola tipeVariabel namaVariabel[ukuran1] [ukuran2] [ukuran3].

Contohnya:

```
int matriksA[3][4][2];
float matrikB[4][4][2];
```

10.2.2 Pengisian Larik Multi Dimensi

Pengisian larik saat deklarasi

```
int array2D[5][2]={{1,12},{2,22},{3,33},{4,44},{5,55}};
```

Pengisian setelah deklarasi

```
int array2D[5][2];
array2D[0][0]=1;
array2D[0][1]=2;
array2D[1][0]=2;
array2D[1][1]=22;
```

10.2.3 Menampilkan Larik Multi Dimensi

Berikut ini contoh menampilkan larik multidimensi untuk larik dua dimensi

```
# include <stdio.h>

int main (){
   int array2D [5][2]={{1 ,12} ,{2 ,22} ,{3 ,33} ,{4 ,44} ,{5 ,55}};
   int i,j;

   printf (" array2D [%d ][% d] = %d\n" ,0,0, array2D [0][0]);
   printf (" array2D [%d ][% d] = %d\n" ,0,1, array2D [0][1]);
   printf (" array2D [%d ][% d] = %d\n" ,1,0, array2D [1][0]);
   printf (" array2D [%d ][% d] = %d\n" ,1,1, array2D [1][1]);
   printf (" array2D [%d ][% d] = %d\n" ,2,0, array2D [2][0]);
   printf (" array2D [%d ][% d] = %d\n" ,2,1, array2D [2][1]);
   printf (" array2D [%d ][% d] = %d\n" ,3,0, array2D [3][0]);
   printf (" array2D [%d ][% d] = %d\n" ,3,1, array2D [3][1]);
```

```
   printf (" array2D [%d ][% d] = %d\n" ,4,0, array2D [4][0]);
   printf (" array2D [%d ][% d] = %d\n" ,4,1, array2D [4][1]);
   return 0;
}
```

Menampilkan larik dua dimensi dengan perulangan

```
# include <stdio .h>

int main (){
int array2D [5][2]={{1 ,12} ,{2 ,22} ,{3 ,33} ,{4 ,44} ,{5 ,55}};
int i,j;

for(i =0;i <5; i ++) {
   for(j =0;j <2; j ++) {
      printf (" array2D [%d ][% d] = %d\n",i,j, array2D [i][j]);
   }
}
printf ("\n");
for(i =0;i <5; i ++) {
   for(j =0;j <2; j ++) {
      printf ("%d ",array2D [i][j]);
   }
   printf ("\n");
}
printf ("\n");
i =0;
   while (i <5) {
      j =0;
      while (j <2) {
         printf ("%d ",array2D [i][j]);
         j ++;
      }
      i ++;
      printf ("\n");
   }
}
```

Mengelola Larik 2 Dimensi

```
# include <stdio.h>

int main (){

   int baris , kolom , matriks [3][4];
   // Input elemen array
```

```
   printf (" Input elemen Array : \n");
   for( baris =0; baris <3; baris ++) {
      for( kolom =0; kolom <4; kolom ++) {
         printf (" matriks [%i ][% i] : ", baris +1, kolom +1) ;
         scanf ("%d", & matriks [ baris ][ kolom ]);
      }
      printf ("\n");
   }
   // Tampilkan elemen Array
   printf (" Isi array : \n");
   for( baris =0; baris <3; baris ++) {
      for( kolom =0; kolom <4; kolom ++) {
         printf ("%d ", matriks [ baris ][ kolom ]);
      }
   printf ("\n");
   }
}
```

Menyalin larik dua dimensi

```
# include <stdio.h>

int main (){

   int baris , kolom , matriksA [3][4] , matriksB [3][4];

   // Input elemen MatriksA
   printf (" Input elemen MatriksA : \n");
   for( baris =0; baris <3; baris ++) {
      for( kolom =0; kolom <4; kolom ++) {
         printf (" matriksA [%i ][% i] : ", baris +1, kolom +1) ;
         scanf ("%d", & matriksA [ baris ][ kolom ]);
      }
      printf ("\n");
   }

   // salin matriksA ke matriksB
   for( baris =0; baris <3; baris ++) {
      for( kolom =0; kolom <4; kolom ++) {
         matriksB [ baris ][ kolom ]= matriksA [ baris ][ kolom ];
      }
   }

   // Tampilkan matriksA dan matriksB
   printf (" Isi Matriks A : \n");
```

```
   for( baris =0; baris <3; baris ++) {
      for( kolom =0; kolom <4; kolom ++) {
         printf ("%d ", matriksA [ baris ][ kolom ]);
      }
   printf ("\n");
   }
   printf (" Isi Matriks B : \n");
   for( baris =0; baris <3; baris ++) {
      for( kolom =0; kolom <4; kolom ++) {
         printf ("%d ", matriksB [ baris ][ kolom ]);
      }
      printf ("\n");
   }
}
```

BAGIAN 11
MODULAR PROGRAMMING

Capaian Pembelajaran
1. Mahasiswa mampu menjelaskan konsep Pemrograman Modular
2. Mahasiswa mampu menjelaskan kelebihan dan kekurangan Pemrograman Modular
3. Mahasiswa mampu membuat rancangan konsep pemrograman modular sederhana dengan bantuan flowchart dan deskripsi singkat dari program tersebut

11.1 Pendahuluan

Pemrograman yang saat ini kita sudah pelajari dari beberapa contoh adalah program yang masih tergolong sederhana. Program yang pendek dan sederhana tersebut tentunya bila dibutuhkan proses maintanance atau perawatan dan juga penuluran kesalahan masih tergolong mudah. Namun bayangkan bila program yang mempunyai skala yang besar melibatkan banyak proses perhitungan maka sudah barang tentu sulit bila menggunakan cara yang selama ini kita gunakan. Kesulitan-kesulitan yang timbul tersebut diantaranya:

- Sulit mencari dan mengingat variabel-variabel yang sudah dideklarasikan
- Sulit melakukan dokumentasi
- Sulit mencari kesalahan program
- Sulit melihat efisiensi algoritma
- Kode program kadang ditulis berulang-ulang padahal mengerjakan suatu hal yang sama

Untuk mengatasi kesulitan-kesulitan tersebut lahirlah sebuah metode modular programming, yaitu memecah program yang besar tersebut ke dalam bagian-bagian kecil yang dinamakan sebagai modul.

Modular programming merupakan paradigma pemrograman yang pertama kali diperkenalkan oleh Information & Systems Institute, Inc. Paradigma ini diperkenalkan pada the National Symposium on Modular Programming pada 1968. Salah satu dari tokoh modular programming adalah Larry Constantine.

Terdapat beberapa keuntungan dari penggunaan paradigma modular programming ini, diantaranya adalah:

- Program lebih pendek
- Mudah dibaca dan dimengerti
- Mudah didokumentasi
- Mengurangi kesalahan dan mudah mencari kesalahan
- Kesalahan yang terjadi bersifat "lokal"

Bahasa Pemrograman C mendukung modular programming dalam pembuatan program. Pada hakikatnya sejak awal penggunaan bahasa C sudah membagi programprogramnya menjadi modul-modul. Perhatikan bahwa setiap kita membuat sebuah program kita menyertakan main() yang merupakan modul utama. Modul pada bahasa C dikenal dengan nama fungsi (function). Bahasa C terdiri dari fungsi-fungsi, baik yang langsung dideklarasikan dalam program ataupun dipisah di dalam header file. Pada contoh-contoh terdahulu kita sering memanggil <stdio.h> dengan perintah #include <stdio.h> yang merupakan library yang berisi modul-modul yang mendukung untuk proses input dan output.

11.2 Fungsi

Fungsi/function adalah suatu kumpulan instruksi/perintah/program yang dikelompokkan menjadi satu Fungsi sendiri letaknya terpisah dari program yang menggunakan fungsi tersebut. Fungsi memiliki nama tertentu yang unik, dan digunakan untuk mengerjakan suatu tujuan tertentu. Dalam bahasa pemrograman lain fungsi dapat disebut sebagai subroutine (basic, VB) atau procedure (Pascal, Delphi).

Keuntungan menggunakan fungsi diantaranya yaitu:

1. Dapat melakukan pendekatan top-down dan divide-andconquer:
    - Top-down: penelusuran program mudah
    - Divide-and-conquer: program besar dapat dipisah menjadi program-program kecil.
2. Kode program menjadi lebih pendek, mudah dibaca, dan mudah dipahami
3. Program dapat dikerjakan oleh beberapa orang sehingga program cepat selesai dengan koordinasi yang mudah.
4. Mudah dalam mencari kesalahan-kesalahan karena alur logika jelas dan sederhana

5. Kesalahan dapat dilokalisasi dalam suatu modul tertentu saja.
6. Modifikasi program dapat dilakukan pada suatu modul tertentu saja tanpa mengganggu program keseluruhan
7. Fungsi – fungsi menjadikan program mempunyai struktur yang jelas.
8. Dengan memisahkan langkah – langkah detail ke satu atau lebih fungsi – fungsi, maka fungsi utama (main) akan menjadi lebih pendek, jelas dan mudah dimengerti.
9. Fungsi –fungsi digunakan untuk menghindari penulisan program yang sama yang ditulis secara berulang – ulang.
10. Langkah – langkah tersebut dapat dituliskan sekali saja secara terpisah dalam bentuk fungsi.
11. Selanjutnya bagian program yang membutuhkan langkah – langkah ini tidak perlu selalu menuliskannya, tidak cukup memanggil fungsi tersebut.
12. Mempermudah dokumentasi.
13. Reusability: Suatu fungsi dapat digunakan kembali oleh program atau fungsi lain

11.3 Kategori Fungsi Dalam C

Fungsi dalam bahasa C dapat dikelompokkan menjadi dua kategori yaitu:

- Standard Library Function

Yaitu fungsi-fungsi yang telah disediakan oleh C dalam file-file header atau librarynya. Misalnya: scanf(), printf(), getch(). Untuk function ini kita harus mendeklarasikan terlebih dahulu library yang akan digunakan, yaitu dengan menggunakan preprosesor directive. Misalnya: #include <conio.h>

- Programmer-Defined Function

Merupakan function yang dibuat oleh programmer sendiri. Function ini memiliki nama tertentu yang unik dalam program, letaknya terpisah dari program utama, dan bisa dijadikan satu ke dalam suatu library buatan programmer itu sendiri yang kemudian juga diinclude-kan jika ingin menggunakannya.

11.4 Perancangan Fungsi

Dalam membuat fungsi, perlu diperhatikan:

- Data yang diperlukan sebagai inputan (input)
- Informasi apa yang harus diberikan oleh fungsi yang dibuat ke pemanggilnya (proses)
- Algoritma apa yang harus digunakan untuk mengolah data menjadi informasi (proses)
- Output fungsi yang bersifat opsional yang berasal dari proses perhitungan

11.5 Deklarasi Fungsi

Deklarasi fungsi diakhiri dengan titik koma. Tipe_data dapat berupa segala tipe data yang dikenal C ataupun tipe data buatan, namun tipe data dapat juga tidak ada dan digantikan dengan void yang berarti fungsi tersebut tidak mengembalikan nilai apapun. Nama fungsi adalah nama yang unik. Argumen dapat ada atau tidak (opsional) yang digunakan untuk menerima argumen/parameter. Antar argumen-argumen dipisahkan dengan menggunakan tanda koma. Perhatikan format umum penulisan fungsi berikut:

```
tipe_data_keluaran nama_fungsi (tipe_data_1 nama_variabel_1,...,
tipe_data_n nama_variabe_n){
   Proses
   return variabel_keluaran
}
```

Pada format tersebut **tipe_data_keluaran**, dapat berupa salah satu tipe data C, misalnya char atau int. Jika penentu tipe tidak disebutkan maka dianggap bertipe int. **nama_fungsi** merupakan nama dari fungsi yang digunakan untuk memanggil fungsi tersebut. **tipe_data_1 nama_variabel_1** merupakan parameter yang diterima oleh fungsi untuk diolah pada proses yang terdapat didalam fungsi . **proses**, berisi deklarasi variabel (jika ada) dan program yang akan melakukan tugas yang akan diberikan kepada fungsi. **return variabel_keluaran**, nilai balikan yang akan dikirim oleh fungsi kepada yang memanggil fungsi tersebut.

Perhatikan contoh deklarasi fungsi berikut:

```
int nilaiTerkecil ( int a, int b, int c){
   int keluaran ;
   // proses
   return keluaran ;
}
```

Untuk memanggil fungsi tersebut adalah dengan menuliskan nama fungsinya.

```
minimal = nilaiTerkecil (3 ,8 ,5);
```

Perhatikan contoh lengkap program yang menggunakan fungsi berikut ini:

```
# include <stdio .h>

int hasilTambah ( int x, int y, int z){
   int penjumlahan ;
   penjumlahan = x + y + z;
   return penjumlahan ;
}

int hasilKurang ( int x, int y, int z){
   int pengurangan ;
   pengurangan = x - y - z;
   return pengurangan ;
}

int main (){
   int a,b,c, hasil ;
   a = 1; b = 2; c = 3;
   hasil = hasilTambah (a, b, c);
   printf (" Hasil penjumlahan : %d\n", hasil );
   hasil = hasilKurang (a, b, c);
   printf (" Hasil pengurangan : %d\n", hasil );
}
```

Pada contoh tersebut terdapat dua fungsi yang dibuat bernama hasilKurang dan hasilTambah. Kedua fungsi tersebut kemudian dipanggil di fungsi utama yaitu fungsi main().

11.6 Penempatan Fungsi

Fungsi dapat diletakan sebelum program utama atau diletakan setelah program utama. Pada conrog sebelumnya kita telah mencoba membuat fungsi dengan menempatkannya di atas fungsi main(). Jika diletakan setelah program utama maka fungsi perlukan deklarasi prototipe fungsi terlebih dahulu di atas program utama sedangkan detail dari fungsi diletakan setelah program utama. Namun jika fungsi diletakan sebelum program utama maka deklarasi dan isi fungsi semuanya diletakan pada bagian atas program utama. Prototipe fungsi berupa :

1. Nama Fungsi
2. Tipe nilai fungsi

3. Jumlah dan tipe parameter
4. Dan diakhiri dengan titik koma, sebagaimana pada pendeklarasian variabel.
5. Untuk alasan dokumentasi program yang baik, sebaiknya semua fungsi yang digunakan dideklarasikan terlebih dahulu dengan menggunaan prototipe
6. Deklarasi fungsi (prototipe) ditulis sebelum fungsi tersebut digunakan

Perhatikan contoh kode berikut yang merupakan program yang menggunakan prototipe fungsi:

```
# include <stdio .h>

/* prototype fungsi tambah () , ada titik koma */
float tambah ( float x, float y);

int main (){
   float a, b, c;
   printf ("A = "); scanf ("%f", &a);
   printf ("B = "); scanf ("%f", &b);
   c = tambah (a, b); /* pemanggilan fungsi tambah () */
   printf ("A + B = %.2 f", c);
   return 0;
}

 /* Definisi fungsi , tanpa titik koma */
float tambah ( float x, float y){
   return (x+y); /* Nilai balik fungsi */
}
```

11.7 Jenis Fungsi

Fungsi jika dilihat dari cara pengembalian nilai terbagi menjadi dua jenis, yaitu Fungsi yang tidak mengembalikan nilai (void) dan fungsi yang mengembalikan nilai (nonvoid). Fungsi yang void sering disebut juga prosedur. Disebut void karena fungsi tersebut tidak mengembalikan suatu nilai keluaran yang didapat dari hasil proses fungsi tersebut. Beberaoa ciri dari fungsi yang tidak mengembalikan nilai adalah tidak adanya keyword return. Selain itu tidak adanya tipe data di dalam deklarasi fungsi. Serta ciri yang lain yaitu menggunakan keyword void. Keyword void sendiri juga digunakan jika suatu function tidak mengandung suatu parameter apapun.

Berbeda halnya dengan fungsi void, fungsi non-void disebut juga dengan function. Suatu fungsi dikatakan non-void ketika mengembalikan nilai kembalian yang berasal dari keluaran hasil proses function tersebut. Beberapa cirinya yaitu adanya keyword return, ada tipe data yang mengawali deklarasi fungsi, tidak ada keyword void, memliki nilai kembalian. Dapat dianalogikan sebagai suatu variabel yang memiliki tipe data tertentu sehingga dapat langsung ditampilkan hasilnya. Perhatikan contoh kode berikut untuk melihat perbedaan fungsi void dan non-void:Keyword void juga digunakan jika suatu function tidak mengandung suatu parameter apapun

```
//ini merupakan contoh fungsi void
void tampilkan_jumlah(int a, int b){
   int jumlah;
   jumlah = a+b;
   printf("%d",jumlah);
}

//ini merupakan contoh fungsi non void
int jumlah(int a,int b){
   int jml;
   jml = a+b;
   return jml;
}
```

BAGIAN 12
PUSTAKA C

12.1 Preprocessor

Preprocessor dari bahasa c merupakan sebuah alat teks substitusi dan yang menginstruksikan compiler untuk melakukan suatu pemrosesan sebelum pemrosesan utama dilakukan. Preprocessor yang selama ini biasa digunakan adalah #include <stdio.h> yang merupakan library standar dari bahasa C yang biasa digunakan untuk melakukan proses input dan output. Semua preprocessor dalam bahasa C dimulai dengan simbol pagar (#) atau biasa disebut dengan simbol hash.

Terdapat beberapa macam preprocessor yang ada dalam bahasa C, diantaranya dapat dilihat pada Tabel 12.1

Tabel 12.1: Beberapa Macam Preprocessor

Directive	Deskripsi
#define	Mensubstitusi macro. Dapat juga digunakan untuk mendeklarasikan konstanta
#include	Memasukkan file kepala dari file lain.
#undef	Macro yang belum didefinisikan
#ifdef	Mengembalikan nilai true jika macro didefinisikan
#ifndef	Mengembalikan nilai tru jika macro tidak didefinisikan
#if	Menguji jika kondisi waktu kompilasi bernilai true
#else	Jika tidak true
#elif	Statement else if
#endif	Akhir dari kondisi if pada preprocessor
#error	Menampilkan pesan error pada stderr
#pragma	Membuat perintah khusus untuk compiler, dengan menggunakan metode standar

Karena pembahasan kita sempitkan pada pustaka bahasa C maka yang kita gunakan adalah preprocessor #include.

12.2 Pustaka Standar Bahasa C

Pustaka pada bahasa C ada yang merupakan pustaka standar, ada pula yang merupakan

pustaka buatan dari programmer. Pustaka standar bahasa C merupakan kumpulan dari fungsi bawaan bahasa C, konstanta, berkas header seperti <stdio.h>, <string.h>, <math.h> dan lain-lain. Beberapa pustaka standar bahasa C diantaranya dapat dilihat pada Tabel 12.2

Tabel 12.2: Daftar pustaka standar bahasa C

Berkas Header	Kegunaan
<assert.h>	Fungsi program assertion, yaitu yang digunakan untuk kepentingan tes diagnostik di program
<ctype.h>	Kumpulan Fungsi tipe karakter , berisi makro dan deklarasi fungsi yang berguna untuk memproses karakter. Sebagai contoh untuk mengetahui suatu huruf berupa hutuf kapital atau bukan.
<locale.h>	Kumpulan Fungsi Lokalisasi
<math.h>	Kumpulan Fungsi matematika, yaitu berisi makro dan deklarasi fungsi yang terkait dengan operasi matematika misalnya untuk menghitung perpangkatan
<setjmp.h>	Kumpulan Fungsi perlompatan
<signal.h>	Kumpulan Fungsi penanganan sinyal
<stdarg.h>	Kumpulan Fungsi penanganan argumen variabel
<stdio.h>	Kumpulan Fungsi standar input dan output
<stdlib.h>	Kumpulan Fungsi standar utilitas, yang diantaranya adalah fungsi untuk mengkonversi string menjadi bilangan
<string.h>	Kumpulan Fungsi penanganan string
<time.h>	Kumpulan Fungsi penanganan operasi tanggal dan waktu

12.2.1 Pustaka Penanganan Tanggal dan Waktu

Untuk penanganan tanggal dan waktu C memiliki sebuah pustaka sendiri yaitu time.h. Pustaka tersebut memiliki kumpulan fungsi diantaranya.

Fungsi	Deskripsi
difftime	menghitung perbedaan waktu dari dua nilai time_t
time	Mengembalikan nilai waktu saat ini
clock	Mengembalikan nilai perhitungan clock dari prosesor
ctime	Melakukan konversi nilai time_t ke representasi tekstual
strftime	Melakukan konversi objek struct tm pada representasi tekstual
wcsftime	Melakukan konversi objek struct tm ke representasi custom wide string
gmtime	Melakukan konversi nilai time_t ke ekspresi kalender dengan koordinat universal GMT
localtime	Melakukan konversi nilai time_t ke ekspresi kalender dengan format waktu lokal
mktime	Melakukan konversi nilai waktu kalender time_t

Perhatikan contoh penerapan salah satu dari fungsi yang terdapat pada library time.h berikut ini

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

int main ()
{
   time_t waktu_mulai, waktu_selesai;
   double selisih;

   printf("Program dimulai...\n");
   time(&waktu_mulai);

   printf("Tunda selama 5 detik...\n");
   sleep(5);

   time(&waktu_selesai);
   selisih = difftime(waktu_selesai, waktu_mulai);

   printf("waktu eksekusi = %f\n", selisih);
   printf("Program selesai...\n");

   return 0;
}
```

Bila program tersebut dijalankan akan menghasilkan output

```
Program dimulai...
Tunda selama 5 detik...
waktu eksekusi = 5.000000
Program selesai...
```

12.2.2 Pustaka Assert

Pustaka assert merupakan pustaka yang memiliki Fungsi program assertion, yaitu yang digunakan untuk kepentingan tes diagnostik di program. Fungsi assertion akan memberhentikan jalannya program bila input dari argumen assertion tidak memenuhi persyaratan dari program. Atau biasa pula hal ini disebut dengan sebuah validasi nilai. Perhatikan contoh kode program berikut:

```c
#include <assert.h>
#include <stdio.h>
int main()
{
   int a;

   printf("Masukkan bilangan bulat yang lebih dari 10: ");
   scanf("%d", &a);
   assert(a >= 10);
   printf("Bilangan bulat yang anda masukkan adalah %d\n", a);

   return 0;
}
```

Bila program tersebut dijalankan dan diberi input angka 5 maka program akan menghasilkan output di layar

```
Masukkan bilangan bulat yang lebih dari 10: 5
waktu: /home/wira/Documents/asersi.c:9: main: Assertion `a >= 10'
failed.
Aborted
```

12.2.3 Pustaka Matematika

Pustaka matematika merupakan kumpulan fungsi yang digunakan untuk operasi matematika. Adapun library yang digunakan adalah bernama math.h. Oleh karena itu penggunaan library ini mengharuskan programmer untuk menuliskan #include <math.h>. Beberapa diantara fungsi matemarika yang ada pada library math.h diantaranya dapat dilihat

pada Tabel 12.3

Tabel 12.3: Fungsi-fungsi pada pustaka matematika

Fungsi	Keterangan
`fabs()`	Memberikan nilai kembalian berupa nilai yang absolut
`ceil()`	Fungsi yang berguna untuk memperoleh bilangan bulat terkecil yang tidak kurang dari nilai argumennya
`floor()`	Fungsi yang berguna untuk memperoleh bilangan bulat yang tidak lebih besar daripada nilai argumennya
`round()`	Fungsi yang berguna untuk mendapatkan bilangan bulat yang terdekat dengan argumennya
`trunc()`	Fungsi yang berguna utuk memperoleh bilangan bulat argumennya
`pow()`	Fungsi yang berguna untuk menghitung perpangkatan
`pow10()`	Fungsi yang berguna untuk menghitung perpangkatan 10^x
`sqrt()`	Fungsi yang berguna untuk perhitungan akar
`cbrt()`	Fungsi yang berguna untuk menghitung $x^{1/3}$
`hypot()`	Fungsi yang berguna untuk menghitung sisi miring segitiga siku-siku
`log()`	Fungsi yang berguna untuk menghitung logaritma alami
`log10()`	Fungsi yang berguna untuk menghitung logaritma basis 10
`log2()`	Fungsi yang berguna untuk menghitung logaritma basis 2
`exp()`	Fungsi yang berguna untuk menghitung nilai eksponensial
`exp2()`	Fungsi yang berguna untuk menghitung 2^x
`fmod()`	Fungsi yang berguna untuk menghitung sisa pembagian x dengan y dari argumennya
`cos()`	Fungsi yang berguna untuk memperoleh kosinus dari nilai argumennya
`sin()`	Fungsi yang berguna untuk memperoleh sinus dari niai argumennya
`tan()`	Fungsi yang berguna untuk memperoleh tangen nilai argumennya
`acos()`	Fungsi yang berguna untuk menghitung arc cosinus dari argumennya
`asin()`	Fungsi yang berguna untuk menghitung arc sinus dari argumennya
`atan()`	Fungsi yang berguna untuk menghitung arc tangen dari argumennya
`atan2()`	Fungsi yang berguna untuk menghitung arc tangen berdasarkan y/x dari argumennya
`cosh()`	Fungsi yang berguna untuk menghitung hyperbolic cosinus
`sinh()`	Fungsi yang berguna untuk menghitung hyperbolic sinus
`tanh()`	Fungsi yang berguna untuk menghitung hyperbolic tangen

`acosh()`	Fungsi yang berguna untuk menghitung arc cosinus hyperbolic
`asinh()`	Fungsi yang berguna untuk menghitung arc sinus hyperbolic
`atanh()`	Fungsi yang berguna untuk menghitung arc tangen hyperbolic

Dari Tabel 12.3 tersebut kita akan coba membuat contoh dari beberapa fungsi yang ada tersebut.

12.2.3.1 Fungsi fabs()

Fungsi fabs merupakan fungsi yang memberikan nilai kembalian berupa nilai absolut. Maka pada fungsi tersebut bila dijalankan akan memberi nilai posiotif dari argumen yang dimasukannya. Perhatikan contoh berikut:

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

int main ()
{
   int a, b;
   a = 1234;
   b = -344;

   printf("Nilai absolut dari %d adalah %lf\n", a, fabs(a));
   printf("Nilai absolut dari %d adalah %lf\n", b, fabs(b));

   return 0;
}
```

Program tersebut akan menghasilkan sebuah output yaitu

```
Nilai absolut dari 1234 adalah 1234.000000
Nilai absolut dari -344 adalah 344.000000
```

12.2.3.2 ceil()

Fungsi ceil merupakan Fungsi yang berguna untuk memperoleh bilangan bulat terkecil yang tidak kurang dari nilai argumennya. Perhatikan contoh kode program berikut:

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

```
 3
 4 int main ()
 5 {
 6    float nilai_1, nilai_2, nilai_3, nilai_4;
 7
 8    nilai_1 = 1.6;
 9    nilai_2 = 1.2;
10    nilai_3 = 2.8;
11    nilai_4 = 2.3;
12
13    printf ("nilai ke 1 = %.1lf\n", ceil(nilai_1));
14    printf ("nilai ke 2 = %.1lf\n", ceil(nilai_2));
15    printf ("nilai ke 3 = %.1lf\n", ceil(nilai_3));
16    printf ("nilai ke 4 = %.1lf\n", ceil(nilai_4));
17
18    return 0;
19 }
```

maka program tersebut akan menghasilkan output berupa

```
nilai ke 1 = 2.0
nilai ke 2 = 2.0
nilai ke 3 = 3.0
nilai ke 4 = 3.0
```

12.2.3.3 Fungsi floor()

Fungsi floor yaitu Fungsi yang berguna untuk memperoleh bilangan bulat yang tidak lebih besar daripada nilai argumennya. Perhatikan contoh berikut

```
 1 #include <stdio.h>
 2 #include <math.h>
 3
 4 int main ()
 5 {
 6    float nilai_1, nilai_2, nilai_3, nilai_4;
 7
 8    nilai_1 = 1.6;
 9    nilai_2 = 1.2;
10    nilai_3 = 2.8;
11    nilai_4 = 2.3;
12
13    printf("nilai ke 1 = %.1lf\n", floor(nilai_1));
```

```
14     printf("nilai ke 2 = %.1lf\n", floor(nilai_2));
15     printf("nilai ke 3 = %.1lf\n", floor(nilai_3));
16     printf("nilai ke 4 = %.1lf\n", floor(nilai_4));
17
18     return 0;
19 }
```

Bila kode program tersebut dijalankan maka akan menghasilkan suatu output

```
nilai ke 1 = 1.0
nilai ke 2 = 1.0
nilai ke 3 = 2.0
nilai ke 4 = 2.0
```

12.2.3.4 Fungsi round()

Fungsi round merupakan Fungsi yang berguna untuk mendapatkan bilangan bulat yang terdekat dengan argumennya. Perhatikan contoh kode program berikut

```
 1 #include <stdio.h>
 2 #include <math.h>
 3
 4 int main ()
 5 {
 6     float nilai_1, nilai_2, nilai_3, nilai_4;
 7
 8     nilai_1 = 1.6;
 9     nilai_2 = 1.2;
10     nilai_3 = 2.8;
11     nilai_4 = 2.3;
12
13     printf("nilai ke 1 = %.1lf\n", round(nilai_1));
14     printf("nilai ke 2 = %.1lf\n", round(nilai_2));
15     printf("nilai ke 3 = %.1lf\n", round(nilai_3));
16     printf("nilai ke 4 = %.1lf\n", round(nilai_4));
17
18     return 0;
19 }
```

Bila kode program tersebut dijalankan maka akan menghasilkan suatu output

```
nilai ke 1 = 2.0
```

```
nilai ke 2 = 1.0
nilai ke 3 = 3.0
nilai ke 4 = 2.0
```

12.2.3.5 Fungsi trunc()

Fungsi trunc merupakan Fungsi yang berguna utuk memperoleh bilangan bulat argumennya. Perhatikan contoh

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

int main()
{
   printf ("Nilai bulat dari 16.99 = %f\n", trunc (16.99) );
   printf ("Nilai bulat dari 20.1  = %f\n", trunc (20.1) );
   return 0;
}
```

Bila kode program tersebut dijalankan maka akan menghasilkan suatu output

12.2.3.6 Fungsi pow()

Fungsi pow atau power merupakan Fungsi yang berguna untuk menghitung perpangkatan. Perhatikan contoh kode program berikut

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

int main ()
{
   printf("Nilai dari 8.0 dipangkatkan dengan 3 = %lf\n", pow(8.0, 3));

   printf("Nilai dari 3.05 dipangkatkan dengan 1.98 = %lf", pow(3.05, 1.98));

   return 0;
}
```

Bila kode program tersebut dijalankan maka akan menghasilkan suatu output

```
Nilai dari 8.0 dipangkatkan dengan 3 = 512.000000
```

```
Nilai dari 3.05 dipangkatkan dengan 1.98 = 9.097324
```

12.2.3.7 Fungsi pow10()

Fungsi pow10 merupakan ungsi yang berguna untuk menghitung perpangkatan 10^x . Perhatikan contoh kode program berikut

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

int main()
{
   int x = 5;
   double hasil;

   hasil = pow10(x);

   printf("Sepuluh dipangkatkan dengan %d hasilnya adalah %lf\n", x, hasil);

   return 0;
}
```

Bila kode program tersebut dijalankan maka akan menghasilkan suatu output

```
Sepuluh dipangkatkan dengan 5 hasilnya adalah 100000.000000
```

12.2.3.8 Fungsi sqrt()

Fungsi pow atau square root merupakan Fungsi yang berguna untuk perhitungan akar. Perhatikan contoh kode program berikut

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

int main ()
{

   printf("Akar dari %lf adalah %lf\n", 4.0, sqrt(4.0) );
   printf("Akar dari %lf adalah %lf\n", 5.0, sqrt(5.0) );

   return 0;
```

```
11 }
```

Bila kode program tersebut dijalankan maka akan menghasilkan suatu output

```
Akar dari 4.000000 adalah 2.000000
Akar dari 5.000000 adalah 2.236068
```

12.2.3.9 Fungsi cbrt()

Fungsi cbrt() atau cube root merupakan Fungsi yang berguna untuk menghitung $x^{1/3}$. Perhatikan contoh kode program berikut

```
 1 #include <stdio.h>
 2 #include <math.h>
 3
 4 int main()
 5 {
 6     double angka = 6, akarpkttiga;
 7
 8     akarpkttiga =  cbrt(angka);
 9     printf("Akar pangkat tiga dari %lf =  %lf", angka, akarpkttiga);
10
11     return 0;
12 }
```

Bila kode program tersebut dijalankan maka akan menghasilkan suatu output

```
Akar pangkat tiga dari 6.000000 =  1.817121
```

12.2.3.10 Fungsi hypot()

Fungsi hypot atau hypotenuse merupakan Fungsi yang berguna untuk menghitung sisi miring segitiga siku-siku. Perhatikan contoh kode program berikut

```
1 #include <stdio.h>
2 #include <math.h>
3
4 int main()
5 {
```

```
    double p, b;
    double sisi_miring;

    p = 5.0;
    b = 12.0;

    sisi_miring = hypot(p, b);

    printf("Sisi miring dari (%.2lf, %.2lf) = %.2lf", p, b, sisi_miring);

    return 0;
}
```

Bila kode program tersebut dijalankan maka akan menghasilkan suatu output

```
Sisi miring dari (5.00, 12.00) = 13.00
```

12.2.3.11 Fungsi log()

Fungsi log merupakan Fungsi yang berguna untuk menghitung logaritma alami. Perhatikan contoh kode program berikut

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

int main ()
{
   double x, y;
   x = 2.7;

   y = log(x);
   printf("log dari %lf = %lf", x, y);

   return 0;
}
```

Bila kode program tersebut dijalankan maka akan menghasilkan suatu output

```
log dari 2.700000 = 0.993252
```

12.2.3.12 Fungsi exp()

Fungsi exp merupakan Fungsi yang berguna untuk menghitung nilai eksponensial. Perhatikan contoh kode program berikut

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

int main ()
{
   double x = 0;

   printf("nilai ekponensial dari %lf = %lf\n", x, exp(x));
   printf("nilai ekponensial dari %lf = %lf\n", x+1, exp(x+1));
   printf("nilai ekponensial dari %lf = %lf\n", x+2, exp(x+2));

   return 0;
}
```

Bila kode program tersebut dijalankan maka akan menghasilkan suatu output

```
nilai ekponensial dari 0.000000 = 1.000000
nilai ekponensial dari 1.000000 = 2.718282
nilai ekponensial dari 2.000000 = 7.389056
```

12.2.3.13 Fungsi fmod()

Fungsi fmod merupakan Fungsi yang berguna untuk menghitung sisa pembagian x dengan y dari argumennya. Perhatikan contoh kode program berikut

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

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

```
    a = 9.2;
    b = 3.7;
    c = 2;
    printf("sisa bagi dari %f/%d = %lf\n", a, c, fmod(a,c));
    printf("sisa bagi dari %f/%f = %lf\n", a, b, fmod(a,b));

    return 0;
}
```

Bila kode program tersebut dijalankan maka akan menghasilkan suatu output

```
sisa bagi dari 9.200000/2 = 1.200000
sisa bagi dari 9.200000/3.700000 = 1.800000
```

12.2.3.14 Fungsi sin()

Fungsi sin atau sinus merupakan Fungsi yang berguna untuk memperoleh sinus dari niai argumennya. Perhatikan contoh kode program berikut

```

```

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

#define PI 3.14159265

int main ()
{
    double x, hasil, nilai;

    x = 45.0;
    nilai = PI / 180;
    hasil = sin(x*nilai);
    printf("Sinus dari %lf adalah %lf derajat", x, hasil);

    return 0;
}
```

Bila kode program tersebut dijalankan maka akan menghasilkan suatu output

```
Sinus dari 45.000000 adalah 0.707107 derajat
```

12.2.3.15 Fungsi asin()

Fungsi asin merupakan Fungsi yang berguna untuk menghitung arc sinus dari argumennya. Perhatikan contoh kode program berikut

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

#define PI 3.14159265

int main ()
{
   double x, hasil, nilai;
   x = 0.9;
   nilai = 180.0 / PI;

   hasil = asin(x) * nilai;
   printf("Arc sinus dari %lf adalah %lf derajat", x, hasil);

   return 0;
}
```

Bila kode program tersebut dijalankan maka akan menghasilkan suatu output

```
Arc sinus dari 0.900000 adalah 64.158067 derajat
```

12.2.3.16 Fungsi sinh()

Fungsi sinh atau sinus hyperbolic merupakan Fungsi yang berguna untuk menghitung hyperbolic sinus. Perhatikan contoh kode program berikut

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

int main ()
{
   double x, hasil;
```

```
    x = 0.5;

    hasil = sinh(x);
    printf("Sinus hiperbolic dari %lf adalah %lf derajat", x, hasil);

    return 0;
}
```

Bila kode program tersebut dijalankan maka akan menghasilkan suatu output

```
Sinus hiperbolic dari 0.500000 adalah 0.521095 derajat
```

12.2.3.17 Fungsi asinh()

Fungsi asinh merupakan Fungsi yang berguna untuk menghitung arc sinus hyperbolic. Perhatikan contoh kode program berikut

```
#include <stdio.h>
#include <math.h>
#define PI 3.141592654

int main()
{
    float angka = 8.0;
    double hasil;
    hasil = asinh(angka);

    printf("invers dari sinh(%.2f) = %.2f dalam radian", angka, hasil);

    hasil=(hasil*180)/PI;
    printf("\ninvers dari sinh(%.2f) = %.2f dalam derajat", angka, hasil);
    return 0;
}
```

Bila kode program tersebut dijalankan maka akan menghasilkan suatu output

```
invers dari sinh(8.00) = 2.78 dalam radian
invers dari sinh(8.00) = 159.08 dalam derajat
```

BAGIAN 13
PEMROSESAN STRING

13.1 Definisi String

Pada dasarnya string merupakan kumpulan dari character yang disatukan pada suatu array. Sehingga memungkinkan untuk string dipanggil berdasarkan character yang ke berapa. Oleh karena itu, maka deklarasi atau pendefinisikan dari string sama dengan deklarasi character, hanya saja menggunakan array. Untuk menampilkan string yang ada pada suatu array, format yang dipergunakan adalah "%s". Ketika diinstruksikan pada komputer untuk menampilkan string, komputer akan memeriksa satu persatu elemen array tersebut serta menampilkannya pada layar monitor. Komputer akan berhenti memeriksa elemen array yang selanjutnya jika menemukan null character '\o' . Sehingga dapat dikatakan misalnya kita memiliki sebuah string seperti di bawah ini

```
char namasaya[5];
```

Maka variabel namasaya dapat diisi dengan 5 character, dimulai dari karakter dengan index array 0 hingga index array 4. Sedangkan untuk character dengan index array ke 5 akan diisi oleh NULL value yaitu '\0'.

13.2 Inisialisasi String

String dapat dilakukan deklarasi dan inisialisasi dengan beberapa cara. Beberapa cara tersebut diantaranya adalah:

- Cara Pertama

```
char namasaya[]="Universitas Tanjungpura";
```

- Cara Kedua

```
char namasaya[]={'t','a','n','j','u','n','g','p','u','r','a','\0'};
```

- Cara Ketiga

```
char namasaya[7]="tanjung";
```

- Cara Keempat

```
char namasaya[11]={'t','a','n','j','u','n','g','p','u','r','a','\0'};
```

13.3 Input dan Output String

Untuk melakukan input dari string dapat dengan beberapa cara, salah satunya adalah dengan menggunakan scanf seperti input pada angka integer. Tapi perlu diperhatikan ketika menggunakan scanf ada sedikit tambahan konfigurasi yaitu adalah

```
scanf(" %[^\n]s",&namasaya);
```

Hal ini dikarenakan program juga akan membaca tombol spasi dan juga tombol enter.

Untuk menampilkan output string pada layar dapat menggunakan simbol %s. Perhatikan contoh lengkap berikut ini:

```
1 #include <stdio.h>
2
3 int main(){
4     char namasaya[15];
5     printf("Masukkan nama Anda : ");
6     scanf(" %[^\n]s",&namasaya);
7     printf("Assalamu 'alaikum, %s. Anda sedang belajar string.\n", namasaya);
8     return 0;
9 }
```

Atau dapat pula menggunakan fungsi **puts()** perhatikan contoh kode berikut yang merupakan modifikasi dari kode sebelumnya

```
#include <stdio.h>
int main(){
    char namasaya[15];
    printf("Masukkan nama Anda : ");
    scanf(" %[^\n]s",&namasaya);
    puts(namasaya);
    return 0;
}
```

Program tersebut bila dimasukkan input Dian Prawira maka akan menghasil output:

```
Masukkan nama Anda : Dian Prawira
Assalamu 'alaikum, Dian Prawira. Anda sedang belajar string.
```

Karena sifat string pada dasarnya adalah array dari char, maka dapat dipanggil per character saja. Misalnya kita memiliki sebuah string seperti di bawah ini

```
char namasaya[5]="Diana";
```

Maka deklarasi dan inisialisasi tersebut bila digambarkan pemetaannya pada ruang memory adalah

0	1	2	3	4	5
D	i	a	n	a	\0

Sehingga sebagai contoh bila kita ingin menampilkan huruf n maka dapat dengan cara

```
printf("%c",namasaya[3]);
```

Dalam pemberian nilai di string kita tidak bisa dengan tanda = cara langsung selayaknya memberikan nilai pada integer seperti contoh berikut:

```
namasaya = "Dian";
```

Tapi kita dapat menggunakan fungsi strcpy() yang merupakan bagian dari pustaka string.h, seperti berikut ini:

```
strcpy(namasaya,"Dian");
```

13.4 Pustaka String

Untuk melakukan pemrosesan string, bahasa pemrograman c juga memiliki pustaka tersendiri. Pustaka tersebut adalah string.h . Terdapat beberapa fungsi yang ada pada pustaka string.h . Antara lain dapat dilihat pada tabel

Tabel 13.1: Daftar Fungsi Pustaka String

Fungsi	Deskripsi
`memchr()`	Pencarian untuk kemunculan pertama dari karakter c (unsigned char) di n byte pertama dari string yang ditunjukkan, oleh str argumen.
`memcmp()`	Membandingkan n bytes pertama dari string pertama dan kedua
`memcpy()`	Menyalin n karakter dari sumber ke tujuan
`memmove()`	Fungsi lain untuk menyalin n karakter dari string perama ke string kedua
`memset()`	Fungsi untuk Menyalin karakter ke n karakter pertama dari string

`strcat()`	Menggabungkan satu string dengan string lain.
`strncat()`	Menggabungkan satu string dengan string lain yang tujuan string yang terbatas pada panjang beberapa karakter.
`strchr()`	Mencari suatu karakter pada suatu string
`strcmp()`	Membandingkan dua buah string
`strncmp()`	Membandingkan n bytes dari dua buah string
`strcoll()`	Membandingkan dua buah string yang hasilnya tergantung dari pengaturan lokasi pada LC_COLLATE
`strcpy()`	Menyalin string
`strncpy()`	Menyalin string dengan pembatasan karakter
`strcspn()`	Menghitung panjang segmen awal string yang pertama, yang seluruhnya terdiri dari karakter yang bukan di sting kedua.
`strerror()`	Mencari array internal untuk nomor kesalahan errnum dan mengembalikan memberikan nilai kembalian berupa pesan kesalahan dalam bentuk string.
`strlen()`	Menghitung panjang dari string
`strpbrk()`	Mencari karakter pertama dari string pertama yang cocok dengan string yang kedua
`strrchr()`	Mencari untuk kejadian terakhir dari karakter c (unsigned char) di string yang ditunjukkan oleh argumen.
`strspn()`	Menghitung panjang segmen awal string pertama yang seluruhnya terdiri dari karakter dalam string kedua.
`strstr()`	Mengambil sebagian string dari suatu posisi sub-string yang dicari. Ketika posisi sub-string ditemukan maka mulai posisi tersebut sampai akhir string merupakan hasilnya
`strtok()`	Membagi sebuah string menjadi beberapa string (token) berdasarkan character yang akan memisahkan string tersebut
`strxfrm()`	Mengubah karakter count pertama pada string yang ditunjuk oleh string kedua sehingga dapat digunakan oleh fungsi strcmp()

13.4.1 memchr()

Perhatikan contoh kode program berikut

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

int main ()
{
   const char stringnya[] = "http://www.untan.ac.id";
   const char tanda = '.';
   char *hasil;

   hasil = memchr(stringnya, tanda, strlen(stringnya));

   printf("String setelah tanda %c adalah %s\n", tanda, hasil);

   return 0;
}
```

Kode tersebut akan menghasilkan output di layar berupa

```
String setelah tanda . adalah .untan.ac.id
```

13.4.2 memcmp()

Perhatikan contoh kode program berikut

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

int main() {

   char pertama[15];
   char kedua[15];
   int hasil;

   memcpy(pertama, "abcdef", 6);
   memcpy(kedua, "ABCDEF", 6);

   hasil = memcmp(pertama, kedua, 5);

   if(hasil > 0){
      printf("kedua kurang dari yang pertama");
   }else if(hasil < 0){
      printf("pertama kurang dari yang kedua");
```

```
   }else{
      printf("pertama sama dengan yang kedua");
   }

   return 0;
}
```

Kode tersebut akan menghasilkan output di layar berupa

```
kedua kurang dari yang pertama
```

13.4.3 memcpy()

Perhatikan contoh kode program berikut

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

int main ()
{
   const char src[50] = "Universitas Tanjungpura";
   char tujuan[50];

   printf("Sebelum menggunakan fungsi memcpy tujuan = %s\n", tujuan);
   memcpy(tujuan, src, strlen(src)+1);
   printf("Setelah menggunakan fungsi memcpy tujuan = %s\n", tujuan);

   return 0;
}
```

Kode tersebut akan menghasilkan output di layar berupa

```
Sebelum menggunakan fungsi memcpy tujuan = Õ#s¨
Setelah menggunakan fungsi memcpy tujuan = Universitas Tanjungpura
```

13.4.4 memmove()

Perhatikan contoh kode program berikut

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

int main ()
{
   char tujuan[] = "stringlama";
   const char sumber[]  = "stringbaru";

   printf("Sebelum penggunaan memmove tujuan = %s, sumber = %s\n",
tujuan, sumber);
```

```
   memmove(tujuan, sumber, 9);
   printf("Setelah penggunaan memmove tujuan = %s, sumber = %s\n",
tujuan, sumber);

   return 0;
}
```

Kode tersebut akan menghasilkan output di layar berupa

```
Sebelum penggunaan memmove tujuan = stringlama, sumber = stringbaru
Setelah penggunaan memmove tujuan = stringbaru, sumber = stringbaru
```

13.4.5 memset()

Perhatikan contoh kode program berikut

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

int main ()
{
   char str[50];

   strcpy(str,"Ini adalah latihan string");
   puts(str);

   memset(str,'$',7);
   puts(str);

   return 0;
}
```

Kode tersebut akan menghasilkan output di layar berupa

```
Ini adalah latihan string
$$$$$$$lah latihan string
```

13.4.6 strcat()

Perhatikan contoh kode program berikut

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

int main ()
{
   char sumber[50], tujuan[50];

   strcpy(sumber, " Ini adalah sumber");
```

```
   strcpy(tujuan, "Ini adalah tujuan");

   strcat(tujuan, sumber);

   printf("Hasil dari penggabungan string : | %s |", tujuan);

   return 0;
}
```

Kode tersebut akan menghasilkan output di layar berupa

```
Hasil dari penggabungan string : | Ini adalah tujuan Ini adalah sumber |
```

13.4.7 strncat()

Perhatikan contoh kode program berikut

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

int main ()
{
   char sumber[50], tujuan[50];

   strcpy(sumber,  "Ini adalah sumber");
   strcpy(tujuan, "Ini adalah tujuan");

   strncat(tujuan, sumber, 15);

   printf("Tujuan akhir dari string : |%s|", tujuan);

   return 0;
}
```

Kode tersebut akan menghasilkan output di layar berupa

```
Tujuan akhir dari string : |Ini adalah tujuanIni adalah sumb|
```

13.4.8 strchr()

Perhatikan contoh kode program berikut

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

int main ()
{
   const char stringnya[] = "http://www.untan.ac.id";
   const char simbol = '.';
```

```
   char *hasil;

   hasil = strchr(stringnya, simbol);

   printf("String setelah simbol %c adalah  %s\n", simbol, hasil);

   return 0;
}
```

Kode tersebut akan menghasilkan output di layar berupa

```
String setelah simbol . adalah  .untan.ac.id
```

13.4.9 strcmp()

Perhatikan contoh kode program berikut

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

int main ()
{
   char pertama[15];
   char kedua[15];
   int hasil;


   strcpy(pertama, "abcdef");
   strcpy(kedua, "ABCDEF");

   hasil = strcmp(pertama, kedua);

   if(hasil < 0)
   {
      printf("pertama kurang dari yang kedua");
   }
   else if(hasil > 0)
   {
      printf("kedua kurang dari yang pertama");
   }
   else
   {
      printf("pertama sama dengan kedua");
   }

   return 0;
```

```
}
```

Kode tersebut akan menghasilkan output di layar berupa

```
kedua kurang dari yang pertama
```

13.4.10 strcoll()

Perhatikan contoh kode program berikut

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

int main ()
{
   char pertama[15];
   char kedua[15];
   int hasil;

   strcpy(pertama, "abc");
   strcpy(kedua, "ABC");

   hasil = strcoll(pertama, kedua);

   if(hasil > 0)
   {
      printf("pertama kurang dari kedua");
   }
   else if(hasil < 0)
   {
      printf("kedua kurang dari pertama");
   }
   else
   {
      printf("pertama sama dengan kedua");
   }

   return 0;
}
```

Kode tersebut akan menghasilkan output di layar berupa

```
pertama kurang dari kedua
```

13.4.11 strcpy()

Perhatikan contoh kode program berikut

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

```
#include <string.h>

int main()
{
   char sumber[40];
   char tujuan[100];

   memset(tujuan, '\0', sizeof(tujuan));
   strcpy(sumber, "Universitas Tanjungpura");
   strcpy(tujuan, sumber);

   printf("Hasil penyalinan string : %s\n", tujuan);

   return 0;
}
```

Kode tersebut akan menghasilkan output di layar berupa

```
Hasil penyalinan string : Universitas Tanjungpura
```

13.4.12 strncpy()

Perhatikan contoh kode program berikut

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

int main()
{
   char sumber[40];
   char tujuan[12];

   memset(tujuan, '\0', sizeof(tujuan));
   strcpy(sumber, "Universitas tanjungpura");
   strncpy(tujuan, sumber, 10);

   printf("Hasil akhir penyalinan string : %s\n", tujuan);

   return 0;
}
```

Kode tersebut akan menghasilkan output di layar berupa

```
Hasil akhir penyalinan string : Universita
```

13.4.13 strcspn()

Perhatikan contoh kode program berikut

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

int main ()
{
   int panjang;
   const char pertama[] = "ABCDEF4960910";
   const char kedua[] = "013";

   panjang = strcspn(pertama, kedua);

   printf("Karakter pertama yang cocok adalah pada posisi array ke
%d\n", panjang + 1);

   return 0;
}
```

Kode tersebut akan menghasilkan output di layar berupa

```
Karakter pertama yang cocok adalah pada posisi array ke 10
```

13.4.14 strerror()

Perhatikan contoh kode program berikut

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

int main ()
{
   FILE *fp;

   fp = fopen("berkas.txt","r");
   if( fp == NULL )
   {
      printf("Pesan Error: %s\n", strerror(errno));
   }

   return 0;
}
```

Jika pada directory yang sama dengan program tidak terdapat file berkas.txt, kode tersebut akan menghasilkan output di layar berupa

```
Pesan Error: No such file or directory
```

13.4.15 strlen()

Perhatikan contoh kode program berikut

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

int main ()
{
   char kata[50];
   int panjang;

   strcpy(kata, "Universitas Tanjungpura");

   panjang = strlen(kata);
   printf("Panjang dari string %s adalah %d\n", kata, panjang);

   return 0;
}
```

Kode tersebut akan menghasilkan output di layar berupa

```
Panjang dari string Universitas Tanjungpura adalah 23
```

13.4.16 strpbrk()

Perhatikan contoh kode program berikut

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

int main ()
{
   const char pertama[] = "abcde2fghi3jk4l";
   const char kedua[] = "34";
   char *hasil;

   hasil = strpbrk(pertama, kedua);
   if(hasil)
   {
      printf("Karakter pertama yang cocok: %c\n", *hasil);
   }
   else
   {
      printf("Katakter tidak ditemukan");
   }
```

```
   return 0;
}
```

Kode tersebut akan menghasilkan output di layar berupa

```
Karakter pertama yang cocok: 3
```

13.4.17 strrchr()

Perhatikan contoh kode program berikut

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

int main ()
{
   int panjang;
   const char kata[] = "http://www.untan.ac.id";
   const char simbol = '.';
   char *hasil;

   hasil = strrchr(kata, simbol);

   printf("String setelah simbol %c  adalah %s\n", simbol, hasil);

   return 0;
}
```

Kode tersebut akan menghasilkan output di layar berupa

```
String setelah simbol .  adalah .id
```

13.4.18 strspn()

Perhatikan contoh kode program berikut

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

int main ()
{
   int panjang;
   const char pertama[] = "ABCDEFG019874";
   const char kedua[] = "ABCD";

   panjang = strspn(pertama, kedua);

   printf("Panjang dari string yang cocok %d\n", panjang);
```

```
   return 0;
}
```

Kode tersebut akan menghasilkan output di layar berupa

```
Panjang dari string yang cocok 4
```

13.4.19 strstr()

Perhatikan contoh kode program berikut

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


int main()
{
   const char diuji[30] = "Universitas Tanjungpura";
   const char penguji[20] = "Tanjungpura";
   char *hasil;

   hasil = strstr(diuji, penguji);

   printf("Hasil subtringnya adalah: %s\n", hasil);

   return 0;
}
```

Kode tersebut akan menghasilkan output di layar berupa

```
Hasil subtringnya adalah: Tanjungpura
```

13.4.20 strtok()

Perhatikan contoh kode program berikut

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

int main(){

   char kalimat[80] = "Sistem Informasi - Universitas Tanjungpura";
   const char s[2] = "-";
   char *kunci;

   kunci = strtok(kalimat, s);

   while(kunci != NULL){
      printf( " %s\n", kunci );
```

```
      kunci = strtok(NULL, s);
   }

   return 0;
}
```

Kode tersebut akan menghasilkan output di layar berupa

```
 Sistem Informasi
  Universitas Tanjungpura
```

13.4.21 strxfrm()

Perhatikan contoh kode program berikut

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

int main()
{
   char tujuan[30];
   char sumber[30];
   int panjang;

   strcpy(sumber, "Universitas Tanjungpura");
   panjang = strxfrm(tujuan, sumber, 30);

   printf("Panjang dari string %s adalah %d", tujuan, panjang);

   return 0;
}
```

Kode tersebut akan menghasilkan output di layar berupa

```
Panjang dari string Universitas Tanjungpura adalah 23
```

DAFTAR PUSTAKA

Burges dan Evans.2002. *The GNU C Programming Tutorial*. Under GNU license

Clifford, S.A. 2013. *Data Structures and Algorithm Analysis*. Department of Computer Science Virginia Tech. Blacksburg

Kadir, Abdul. 2015. *From Zero to a Pro*. Penerbit Andi. Yogyakarta

Munir, Rinaldi. 2007. *Algoritma & Pemrograman dalam Bahasa Pascal dan C*. Penerbit Informatika. Bandung

Rahmat, Antonius. 2010. *Algoritma dan Pemrograman dengan Bahasa C*. Penerbit Andi. Yogyakarta

TutorialsPoint. (2016, 13 Desember). *C Programming*. Diperoleh 13 Desember 2015, dari https://www.tutorialspoint.com/cprogramming