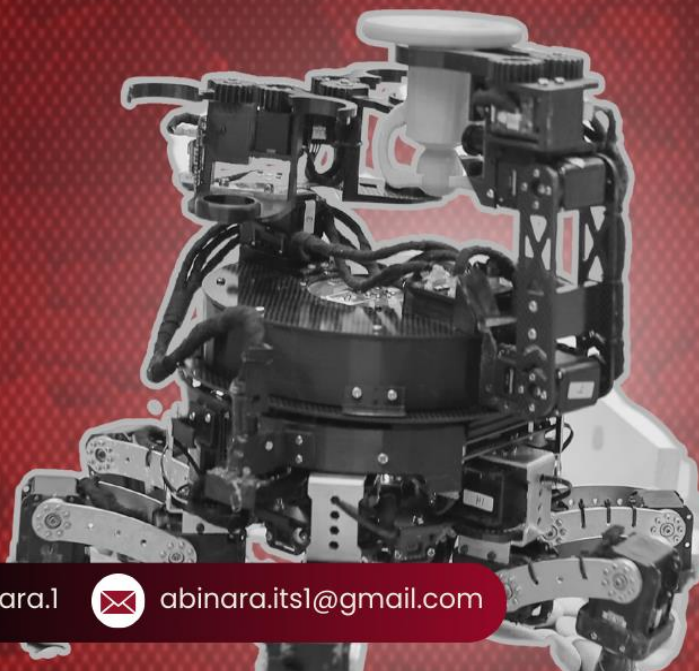
MODULE PROGRAMMING : BASIC C

Developing Next-Gen SAR Robots

ABINARA 1



#WaniJuara
#WaniJuara
#WaniJuara

abinara1its | Abinara-1 ITS | abinara.1 | abinara.its1@gmail.com

Pengenalan Bahasa C

❖ Program "Hello, world."

Mari kita mulai dengan membuat program sederhana, yakni program untuk menampilkan kalimat “Hello, world!” pada layar (konsol).

Buatlah program baru dengan format penamaan (nama_file).c, ex : hello.c
kemudian isi dengan kode berikut :

```c
#include <stdio.h>

int main()
{
    printf("Hello, world!\n");
    return 0;
}
```

Salinlah kode di atas pada IDE, kemudian compile dan jalankan. Proses compilation akan menghasilkan output sebagai berikut.

```
Hello, world!
```

❖ Struktur Dasar Program C

Setiap kita ingin membuat program C, kita harus menuliskan struktur seperti berikut ini :

```c
#include <stdio.h>

int main(){
    // kode atau logika program kita di sini
    return 0;
}
```

Ini adalah struktur dasar yang wajib dipahami.

Kalau kita perhatikan, struktur program tersebut dibagi menjadi dua bagian utama:

1. Bagian Include
2. Blok Fungsi Main

Mari kita bahas lebih detail :

❖ Header File

Baris 1 pada kode di atas disebut dengan preprocessor directive. Preprocessor yang digunakan dalam hal ini adalah `#include`.

Preprocessor `#include` menjelaskan bahwa program tersebut menyertakan file header `<stdio.h>`. File header tersebut merupakan library bawaan C yang berisi fungsi-fungsi penting yang dibutuhkan oleh program, misalnya pada kasus diawal modul, program membutuhkan fungsi `printf()` untuk mencetak kalimat "`Hello, world!`". Tanpa adanya library, program tersebut tidak akan bisa di-compile.

❖ Fungsi main()

Fungsi `main()` pada kode tersebut ditunjukan pada baris ke 3 hingga baris ke 7.

Apa itu Fungsi `main()`?
Fungsi `main()`adalah fungsi utama dalam program. Fungsi ini akan dieksekusi pertama kali saat program dijalankan.

Karena itu, kita harus menuliskan logika program di dalam fungsi ini.

❖ Whitespace

Dapat diperhatikan pada kode bahwa baris 2 dan 4 kosong (tidak terdapat karakter apapun). Ini disebut dengan whitespace. Whitespace adalah area kosong pada kode, dan biasanya dtujukan agar kode mudah dibaca.

Terdapat tiga jenis whitespace, yakni space, tab, dan new line. Baris 2 dan 4 adalah contoh dari new line.

❖ Penulisan Blok Kode

Blok kode adalah kumpulan statement atau ekspresi. Blok kode pada program C dibungkus dengan kurung kurawal { ... }

❖ Penulisan Statement

Statemen adalah perintah-perintah atau fungsi untuk melakukan sesuatu.
Contoh:

```
printf("Hello World!");
```

Artinya : kita menyuruh komputer untuk mencetak teks Hello World! ke console.

Perhatikan juga, setiap statement di dalam program C wajib diakhiri dengan titik koma (;), kalau tidak maka program akan error.

❖ Penulisan Komentar

Komentar adalah bagian yang tidak akan dieksekusi oleh komputer. Komentar bisanya digunakan untuk membuat keterangan pada kode program.

Contoh Komentar :

```
// ini adalah komentar
```

Kita bisa menggunakan garis miring ganda (//) untuk membuat komentar satu baris dan tanda /**/ untuk membuat komentar beberapa baris.

Contoh :

```
#include <stdio.h>

int main(){
    // ini adalah komentar satu baris
    printf("Hello World!");

    /*
        ini adalah komentar
        yang lebih
        dari satu baris
    */
    return 0;
}
```

Keyword dan Identifier

❖ Keyword

Keyword merupakan kata khusus yang telah dipesan (reserved) pada bahasa pemograman. Kata-kata khusus tersebut mempunyai makna tersendiri bagi compiler, dan kata-kata tersebut merupakan bagian dari sintaks dan tidak dapat digunakan sebagai identifier. Berikut adalah daftar keyword pada bahasa C.

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

❖ Identifier

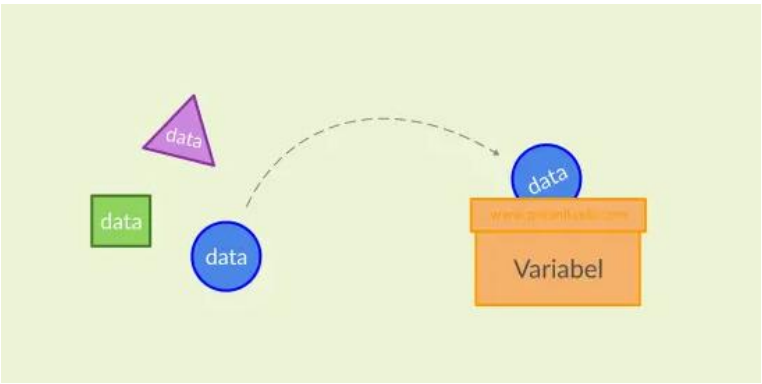
Identifier merujuk kepada penamaan sebuah entitas, seperti pada variabel, fungsi, structures dan lain-lain. Karena identifier menamai sebuah entitas, maka nama dari identifier harus unik (dua entitas tidak boleh mempunyai nama identifier yang sama).

Aturan penamaan identifier:

1. Identifier bukan merupakan keyword.
2. Identifier hanya boleh terdiri dari huruf lowercase, huruf uppercase, digit, dan simbol underscore (_).
3. Identifier tidak boleh mengandung whitespace.
4. Identifier harus dimulai dengan huruf atau simbol underscore. Tidak boleh dimulai dengan digit/angka.
5. Bersifat case-sensitive, artinya identifier variable berbeda dengan vAriaBle.

Apa itu Variabel dan Tipe Data?

Variabel adalah sebuah tempat menyimpan sebuah nilai. Sementara tipe data adalah jenis nilai yang akan tersimpan dalam variabel.



Variabel

❖ Aturan Penulisan Variabel pada C

Ada beberapa aturan penulisan variabel yang harus kamu ketahui :

1. Nama variabel tidak boleh didahului dengan simbol dan angka.
2. Nama variabel tidak boleh menggunakan kata kunci yang sudah ada pada bahasa C, contoh: if, int, void, dll.
3. Nama variabel bersifat case sensitive, artinya huruf besar dan kecil dibedakan, contoh: nama dan Nama adalah dua variabel yang berbeda.
4. Disarankan menggunakan underscore untuk nama variabel yang terdiri dari dua suku kata, contoh: nama_lengkap.

❖ Deklarasi dan Definisi Variabel

Pada bahasa C, variabel harus dideklarasikan terlebih dahulu sebelum bisa digunakan. Seperti halnya gelas tadi, gelas tersebut harus ada terlebih dahulu sebelum bisa digunakan.
Untuk mendeklarasikan sebuah variabel, sintaksnya adalah sebagai berikut :
`<tipe_data> <identifier>`

Misalkan, potongan kode berikut mendeklarasikan variabel x yang bertipe int.
`int x`
Jika ingin mendeklarasikan lebih dari satu variabel dengan tipe yang sama, bisa menggunakan operator koma (,).
`<tipe_data> <variabel1>, <variabel2>, ... dst;`

```
int x, y;
```

❖ Pengisian Nilai Pada Variabel

Setelah dideklarasikan, variabel dapat diisi oleh sebuah nilai. Untuk melakukannya, gunakan operator assignment `(simbol =).`

identifier_variabel = <nilai yang bersesuaian>

```
int x, y;
x = 10;
y = -2;
```

Dalam hal ini, variabel x dan y akan mempunyai nilai masing-masing 10 dan -2.

❖ Inisialisasi Variabel

Deklarasi dan pengisian nilai pada variabel dapat dilakukan dalam satu instruksi sekaligus. Hal ini disebut dengan inisialisasi. Dengan melakukan inisialisasi variabel, berarti kita memberikan nilai awal pada variabel tersebut.

```
tipe_data identifier_variabel = <nilai yang bersesuaian>;
```

```
int x = 10;
```

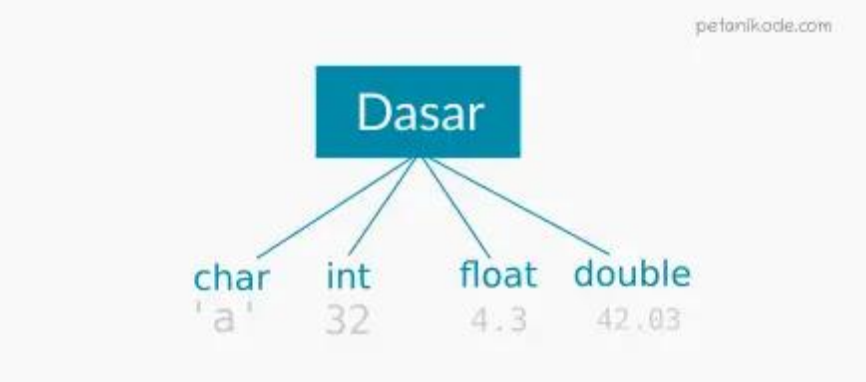
Contoh Program dengan Variabel

```
#include <stdio.h>

int main(){
    // membuat variabel dengan tipe data integer
    int berat;
    // membuat variabel dan langsung mengisinya
    int tinggi = 178;

    // mengisi nilai ke variabel
    berat = 54;

    return 0;
}
```

TIPE DATA

❖ Tipe Data Dasar pada C

Sesuai namanya, tipe data dasar adalah tipe data yang paling dasar dalam bahasa pemrograman C.



1. Char : adalah tipe data yang berisi 1 huruf atau 1 karakter;
2. Integer : adalah tipe data yang berupa angka;
3. Float : adalah tipe data yang berupa bilangan pecahan
4. Double : adalah tipe data sama seperti float, namun double memiliki ukuran penyimpanan yang lebih besar dibandingkan float.

Tipe data dasar diatas memiliki panjang dan ukuran, perhatikan tabel berikut.

Tipe Data	Ukuran	Panjang	Contoh
`char`	1 byte	-128 — 127 atau 0 — 255	`'A', 'a', '3'`
`int`	2 byte	-2,147,483,648 — 2,147,483,647	`32, 1, 4`
`float`	4 byte	1.2E-38 — 3.4E+38	`4.3, 2.2, 6.0`
`double`	8 byte	2.3E-308 — 1.7E+308	`4.2, 4.22, 3.2`

Format penulisan koma untuk tipe data float dan double menggunakan tanda titik, mengikuti format internasional.

Jika kamu menulis seperti ini 2,5 maka akan salah. Penulisan yang benar adalah 2.5.

Lalu untuk penulisan tipe data char harus diapit dengan tanda petik.

Mengapa?
Karena kalau misalnya karakter itu adalah spasi tanpa tanda petik kita tidak akan bisa mengisinya dengan karakter spasi.

Tipe Bilangan Bulat (integer)

Tipe Data	Memori (Byte)	Jangkauan Nilai			Format Specifier
		Min		Max	
short	2	-2^{15}	s.d	$2^{15} - 1$	%hi
unsigned short	2 - 4	0	s.d	$2^{16} - 1$	%hu
int	4	-2^{31}	s.d	$2^{31} - 1$	%d
unsigned int	4	0	s.d	$2^{32} - 1$	%u
long	4	-2^{31}	s.d	$2^{31} - 1$	%ld
unsigned long	4	0	s.d	$2^{32} - 1$	%lu
long long	8	-2^{63}	s.d	$2^{63} - 1$	%lld
unisgned long long	8	0	s.d	$2^{64} - 1$	%llu

Seperti namanya, tipe-tipe data di atas adalah tipe data yang digunakan untuk merepresentasikan bilangan bulat (positif dan negatif) dan bilangan 0. Misalnya, 0, -5, 12, -1, 200 dsb.

Tipe Bilangan Real (floating)

Tipe Data	Memori (Byte)	Jangkauan Nilai	Format Specifier
float	4	1.2E-38 — 3.4E+38	%f
double	8	2.3E-308 — 1.7E+308	%lf

Tipe data di atas digunakan untuk menyimpan data berupa bilangan real (floating-point)/bilangan berkoma. Misalnya, 2.35, -12.246, 0.005 dsb.

Tipe Karakter

Tipe Data	Memori (Byte)	Jangkauan Nilai	Format Specifier
char	1	-2^7 s.d $2^7 - 1$	%c
unsigned char	1	0 s.d $2^8 - 1$	%c

Penggunaan paling umum dari tipe data di atas adalah untuk merepresentasikan satu karakter. Misalnya, ‘A’, ‘-‘, dan sebagainya.

Contoh Program :

```c
#include <stdio.h>

int main(){
    int usia = 19;
    float berat = 65.3;
    double tinggi = 178.43;
    char jenis_kelamin = 'L';
}
```

Input dan Output Dasar

❖ Output Dasar

Fungsi printf()
merupakan fungsi untuk menampilkan output ke layar komputer. Fungsi ini terdapat pada library stdio.h.
Oleh sebab itu, ketika kita diharuskan untuk menuliskan #include <stdio.h> di bagian atas program agar bisa menggunakan fungsi ini.

Berikut ini struktur dasar fungsi printf() :

"format" adalah sebuah teks (string) untuk ditampilkan. Lalu tanda ... akan berisi sebuah variabel atau nilai untuk ditampilkan berdasarkan format yang diberikan pada teks "format".

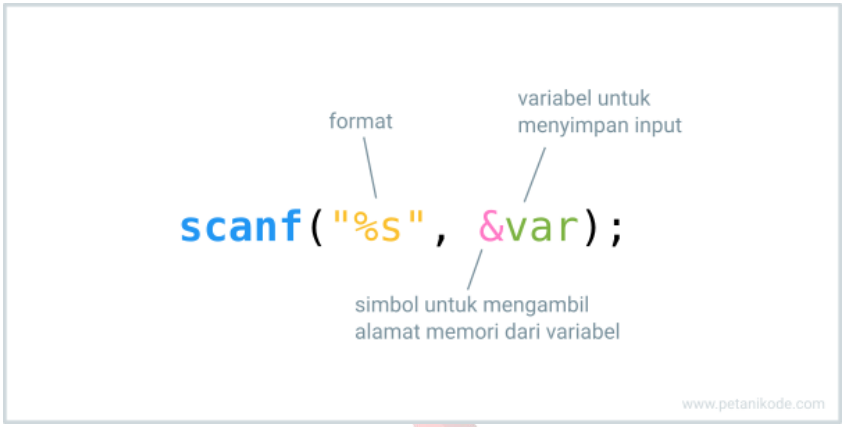
Mari kita lihat contohnya :

```c
#include <stdio.h>

int main(){
    int usia = 19;
    float berat = 65.3;
    double tinggi = 178.43;
    char jenis_kelamin = 'L';

    printf("Usia: %d tahun\n", usia);
    printf("Tinggi: %.2f cm\n", tinggi);
    printf("Berat: %.2f Kg\n", berat);
    printf("Jenis kelamin: %c\n", jenis_kelamin);

    return 0;
}
```

\n adalah simbol untuk membuat baris baru (new line)

❖ Input Dasar

Fungsi scanf()
adalah fungsi untuk mengambil input dari keyboard. Fungsi ini memiliki format seperti fungsi printf().



Contoh :
Program di bawah menerima input berupa bilangan bulat yang disimpan pada variabel n, kemudian mencetak nilai variabel n dengan format
“n mempunyai nilai = n”.

```c
#include <stdio.h>

int main()
{
    int n;
    scanf("%d", &n);
    printf("n mempunyai nilai = %d", n);
    return 0;
}
```

Input :
3

Output :

```
n mempunyai nilai = 3
```

Operator

Operator adalah sebuah simbol yang digunakan untuk melakukan operasi tertentu.

Dilihat dari jumlah operannya, operator dibagi menjadi tiga jenis, yaitu :

- Unary – yakni operator yang bekerja pada satu operan, misalnya -5.
- Binary – yakni operator yang bekerja pada dua operan, misalnya 2 + 3.
- Ternary – yakni operator yang bekerja pada tiga operan. (Akan dibahas pada bagian selanjutnya).

Dilihat dari kegunaannya, berikut adalah jenis-jenis operator pada bahasa C.

❖ Operator Assignment

Operator Assignment digunakan untuk mengisikan (assign) sebuah nilai ke variabel. Simbol yang biasa digunakan adalah tanda sama dengan (=). Contohnya :

```
int x, y;
x = 4;
y = 3;
x = x + y; // x = 7
y = x + x; // y = 14
```

❖ Operator Aritmatika

Seperti namanya, operator aritmatika melakukan operasi layaknya pada matematika seperti penjumlahan, pengurangan, pembagian dsb. Beberapa operator menggunakan simbol yang sama pada matematika (penjumlahan dengan simbol '+', pengurangan dengan '-', dst.). Operator-operator aritmatika pada bahasa C adalah sebagai berikut.

Simbol	Operasi	Contoh
+	Penjumlahan pada dua operan	a + b
-	Pengurangan pada dua operan	a - b
*	Perkalian pada dua operan	a * b
/	Pembagian pada dua operan	a / b
%	Menghitung sisa pembagian dua operan (operasi modulo)	a % b

❖ Operator Increment dan Decrement

Operator `++` disebut dengan operator increment, sedangkan operator `--` merupakan operator decrement. Kedua operator ini digunakan untuk menambah (increment)/mengurangi (decrement) nilai dari suatu variabel sebanyak satu.

Terdapat dua cara untuk menggunakan operator ini.

- **Prefix** - yakni dengan meletakkan operator increment/decrement didepan nama variabel.

Cara kerja dari operator increment/decrement prefix adalah dengan menambahkan/mengurangi nilai variabel sebanyak satu terlebih dahulu, sebelum operan tersebut digunakan pada operasi lainnya pada sekuens intstruksi yang sama. Untuk lebih jelasnya, perhatikan potongan kode berikut

```
int a, b;
a = 5;
b = ++a; // Nilai b sekarang adalah 6
a = --b; // Nilai a sekarang adalah 5
```

Di sini, saat instruksi `b = ++a;` dieksekusi, yang terjadi pertama kali adalah nilai dari **a** ditambahkan satu terlebih dahulu, kemudian baru di-assign nilainya ke variabel **b**.

- **Postfix** - yakni dengan meletakkan operator increment/decrement di belakang nama variabel.

Cara kerja dari operator increment/decrement postfix berbeda dari prefix. Pada postfix, nilai variabel akan ditambah satu setelah operan digunakan pada operasi lainnya pada sekuens instruksi yang sama.

```
int a, b;
a = 5;
b = a++; // Nilai b sekarang adalah 5
a = b--; // Nilai a sekarang adalah 5
```

Di sini, saat instruksi `b = a++;` dieksekusi, yang terjadi pertama kali adalah nilai dari **a** akan di-assign terlebih dahulu ke variabel **b**, kemudian baru ditambahkan satu. Karena itulah variabel b mendapat nilai dari a sebelum terjadi penambahan.

Untuk lebih jelasnya, perhatikan dan cermati code pada halaman berikutnya.

```c
#include <stdio.h>

int main() {
    int a = 5;
    int b = 5;
    int c;
    int d;

    // Prefix Increment
    // Pertama, nilai a ditambahkan, kemudian nilai baru dari a digunakan
    c = ++a;
    printf("Prefix Increment: a = %d, c = %d\n", a, c); // a = 6, c = 6

    // Postfix Increment
    // Pertama, nilai b digunakan dalam ekspresi, kemudian b ditambahkan
    d = b++;
    printf("Postfix Increment: b = %d, d = %d\n", b, d); // b = 6, d = 5

    // Resetting values for demonstration
    a = 5;
    b = 5;

    // Prefix Decrement
    // Pertama, nilai a dikurangi, kemudian nilai baru dari a digunakan
    c = --a;
    printf("Prefix Decrement: a = %d, c = %d\n", a, c); // a = 4, c = 4

    // Postfix Decrement
    // Pertama, nilai b digunakan dalam ekspresi, kemudian b dikurangi
    d = b--;
    printf("Postfix Decrement: b = %d, d = %d\n", b, d); // b = 4, d = 5

    return 0;
}
```

Hasil Output

Menjalankan kode di atas akan menghasilkan output berikut :

```
Prefix Increment: a = 6, c = 6
Postfix Increment: b = 6, d = 5
Prefix Decrement: a = 4, c = 4
Postfix Decrement: b = 4, d = 5
```

❖ Operator Relasional

Operator Relasional digunakan untuk memeriksa relasi dan membandingkan nilai dari dua operan. Jika benar akan menghasilkan nilai TRUE (direpresentasikan angka 1), jika salah maka akan menghasilkan nilai FALSE
(direpresentasikan angka 0).

Operator	Simbol	Keterangan	Contoh
Sama dengan	==	Digunakan untuk memeriksa apakah kedua operan memiliki nilai yang sama.	5 == 2 (FALSE) 5 == 5 (TRUE)
Tidak Sama dengan	!=	Digunakan untuk memeriksa apakah kedua operan memiliki nilai yang tidak sama.	5 != 2 (TRUE) 5 != 5 (FALSE)
Lebih besar	>	Digunakan untuk membandingkan apakah operan pertama lebih besar nilainya dari operan kedua.	5 > 2 (TRUE) 5 > 5 (FALSE) 2 > 4 (FALSE)
Lebih kecil	<	Digunakan untuk membandingkan apakah operan pertama lebih kecil nilainya dari operan kedua.	5 < 2 (FALSE) 5 < 5 (FALSE) 2 < 4 (TRUE)
Lebih besar sama dengan	>=	Digunakan untuk membandingkan apakah operan pertama lebih besar atau sama nilainya dari operan kedua.	5 >= 2 (TRUE) 5 >= 5 (TRUE) 2 >= 4 (FALSE)
Lebih kecil sama dengan	<=	Digunakan untuk membandingkan apakah operan pertama lebih kecil atau sama nilainya dari operan kedua.	5 <= 2 (FALSE) 5 <= 5 (TRUE) 2 <= 4 (TRUE)

❖ Operator Logika

Operator Logika digunakan untuk melakukan tes pada kondisi/ekspresi, apakah kondisi tersebut benar atau salah. Operator logika hanya akan menghasilkan nilai TRUE (jika benar) atau FALSE (jika salah). TRUE direpresentasikan oleh angka 1, sedangkan FALSE oleh angka 0.

Operator-operator logika dalam bahasa C adalah sebagai berikut.

Operator	Simbol	Keterangan	Nilai Kebenaran
Logical NOT	!	Operator NOT digunakan untuk membalikkan kondisi, TRUE menjadi FALSE dan FALSE menjadi TRUE.	!1 = 0 !0 = 1
Logical AND	&&	Operator AND akan menghasilkan nilai TRUE jika kedua operan mempunyai nilai TRUE.	1 && 1 = 1 0 && 1 = 0 1 && 0 = 0 0 && 0 = 0
Logical OR	||	Operator OR akan menghasilkan nilai TRUE jika salah satu operan mempunyai nilai TRUE.	1 || 1 = 1 0 || 1 = 1 1 || 0 = 1 0 || 0 = 0

Operator logika pada umumnya digunakan bersamaan dengan operator relasional untuk melakukan tes pada ekspresi yang berhubungan dengan kebenaran suatu kondisi. Penggunaan paling umum adalah untuk melakukan percabangan (akan dipelajari di bagian selanjutnya).

Contoh :

```
int a, b, c, d;
a = 11;
b = 24;
c = 11;
d = ((a == c) && (b > a));                // 1 (TRUE)
d = ((a >= b) || (a < c));                // 0 (FALSE)
d = ((b != b) || (b > c)) && (c == a);    // 1 (TRUE)
```

❖ Operator Bitwise

Operator Bitwise, seperti namanya digunakan untuk melakukan operasi pada dua operan dalam skala biner (bilangan basis 2). Sebelum mempelajari lebih lanjut cara kerja operasi bitwise, sebaiknya kamu harus paham terlebih dahulu mengenai bilangan dalam basis biner.

Terdapat 6 jenis operator bitwise, yakni AND, OR, XOR, COMPELEMENT, SHIFT LEFT, dan SHIFT RIGHT. Untuk lebih memahami perbedaan cara kerja operator bitwise, perhatikan tabel berikut.

Operator	Simbol	Keterangan
Bitwise AND	&	Mengevaluasi bit dari dua operan. Menghasilkan 1 apabila keduanya 1, jika tidak menghasilkan nilai 0.
Bitwise OR	|	Mengevaluasi bit dari dua operan. Menghasilkan 1 apabila salah satu nilainya 1, jika keduanya 0, maka menghasilkan nilai 0.
Bitwise XOR	^	Mengevaluasi bit dari dua operan. Menghasilkan 1 apabila bit pada kedua operan nilainya berbeda. Jika sama, maka menghasilkan nilai 0.
Bitwise COMPLEMENT	~	Membalik semua nilai bit, dari 1 menjadi 0 dan 0 menjadi 1 (dalam panjang bit).
Bitwise SHIFT LEFT	<<	Menggeser bit ke kiri sebanyak n (operan kedua).
Bitwise SHIFT RIGHT	>>	Menggeser bit ke kanan sebanyak n (operan kedua).

Agar lebih memahaminya, perhatikan penggunaan Operator Bitwise pada contoh dihalaman berikutnya.

Contoh penggunaan operator bitwise:

Misal 12 dan 5. Representasi 12 dan 5 dalam basis biner adalah 12 = (1100) dan 5 = (0101). Maka, operasi bitwise adalah sebagai berikut.

Bitwise AND

```
12 = (1100)
 5 = (0101)
------------ &
 4 = (0100)
```

Bitwise OR

```
12 = (1100)
 5 = (0101)
------------ |
13 = (1101)
```

Bitwise XOR

```
12 = (1100)
 5 = (0101)
------------ ^
 9 = (1001)
```

Bitwise COMPLEMENT

```
 12 = (1100)
~12 = (0011)
```

Bitwise SHIFT LEFT

Misal kita hendak menggeser bit bilangan 13 ke kiri sebanyak 2, maka 13 << 2.

```
     13 = (001101)
13 << 2 = (110100)
```

Bisa diperhatikan, bit paling kanan setelah digeser akan diisi oleh 0. Maka hasil dari 13 << 2 = 52.

Bitwise SHIFT RIGHT

Misal kita hendak menggeser bit bilangan 13 ke kanan sebanyak 2, maka 13 >> 2.

```
     13 = (001101)
13 >> 2 = (000011)
```

Bisa diperhatikan, bit paling kiri setelah digeser akan diisi oleh 0. Maka hasil dari 13 >> 2 = 3.

❖ Operator Gabungan

Operator Gabungan adalah operator yang terdiri dari gabungan dua operator. Tujuan dari operator gabungan adalah untuk mempersingkat penulisan kode. Berikut adalah operator gabungan dalam bahasa C.

Operator	Contoh	Ekuivalen Dengan
+=	a += b	a = a + b
-=	a -= b	a = a - b
*=	a *= b	a = a * b
/=	a /= b	a = a / b
%=	a %= b	a = a % b
&=	a &= b	a = a & b
|=	a |= b	a = a | b
^=	a ^= b	a = a ^ b
>>=	a >>= b	a = a >> b
<<=	a <<= b	a = a << b

❖ Operator Lain

Selain operator-operator yang telah dijelaskan sebelumnya, terdapat beberapa operator lain yang terdapat pada bahasa C. Berikut penjelasannya.

Nama Operator	Simbol	Keterangan
Alamat memori	&	untuk mengambil alamat memori
Pointer	*	untuk membuat pointer
Ternary	? :	untuk membuat kondisi
Increment	++	untuk menambah 1
Decrement	--	untuk mengurangi 1

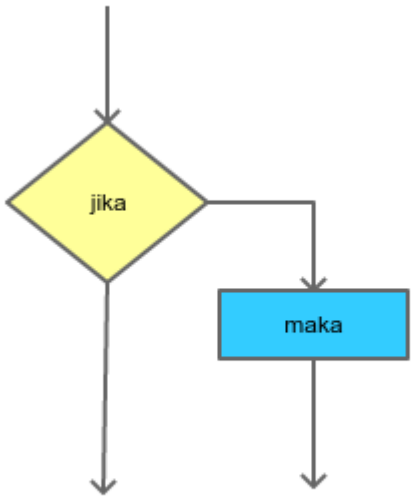
Percabangan

Apa itu percabangan?

Buat yang belum pernah kuliah atau belajar tentang algoritma dan flowchart, mungkin ini istilah yang baru pertama kamu dengar.

Istilah ini sebenarnya untuk menggambarkan alur program yang bercabang.

Pada flow chart, logika “jika.. maka” digambarkan dalam bentuk cabang.



Karena itu, ini disebut percabangan.

Percabangan akan mampu membuat program berpikir dan menentukan tindakan sesuai dengan logika/kondisi yang kita berikan.

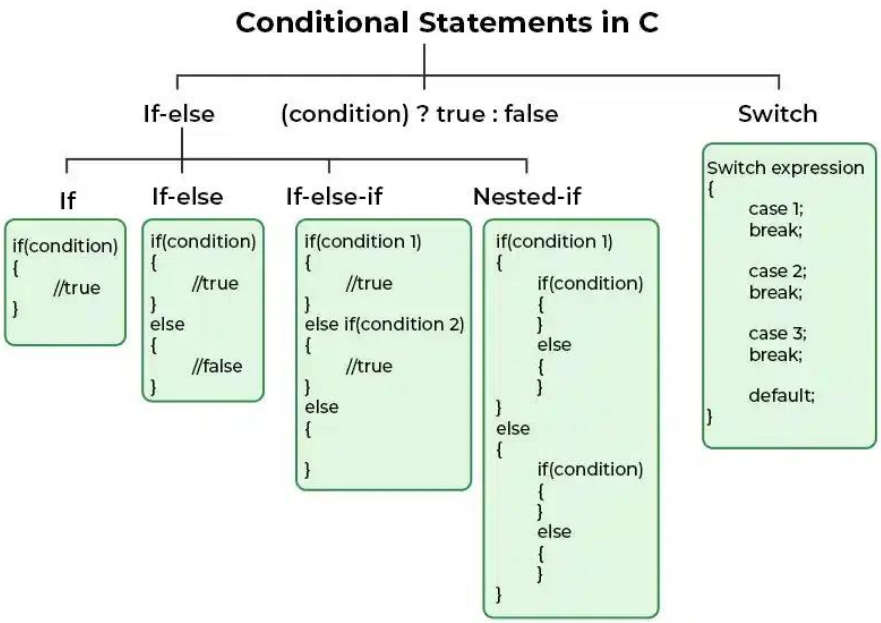
Bisa juga di artikan sebagai alur program yang bercabang untuk menjalankan suatu instruksi berdasarkan logika/kondisi yang ditetapkan.

Control Flow

Control Flow adalah cara kita mengatur jalan penyataan, instruksi, dan pemanggilan fungsi suatu program. Tanpa control flow, program kita hanya bergerak dari atas ke bawah saja (sequential). Control flow bahasa C ada 2, yaitu percabangan (selection) dan perulangan (repetition). Di Modul ini, kita akan memfokuskan pembahasan di percabangan terlebih dahulu.

Percabangan

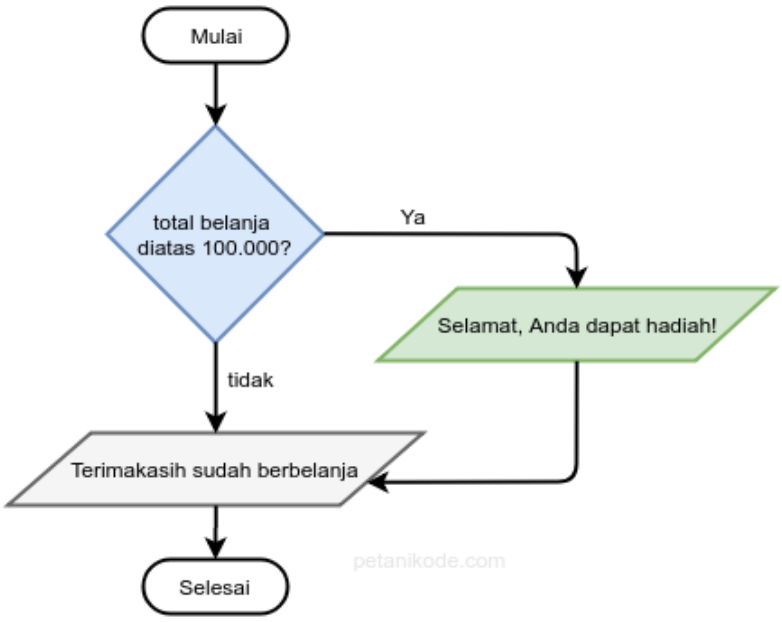
❖ Macam Percabangan Bahasa C



1. Percabangan IF
2. Percabangan if/else
3. Percabangan if/else/if
4. Percabangan Switch/Case
5. Percabangan dengan Operator Ternary
6. Percabangan Bersarang (Nested)

❖ Percabangan If

Percabangan if merupakan percabangan yang paling sederhana karena hanya memiliki satu blok pilihan saat kondisi bernilai benar.



Sintaks yang digunakan dalam percabangan menggunakan if adalah sebagai berikut.

```
if (<Ekspresi/Kondisi>) {

    // Kode yang akan dieksekusi jika kondisi tersebut benar (TRUE)

}
```

Cara kerja percabangan if yaitu memeriksa dan mengevaluasi suatu kondisi untuk menentukan apakah instruksi selanjutnya dalam bracket akan dijalankan atau tidak oleh program.

Jika kondisi tersebut benar/bernilai TRUE (1), kode yang di dalam bracket akan dieksekusi. Sebaliknya jika kondisi tersebut salah/bernilai FALSE (0), kode yang di dalam bracket tidak akan dieksekusi.

Contoh :

Sebagai contoh, di dashboard mobil terdapat indikator bahan bakar yang akan menyala jika bahan bakar yang tersisa kurang dari level tertentu (misal kurang dari 10 liter) dengan kondisi "Apakah bahan bakar kurang dari 10 liter?".

Pada kasus ini terdapat kondisi

- Jika bahan bakar kurang dari 10 liter maka nyalakan lampu indikator.

```
#include <stdio.h>

int main()
{
    int gasoline = 0;
    gasoline = 3;
    if (gasoline < 10) //jika bahan bakar kurang dari 10 liter
    {
        // Apa yang perlu dilakukan ketika kondisi terpenuhi?
        printf("Lampu Indikator menyala!\n");
    }
}
```

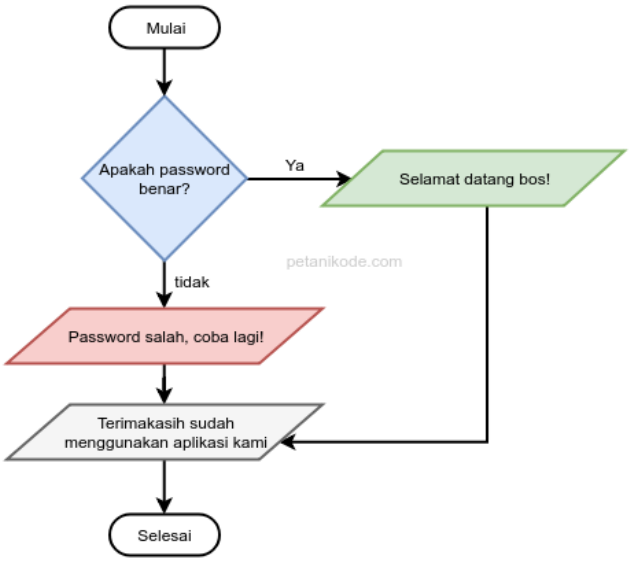
Output :

```
Lampu Indikator menyala!
```

❖ Percabangan If-Else

Percabangan if/else merupakan percabangan yang memiliki dua blok pilihan.

Blok pilihan pertama untuk kondisi benar, dan pilihan kedua untuk kondisi salah (else).



Sintaks yang digunakan dalam percabangan menggunakan if-else adalah sebagai berikut.

```
if (<Ekspresi/Kondisi>) {

    // Kode yang akan dieksekusi jika kondisi tersebut benar

}
else {

    // Kode yang akan dieksekusi jika kondisi tersebut salah

}
```

Cara kerja percabangan if-else yaitu memeriksa kondisi dalam if.

Jika kondisi tersebut bernilai TRUE (1), Program akan menjalankan kode di dalam bracket if. Sebaliknya jika kondisi tersebut bernilai FALSE (0), kode di bawah else lah yang akan dijalankan.
Contoh :
Pada kasus ini terdapat contoh teman, ketika kuotanya kurang dari 100 MB, akan meminta hotspot dari orang sekitarnya, namun Ketika kuota diatas 100 MB. Teman tersebut akan konfirmasi bahwa “Kuota Masih Ada”.

Dalam kode ini, kondisi didefinisikan kuota = 200;

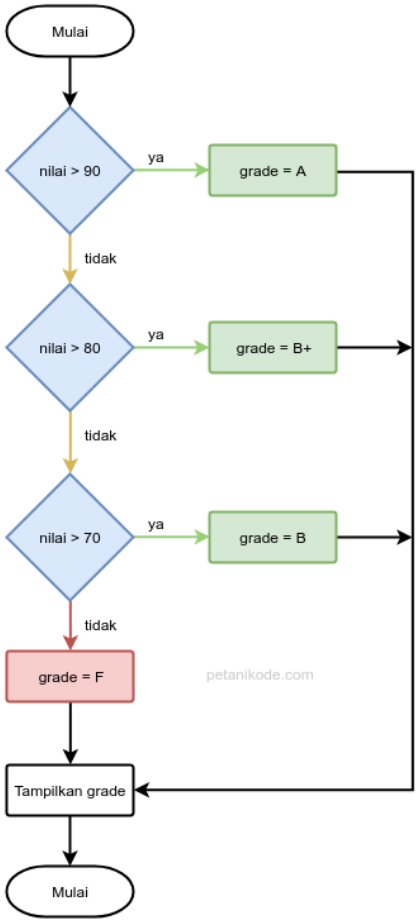
```c
#include <stdio.h>

int main()
{
    int kuota = 200;

    if (kuota > 100) { // Jika kuota lebih dari 100
        printf("Kuota Masih Ada");
    }
    else {
        printf("Hotspot Donk!");
    }
    return 0;
}
```

```
Kuota Masih Ada
```

❖ Percabangan if-else if

Percabangan if/else/if merupakan percabangan yang memiliki lebih dari dua blok pilihan.

Coba perhatikan flowchart berikut:



Sintaks yang digunakan dalam percabangan menggunakan if-else if adalah sebagai berikut.


```
if (<Ekspresi/Kondisi>) {

    // Kode yang akan dieksekusi jika kondisi tersebut benar

}
else if (<Ekspresi/Kondisi>) {

    // Kode yang akan dieksekusi jika kondisi tersebut salah

}
// Boleh menambahkan else{} apabila perlu
```

Cara kerja percabangan if-else yaitu memeriksa kondisi dalam if :

- Jika kondisi tersebut benar/bernilai TRUE (1), Program akan menjalankan kode di dalam bracket if.
- Apabila kondisi pertama tidak memenuhi, maka ia akan memerika kondisi didalam else-if, apabila benar/bernilai TRUE (1), maka ia akan menjalankan perintah dalam bracket tersebut, apabila tidak maka ia akan menjalankan sequence selanjutnya.
- Apabila kita menyediakan statement else diakhir, maka ketika seluruh kondisi if dan else-if salah/tidak memenuhi atau FALSE (0), maka secara otomatis ia akan menjalankan perintah di dalam else tersebut.

Contoh Program :
Program klasifikasi umur seseorang dibawah ini, menggunakan percabangan If-else If , yang didefinisikan umur pengguna 25 tahun :

```c
#include <stdio.h>

int main() {
    int age = 25; // Inisialisasi variabel age dengan nilai 25

    // Menentukan kategori usia berdasarkan nilai age
    if (age < 13) {
        printf("Kamu Anak-Anak\n");
    } else if (age >= 13 && age < 20) {
        printf("Kamu Remaja\n");
    } else if (age >= 20 && age < 65) {
        printf("Kamu sudah Dewasa\n");
    } else {
        printf("Kamu Sepuh\n");
    }

    return 0; // Mengembalikan nilai 0 untuk menandakan program berakhir dengan sukses
}
```

Output :

```
Kamu sudah Dewasa
```

❖ Percabangan Switch-Case

Selain penggunaan statement if untuk memilih diantara banyak alternatif, terdapat pula statement switch yang memiliki fungsi yang sama, untuk memilih diantara banyak alternatif berdasarkan sebuah kondisi. Kondisi pada statemen switch berisi ekspresi yang dapat menggunakan sebuah variable tunggal bertipe int atau char yang akan diperiksa nilainya di setiap blok case.

Sintaks untuk Switch-Case :

```
switch(variabel){
    case <value>:
        // blok kode
        break;
    case <value>:
        // blok kode
        break;
    /* Anda bisa memiliki jumlah case sebanyak mungkin */

    /* blok kode Switch-Case diakhiri dengan "default", yaitu bagian
      kode yang akan dieksekusi jika tidak ada case yang memenuhi */

    default:
        // blok kode
}
```

Pada <value>, kita bisa isi dengan nilai yang nanti akan dibandingkan dengan variabel.

Setiap case harus diakhiri dengan break. karena apabila tidak maka ia akan tetap menjalankan blok case di bawahnya hingga bertemu break lain atau pada akhir blok switch. Tapi khusus untuk default, tidak perlu diakhiri dengan break karena dia terletak di bagian akhir dan memang fungsinya untuk mengakhiri switch.

Agar lebih bisa memahami percabangan Switch-Case bisa lihat contoh program pada halaman selanjutnya.

```c
#include <stdio.h>

int main()
{
    char platNomor;
    printf("Masukkan huruf awal plat nomor anda: ");
    scanf("%c", &platNomor);

    switch(platNomor)
    {
        case 'L':
            printf("Surabaya");
            break;

        case 'B':
            printf("Jakarta");
            break;

        case 'D':
            printf("Bandung");
            break;

        case 'N':
            printf("Malang");
            break;

        default:
            printf("Karakter plat nomor tidak diketahui");
    }
    return 0;
}
```

Dalam contoh di atas, ekspresi yang digunakan adalah **PlatNomor**, di mana case-case nya adalah, huruf plat nomor tersebut, L, B, D, N, dan sebagainya.

❖ Percabangan dengan Operator Ternary

Operator kondisional adalah satu-satunya operator ternary dalam bahasa C. Dan bisa dibilang : Bentuk singkatnya dari if/else.

Operator ini mempunyai bentuk sintaks sebagai berikut :

```
(kondisi) ? (true) : (false)
```

Ekspresi/kondisi yang akan dievaluasi diletakkan sebelum tanda tanya (?). Apabila menghasilkan TRUE, maka program akan mengeksekusi bagian di kiri tanda titik dua. Jika FALSE, akan mengeksekusi bagian di kanan tanda titik dua.

Contoh Program :

```c
#include <stdio.h>

int main()
{
    int mark;

    scanf("%d", &mark);
    printf(mark >= 75 ? "Lulus\n\n" : "Tidak Lulus\n");
    return 0;
}
```

Program di atas ekuivalen dengan :

```c
#include <stdio.h>

int main()
{
    int mark;

    scanf("%d", &mark);
    if (mark >= 75) {
        printf("Lulus\n\n");
    } else {
        printf("Tidak Lulus\n");
    }
    return 0;
}
```

❖ Percabangan Bersarang (Nested)

Semua bentuk blok percabangan di atas dapat kita buat di dalam percabangan yang lainnya. Ini disebut dengan percabangan bersarang atau nested if.
Percabangan bersarang adalah bentuk blok percabangan yang di dalamnya juga terdapat percabangan lain (nested).

Contoh program :

```c
#include <stdio.h>

int main() {
    int angka;

    scanf("%d", &angka);    // Meminta input angka dari user

    // Memeriksa apakah angka positif, negatif, atau nol
    if (angka > 0) {
        printf("Angka Positif\n");

        // Percabangan bersarang untuk memeriksa apakah ganjil atau genap
        if (angka % 2 == 0) {
            printf("Genap.\n");
        } else {
            printf("Ganjil.\n");
        }
    }
    else if (angka < 0) {
        printf("Angka Negatif\n");

        // Percabangan bersarang untuk memeriksa apakah ganjil atau genap
        if (angka % 2 == 0) {
            printf("Genap.\n");
        } else {
            printf("Ganjil.\n");
        }
    }
    else {
        printf("Angka NOL.\n");
    }

    return 0;
}
```

Perulangan

Perulangan atau looping memungkinkan kita untuk mengeksekusi potongan kode berulang-ulang hingga mencapai suatu kondisi. Ada 3 jenis perulangan dalam bahasa C, yaitu for, while, dan do – while.

Dalam merancang perulangan, kita setidaknya harus mengetahui 3 komponen :

1. Kondisi awal perulangan.
2. Kondisi pada saat perulangan.
3. Kondisi yang harus dipenuhi agar perulangan berhenti.

❖ Perulangan For

Berikut format dasar struktur perulangan for dalam bahasa C :

```
for (inialisasi_statement; kondisi/ekspresi; update_statement) {
    //  Potongan kode yang dieksekusi
    .
    .
}
```

Bagian inialisasi_statement digunakan untuk inisialisasi variabel yang akan digunakan dalam perulangan. Bagian ini hanya dijalankan sekali saja pada saat awal perulangan.

Selanjutnya kondisi/ekspresi akan dievaluasi. Jika menghasilkan TRUE, maka akan mengeksekusi potongan kode. Jika menghasilkan FALSE, maka perulangan berhenti.

Setelah potongan kode selesai dieksekusi, akan mengeksekusi bagian update_statement. Biasanya bagian ini digunakan sebagai increment/decrement.

Lalu akan mengevaluasi ekspresi/kondisi lagi, dan begitu seterusnya.

Agar lebih mudah dalam memahami perulangan For, perhatikan dengan seksama contoh code program perulangan For pada halaman berikutnya.

```c
#include <stdio.h>
int main()
{
    int i;
    //    init; condition; increment
    for (i = 0; i < 10  ; i++) {
        printf("Hello World!\n");
    }
}
```

1. Awalnya i bernilai 0
2. For statement akan memeriksa nilai i apakah kurang dari 10
3. Apabila TRUE maka jalankan kode dalam for block, yakni print hello world
4. Setelah command dalam block for selesai dijalankan, maka variabel i akan di-increment, dan diperiksa lagi.
5. Apabila i kurang dari 10, maka command dalam block dieksekusi, apabila tidak maka for loop akan berhenti

Output :

```
Hello World!
Hello World!
Hello World!
Hello World!
Hello World!
Hello World!
Hello World!
Hello World!
Hello World!
Hello World!
```

Apakah nama variabelnya harus selalu `i` ?
Tidak.

Kita juga bisa menggunakan nama lain.

❖ Perulangan While

Dalam perulangan For, ketiga syarat (kondisi awal perulangan, kondisi pada saat perulangan, dan kondisi akhir perulangan.) ditulis dalam 1 baris perintah,
seperti : for (i = 1; i < 5; i++). Di dalam perulangan While, ketiga kondisi ini saling terpisah.

```
//initial value misal, i = 0
while (<Ekspresi/Kondisi>) {
    // Potongan kode yang ingin dieksekusi
    .
    .
    .
    // increment/decrement misalnya, i++
}
```

Cara kerja perulangan while mirip dengan if. Jika pada if potongan kode akan dieksekusi sekali saja apabila ekspresi/kondisi bernilai TRUE, pada while potongan kode akan terus dieksekusi hingga ekspresi/kondisi menghasilkan FALSE.

Contoh :

```
#include <stdio.h>

int main()
{
    int i = 0;
    while (i < 10)
    {
        printf("Hello World! ke-%d \n",i);
        i++;
    }
    return 0;
}
```

Sehingga pada contoh di atas:

- Pada awalnya, variabel i bernilai 0.
- Sequence selanjutnya adalah while, dan i bernilai kurang dari 10 (TRUE), maka kode didalam while akan dijalankan, yakni print Hello world ke-i.
- Setelah melakukan print hello world, variabel i akan di increment, dan kembali ke statement while untuk memeriksa apakah i masih kurang dari 10 setelah di-increment
- Karena setelah i di-increment nilainya masih 1 dan kurang dari 10, maka while akan dijalankan lagi hingga i bernilai 10 yang berarti tidak memenuhi kondisi while.

❖ Perulangan Do-While

Sintaks dari perulangan do–while adalah sebagai berikut.

```
do {
    // Potongan kode yang dieksekusi.
    .
    // increment/decrement
} while (<Ekspresi/Kondisi>)
```

Cara kerja dari perulangan do–while mirip dengan perulangan while. Hanya saja, pada perulangan do–while, potongan kode dijamin akan dieksekusi tepat satu kali sebelum mengevaluasi ekspresi/kondisi.

Jadi, Perulangan do–while akan melakukan perulangan sebanyak 1 kali terlebih dahulu, lalu mengecek kondisi yang ada di dalam kurung.

Jadi perbedaannya :
“ Perulangan do/while akan mengecek kondisi di belakang (sesudah mengulang), sedangkan while akan mengecek kondisi di depan atau awal (sebelum mengulang).”

Contoh :

```
#include <stdio.h>
int main()
{
    int num = 0;
    do
    {
        printf("Num sekarang adalah %d\n",num);
        printf("Masukkan bilangan integer positif (-1 untuk keluar ): ");
        scanf("%d", &num);
    } while (num != -1);
    return 0;
}
```

❖ Perulangan Bersarang (Nested Loop)

Di dalam blok perulangan, kita juga dapat membuat perulangan.

Ini disebut dengan nested loop atau perulangan bersarang atau perulangan di dalam perulangan.

Contoh :

```c
#include <stdio.h>

int main() {

    int n = 5;

    // Looping luar
    for(int i = 1; i <= n; i++) {
        // Looping dalam
        for(int j = 1; j <= i; j++) {
            printf("* ");
        }

        printf("\n");
    }

    return 0;
}
```

Hasil Output :

```
*
* *
* * *
* * * *
* * * * *
```

Pada output tersebut, kita dapat melihat bahwa jumlah bintang pada setiap baris bertambah sesuai dengan nomor barisnya. Perulangan Bersarang (Nested Loop) sangat berguna untuk menghasilkan pola seperti ini, serta untuk berbagai aplikasi lain yang memerlukan iterasi di dalam iterasi, seperti pengolahan matriks atau tabel.

Break and Continue

Keyword break dan continue digunakan untuk mengendalikan (kontrol) alur pada perulangan. Berikut penjelasannya.

❖ Break

Perintah break digunakan untuk menghentikan perulangan (secara paksa). Apabila perintah break pada suatu perulangan dijalankan, maka perulangan tersebut akan dihentikan (secara paksa) dari titik dimana perintah break muncul.

Contoh :

```c
#include <stdio.h>

int main()
{
    int i;
    for (i = 1; i <= 6; i++) {
        printf("%d\n", i);
        //   Jika i adalah 4, maka keluar dari perulangan
        if (i == 4) {
            break;
        }
    }
    return 0;
}
```

❖ Continue

Kebalikan dari perintah break, perintah continue digunakan untuk melanjutkan perulangan. Pada perulangan, apabila menemui perintah continue, maka perintah-perintah dibawah continue akan diabaikan dan kembali akan mengevaluasi ekspresi/kondisi. Sedangkan pada perulangan for akan langsung mengekekusi bagian update_statement kemudian mengevaluasi ekspresi/kondisi.

Contoh code ada pada halaman selanjutnya.

Contoh penggunaan continue:

```c
#include <stdio.h>

int main()
{
    int i;
    for (i = 1; i <= 6; i++) {
        // Jika i adalah 4, maka abaikan perintah printf()
        if (i == 4) {
            continue;
        }
        printf("%d\n",i);
    }
    return 0;
}
```

❖ Infinite Loop

Infinite loop adalah kasus di mana perulangan tidak akan pernah berhenti. Hal ini terjadi karena perulangan tidak akan pernah menemui kondisi yang membuatnya berhenti.

Contoh di bawah akan menghasilkan infinite loop.

```c
#include <stdio.h>

int main()
{
    int i;
    for (i = 1; i <= 100; i--) {
        // Perulangan tak akan pernah berhenti
    }
}
```

Array

❖ Pengenalan Array

Array merupakan jenis struktur data yang menampung elemen betipe data sama secara sekuensial dengan ukuran (kapasitas) yang tetap (fixed-size). Bayangkanlah sebuah array sebagai sekumpulan elemen sejenis yang disusun secara berurutan pada satu identifier (nama).

Kesimpulannya, Array merupakan struktur data yang digunakan untuk menyimpan sekumpulan data dalam satu tempat.

❖ Deklarasi Array

Array juga sama seperti variabel, perlu dideklarasikan terlebih dahulu sebelum bisa digunakan. Deklarasi array sama seperti variabel, hanya saja saat pendeklarasiannya perlu dituliskan ukurannya.

```
tipe_data identifier_array[size];
```

❖ Inisialisasi Array

Kita juga bisa menginisialisasi elemen-elemen pada array setelah dideklarasikan. Sintaksnya adalah seperti berikut.

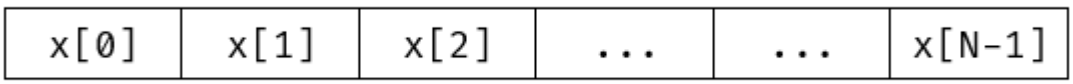
```
tipe_data identifier_array[size] = {elem1, elem2, elem3, ....}
```

Jadi, pada pemrograman C, array dapat kita buat dengan cara seperti ini.

```
// membuat array kosong dengan tipe data integer dan panjang 10
int nama_array[10];

// membuat array dengan langsung diisi
int nama_arr[3] = {0, 3, 2}
```

❖ Mengakses Array

Seperti yang telah dijelaskan sebelumnya, array disimpan secara sekuensial (pada blok memori secara berurutan). Lalu bagaimana kita mengakses tiap elemennya? Pengaksesan elemen pada array dilakukan dengan menuliskan identifier array-nya lalu digabung dengan menggunakan operator subscript `[]` dengan menyertakan indeks didalamnya.

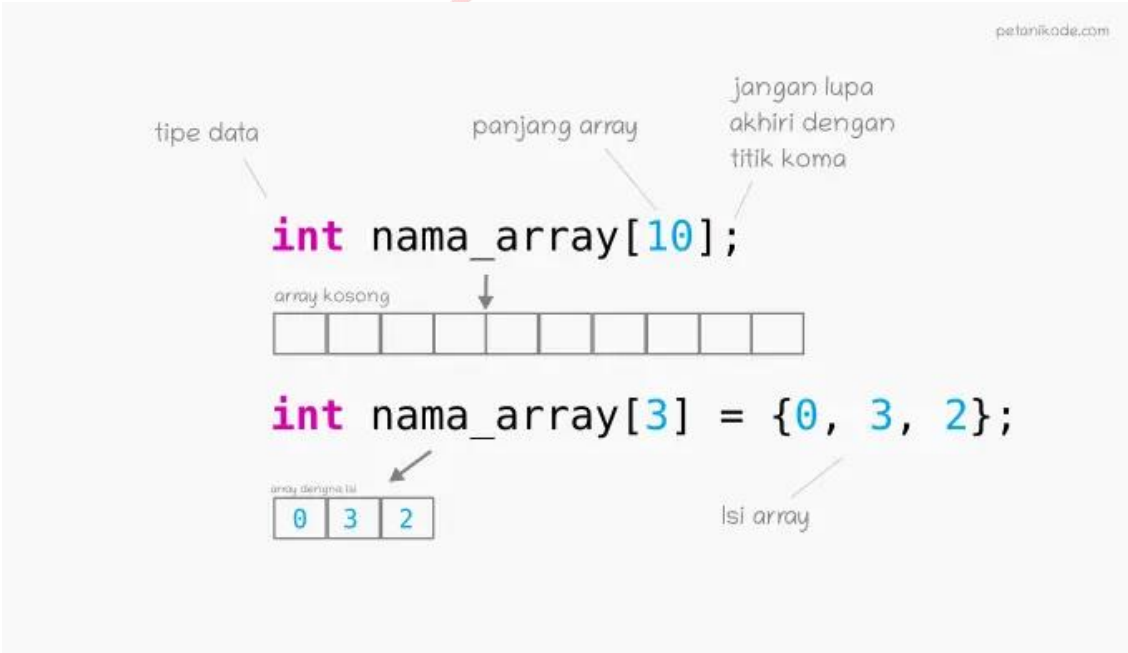


Indeks pada array menggunakan *zero-based index*, yang artinya elemen pertama pada array ditunjukkan oleh indeks ke 0 (bukan ke 1) dan elemen terakhir ditunjukkan oleh indeks ke N-1 (misal N adalah panjang array). Elemen-elemen pada array dapat diperlakukan sama seperti halnya variabel. Kita dapat melakukan assignment, operasi aritmatika, dan lain-lain.

Contoh :

```c
#include <stdio.h>

int main ()
{

  int a[10]; //Deklarasi Array

  int b[5] = {0, 1, 2, 3, 4}; //Inisialisasi Array

  a[0] = 50;
  a[1] = 20;

  printf("%d %d\n", a[0], a[1]);

  return 0;

}
```

Mengapa kita perlu array? Andaikan kita membutuhkan 1000 inputan data, kita tidak perlu membuat deklarasi variabel berjumlah 1000, misalkan variabel a1 hingga a1000. Namun, kita cukup menuliskan 1 identifier array berkapasitas 1000.



Dimensi Array

❖ Array Satu Dimensi

Sebuah array dikatakan berdimensi satu apabila tiap elemennya hanya menyimpan satu data/objek. Contoh-contoh pada penjelasan sebelumnya merupakan array satu dimensi.

Contoh array satu dimensi :

```
int main()
{
    int arr[5];
    arr[0] = 4;
    arr[1] = 2;
    arr[2] = 3;
    return 0;
}
```

Jika diilustrasikan, maka array tersebut akan tampak seperti di bawah.

index	0	1	2	3	4
nilai	4	2	3	?	?

❖ Array Multidimensi

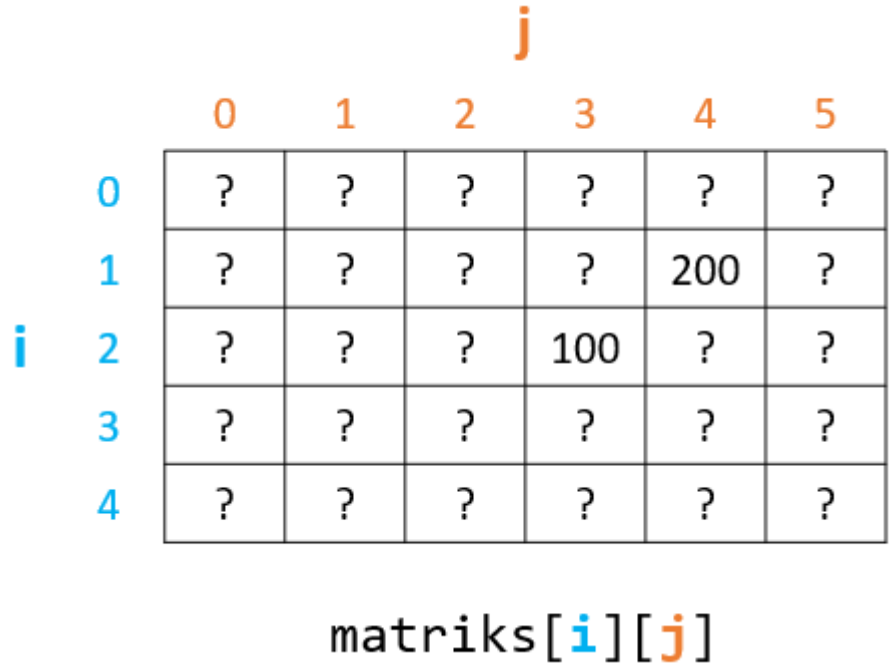
Sebuah array dikatakan multidimensional apabila tiap elemen array menampung array lainnya. Apabila array satu dimensi hanya memiliki sebuah index, array multidimensi memiliki dua atau lebih index untuk mengakses elemen dalam array tersebut.

Cara deklarasinya pun berbeda dari array satu dimensi. Kita memerlukan N buah kurung siku untuk membuat array dengan N-dimensi.

Berikut adalah contoh program dengan array dua dimensi:

```
int main ()
{
    int matriks[5][6];
    matriks[2][3] = 100;
    matriks[1][4] = 200;
    return 0;
}
```

Apabila diilustrasikan, bentuk array dua dimensi layaknya baris dan kolom, seperti gambar di bawah.



Selain bentuk dua dimensi, kita dapat membuat array hingga N-dimensi, sesuai kebutuhan.

String

❖ Pengenalan String

Secara umum, string merupakan kumpulan dari satu atau lebih karakter. Spesifik pada bahasa C, string didefinisikan sebagai kumpulan karakter yang diakhiri oleh karakter null ('\0').

Misalkan string "Dasar", pada bahasa C direpresentasikan sebagai kumpulan karakter 'D', 'a', 's', 'a', 'r', dan '\0'.

❖ Representasi String

Pada bahasa C, string direpresentasikan oleh array bertipe char. Contoh pendeklarasian string :

```c
#include <stdio.h>

int main ()
{
    char str[] = "Halo";
    return 0;
}
```

Contoh di atas akan mendeklarasikan string bernama str dengan kapasitas 5 karakter, di mana str[0] = 'H', str[1] = 'a', str[2] = 'l', str[3] = 'o', dan str[4] = '\0'. Perhatikan bahwa str[4] berisi karakter '\0' (null character), walaupun dalam literal string di atas tidak ada karakter tersebut.

Dalam bahasa C, karakter null digunakan untuk menandakan akhir dari sebuah string.

Contoh pendeklarasian string (2) :

```c
#include <stdio.h>

int main () {

  char array[10];

  return 0;
}
```

Contoh di atas akan mendeklarasikan string bernama array yang dapat menampung maksimal 10 karakter, termasuk null character.

Untuk menerima input string dari user, kita dapat menggunakan scanf atau gets. Perintah scanf akan membaca inputan string dari user dan berhenti ketika ada whitespace ataupun interupsi dari pengguna. Sedangkan gets akan membaca satu baris kumpulan karakter hingga enter atau interupsi dari pengguna.

Contoh source code penggunaan scanf untuk membaca string :

```c
#include <stdio.h>

int main () {

    char arr[100];
    while(1)
    {
        scanf("%s", arr);
        printf("-- %s\n", arr);
    }
    return 0;

}
```

Contoh penggunaan gets untuk membaca string :

```c
#include <stdio.h>

int main () {

    char arr[100];
    while(true)
    {
        gets(arr);

        printf("-- %s\n", arr);
    }
  return 0;

}
```

String yang dibaca dengan mengunakan scanf atau gets akan secara otomatis memiliki null character di akhir.

❖ Fungsi-Fungsi String

Ada beberapa fungsi yang bisa kita gunakan untuk memanipulasi string sesuai dengan kebutuhan.

Dalam bahasa pemrograman C, terdapat library yang dibuat dengan tujuan memudahkan pengguna dalam mengolah string. Library tersebut tersimpan dalam <string.h>, oleh karena itu, untuk mengakses library ini, diperlukan tambahan preprocessor, yaitu:

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

Berikut adalah fungsi-fungsi yang dibagi berdasarkan kegunaannya dalam mengolah sebuah string (diambil dari www.cplusplus.com) :

Copying	
memcpy	Menyalin blok memori
memmove	Memindahkan blok memori
strcpy	Menyalin string
strncpy	Menyalin karakter string

Concatenation	
strcat	Menggabungkan string
strncat	Menambahkan karakter dari string

Comparison	
memcmp	Membandingkan 2 blok memori
strcmp	Membandingkan 2 string
strcoll	Membandingkan 2 string menggunakan lokal
strncmp	Membandingkan karakter dari 2 string
strxfrm	Ubah string menggunakan lokal

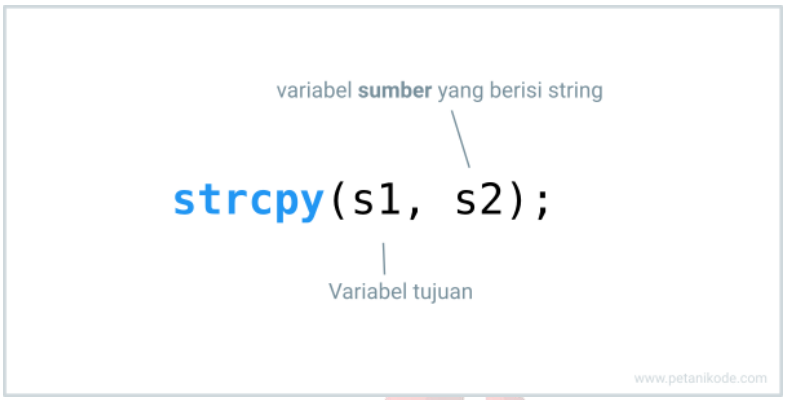
Searching	
memchr	Menemukan karakter dalam blok memori
strchr	Menemukan kemunculan sebuah karakter string
strcspn	Mendapatkan rentang karakter dalam string
strpbrk	Menemukan karakter dalam string
strrchr	Menemukan kemunculan terakhir karakter dalam string
strspn	Mendapatkan rentang himpunan karakter dalam string
strstr	Menemukan substring
strtok	Memisahkan string menjadi token-token

Other	
memset	Mengisi blok memori
strerror	Mendapatkan pointer ke string dengan pesan error
strlen	Mendapatkan panjang string

Berikut adalah beberapa fungsi dan penjelasannya :

1. Fungsi strcpy()

Fungsi strcpy() digunakan untuk meng-copy string dari sebuah variabel ke variabel yang lainnya.

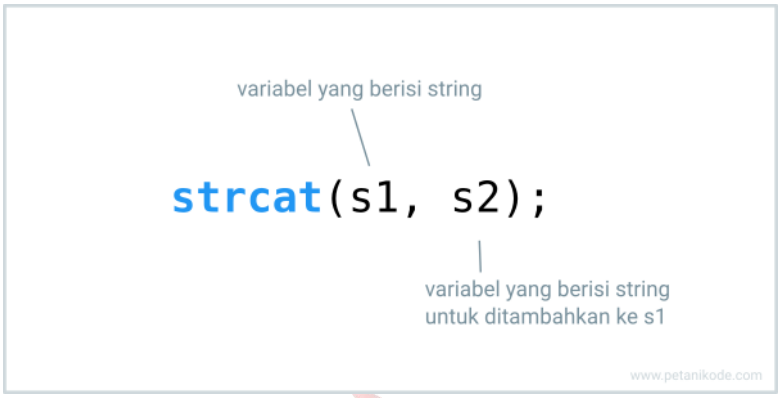


Contoh penggunaan dalam kode program :

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

int main () {

    char a[] = "Halo";
    char b[10];

    // Copy string a ke string b
    strcpy(b, a);

    printf("%s\n", b);

    return 0;

}
```

2. Fungsi strcat()

Fungsi strcat digunakan untuk melakukan penempelan sebuah string pada akhir string yang lain.

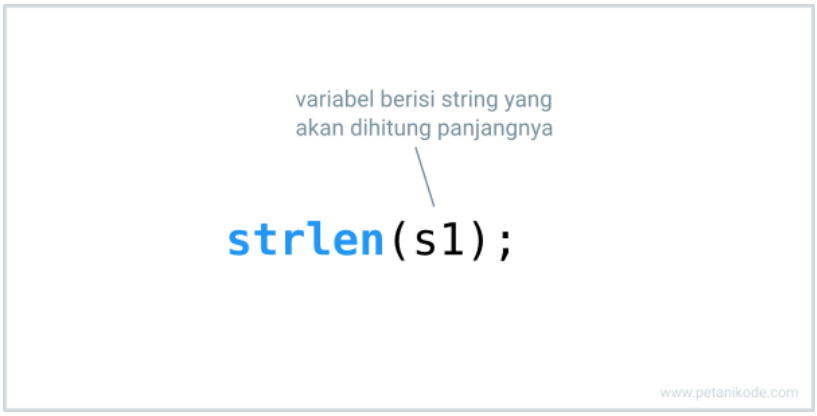


Contoh penggunaan dalam kode program :

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

int main () {

    char a[] = "Halo";
    char b[] = " Kawan";
    char c[20];

    // Copy string a ke string c
    strcpy(c, a);

    // Tempelkan string b ke akhir string c
    strcat(c, b);

    printf("%s\n", c);

    return 0;

}
```

3. Fungsi strlen()

Fungsi strlen() digunakan untuk menghitung panjang string.



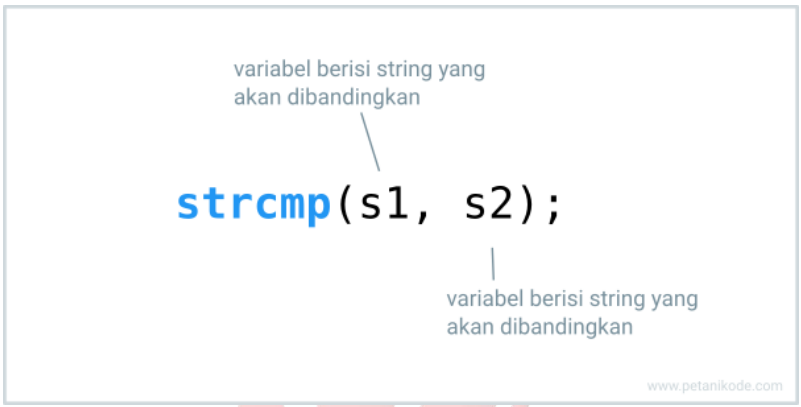
Contoh penggunaan dalam kode program :

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

int main () {

    char a[] = "Halo";

    printf("Panjang string a adalah %d\n", strlen(a));

    return 0;

}
```

4. Fungsi strcmp()

Fungsi strcmp() digunakan untuk membandingkan string dengan string yang lainnya.

Return value dari fungsi ini dapat berupa bilangan negatif, nol ataupun positif.
Fungsi ini akan menghasilkan nilai 0 apabila kedua string yang dibandingkan sama.
Jika fungsi ini mengembalikan nilai negatif, maka *str1* memiliki tingkat leksikografi lebih kecil dari *str2*. Sedangkan jika fungsi ini mengembalikan nilai postifif, maka *str1* memiliki tingkat leksikografi lebih besar dari *str2*.



Berikut adalah contoh penggunaan fungsi ini dalam kode progam :

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

int main () {

    char a[] = "Halo";
    char b[] = "Hai";
    char c[] = "Halo";

    if(strcmp(a, b) == 0) printf("String a sama dengan b\n");
    else printf("String a tidak sama dengan b\n");

    if(strcmp(a, c) == 0) printf("String a sama dengan c\n");
    else printf("String a tidak sama dengan c\n");

    return 0;

}
```

Source :

"Modul 0: Pengenalan Pemrograman". github.com/AlproITS. 9 Oktober 2020. 14 Juli 2024. https://github.com/AlproITS/DasarPemrograman/wiki/Modul-0:-Pengenalan-Pemrograman

"Belajar Pemrograman C #03: Struktur Dasar dan Aturan Penulisan Program C yang Harus dipahami". petanikode.com. 20 Januari 2019. 14 Juli 2024. https://www.petanikode.com/c-syntak/

"Belajar Pemrograman C #04: Mengenal Fungsi Input dan Output pada C". petanikode.com. 10 Maret 2019. 15 Juli 2024. https://www.petanikode.com/c-input-output/

"Belajar Pemrograman C #05: Mengenal Variabel, Tipe Data, Konstanta". petanikode.com. 18 Mei 2019. 15 Juli 2024. https://www.petanikode.com/c-variabel/

"Belajar Pemrograman C #06: Enam Macam Operator yang Harus diketahui pada C". petanikode.com. 18 Mei 2019. 16 Juli 2024. https://www.petanikode.com/c-operator/

"Modul 1: Percabangan". github.com/AlproITS. 12 Oktober 2020. 19 Juli 2024. https://github.com/AlproITS/DasarPemrograman/wiki/Modul-1:-Percabangan

"Belajar Pemrograman C #07: Mengenal 6 Macam Bentuk Blok Percabangan". petanikode.com. 18 Mei 2019. 19 Juli 2024. https://www.petanikode.com/c-percabangan/

"Belajar Bahasa C #6 : Macam Bentuk Percabangan Bahasa C." praktekotodidak.com. 5 Januari 2023. 20 Juli 2024. https://praktekotodidak.com/percabangan-bahasa-c/

"Decision Making in C (if , if..else, Nested if, if-else-if)". geeksforgeeks.org. 10 Juli 2024. 20 Juli 2024. https://www.geeksforgeeks.org/decision-making-c-cpp/

"Modul 2: Perulangan, Array, dan String". github.com/AlproITS. 5 November 2020. 23 Juli 2024. https://github.com/AlproITS/DasarPemrograman/wiki/Modul-2:-Perulangan,-Array,-dan-String

"Belajar Pemrograman C #08: Memahami Blok Perulangan pada C". petanikode.com. 18 Mei 2019. 23 Juli 2024 https://www.petanikode.com/c-perulangan/

"Belajar Pemrograman C #10: Mengenal Struktur Data Array pada C". petanikode.com.18 Mei 2019. 23 Juli 2024. https://www.petanikode.com/c-array/

"Decision Making in C (if , if..else, Nested if, if-else-if)". geeksforgeeks.org. 10 Juli 2024. 20 Juli 2024. https://www.geeksforgeeks.org/decision-making-c-cpp/

"Tutorial Belajar C Part 37: Perulangan FOR Bahasa C". duniailkom.com. 23 Maret 2019. 23 Juli 2024 https://www.duniailkom.com/tutorial-belajar-c-perulangan-for-bahasa-c/

"Tutorial Belajar C Part 38: Perulangan WHILE Bahasa C". duniailkom.com. 24 Maret 2019. 23 Juli 2024 https://www.duniailkom.com/tutorial-belajar-c-perulangan-while-bahasa-c/

"Belajar Bahasa C : #5 Perulangan(Looping)". sobatambisius.com 2 September 2021. 23 Juli 2024. https://www.sobatambisius.com/2021/09/belajar-bahasa-c-5-perulanganlooping.html

"Belajar Pemrograman C #14: Mengenal Tipe Data String pada C". petanikode.com. 26 November 2019. 24 Juli 2024. https://www.petanikode.com/c-string/