LECTURE NOTES

ALGO.BINUS COMPREHENSIVE GUIDE

by JOLLYBEE

Data Types

Dalam bahasa pemrograman C, dikenal beberapa tipe data seperti berikut:

Tipe Data	Ukuran	Batas Nilai	Format	Keterangan
char	1 byte	-128 s/d 127	%c	Karakter/string
int	4 bytes	-2,147,483,648 s/d 2,147,483,647	%d	Integer/bilangan bulat
unsigned int	4 bytes	0 s/d 4,294,967,295	%u	Integer/bilangan bulat (non-negatif)
long long int	8 bytes	-9,223,372,036,854,775,808 s/d 9,223,372,036,854,775,807	%lld	Integer/bilangan bulat
unsigned long long int	8 bytes	0 s/d 18,446,744,073,709,551,615	%llu	Integer/bilangan bulat (non-negatif)
float	4 bytes	-3.4E-38 s/d 3.4E+38	%f	Float/bilangan pecahan
double	8 bytes	-1.7E-308 s/d 1.7E+308	%lf	Bilangan pecahan presisi ganda

Contoh deklarasinya adalah sebagai berikut:

Source Code

```
int angka;
float angkaKoma = 5.2432343;
char huruf = 'c';
unsigned long long int angkaBesarBanget = 10000000000000000000;
```

Exercise 1

1. Apa yang terjadi jika nilai suatu variabel melebihi batasan nilai tipe datanya?
2. Apa yang terjadi jika suatu variabel dengan tipe data unsigned int diberi nilai negatif?
3. Jika x di-deklarasikan sebagai int x = 7.60, berapakah nilai x?

IO (Input & Output)

Dalam bahasa pemrograman C, input dan output dari dan ke console dapat dilakukan dengan fungsi yang bernama scanf dan printf. Untuk menggunakan fungsi tersebut, dibutuhkan header file bernama stdio.h**.**

Stdio merupakan singkatan dari standard input output.

Scanf dan printf memiliki struktur seperti ini:

Source Code

```
scanf("Format", &variabel);
printf("Format", variabel);
```

Perhatikan penambahan simbol & di awal variabel penampung. Simbol ini merujuk ke pointer untuk alamat memory dari variabel tersebut, seolah-olah kita memasukan data ke alamat tersebut. Tetapi perlu diperhatikan bahwa penambahan simbol & tidak berlaku untuk tipe data string (char str[]), selebihnya akan dibahas di bagian string.

Berikut adalah contoh program yang meminta user untuk sebuah angka dan menampilkannya lagi di layar:

Source Code

```
#include <stdio.h>

int main() {

	int n;
	printf("Masukkan sebuah angka: ");
	scanf("%d", &n);
	printf("Angka yang di-inputkan oleh user adalah: %d", n);
	//ini adalah inline comment

	/*
	ini
	adalah
	multi-line
	comment
	*/
	return 0;
}
```

Output

```
Masukkan sebuah angka: 7
Angka yang di-inputkan oleh user adalah: 7
```

Ada beberapa karakter spesial dalam bahasa pemrograman C yang kita sebut escape sequences. Berikut adalah beberapa escape sequences tersebut:

Escape Sequences	Fungsi
\'	Print karakter ‘
\”	Print karakter “
\?	Print karakter ?
\\	Print karakter \
\b	Backspace (menghapus karakter sebelumnya)
\n	New line (membuat baris baru)
\r	Carriage return (kembali ke awal baris)
\t	Horizontal tab
\v	Vertical tab

Exercise 2

1. Apa output yang dihasilkan cuplikan program di bawah?

Source Code
```
printf("I\nCan\nShow\nYou\nThe\nWorld\n");
```

2. Apa yang salah pada cuplikan program di bawah?

Source Code
```
#include <stdoi.h>

int main() {

	int n;
	scanf("%c", n);
	printf("%d\n", n);
	return 0;
}
```

3. Apa guna simbol & pada fungsi scanf?

Operators

Operator adalah simbol khusus yang biasa dilibatkan dalam pembuatan program untuk melakukan suatu operasi matematika maupun manipulasi logika. Beberapa jenis operator yang ada di bahasa pemgrograman C adalah:

- Assignment operators
- Arithmetic operators
- Comparison operators
- Logical operators

#1 Assignment Operators

Operator ini berfungsi untuk memberikan nilai pada variabel. Simbol = (sama dengan) digunakan untuk memberi nilai. Operasi ini berlangsung dari kanan ke kiri dengan mengambil nilai yang ada di kanan simbol = dan menaruhnya di variabel yang ada di kiri.

Source Code

```
#include <stdio.h>

int main(){

    int a, b;
    a = 10;
    b = 7;
    a = b;
    a = 4;

    printf("a:%d b:%d", a, b);
    return 0;
}
```

Output

```
a:4 b:7
```

#2 Arithmetic Operators

Operator ini digunakan untuk melakukan operasi aritmetika dalam bahasa pemrograman. Terdapat beberapa macam operasi aritmetika, contohnya adalah:

Simbol	Fungsi	Contoh
+	Penjumlahan	x+y
-	Pengurangan	x-y
*	Perkalian	x*y
/	Pembagian (pembulatan ke bawah untuk tipe data integer)	x/y
%	Modulus (mencari hasil sisa bagi)	x%y
++	Increment (menambahkan nilai dengan 1)	x++ atau ++x
--	Decrement (mengurangi nilai dengan 1)	y-- atau --y

Source Code

```
#include <stdio.h>

int main(){

    int a, b;
    a = 10;
    b = 7;

    printf("Tambah: %d\n", a+b);
    printf("Kurang: %d\n", a-b);
    printf("Kali: %d\n", a*b);
    printf("Bagi: %d\n", a/b);
    printf("Modulus: %d\n", a%b);
    printf("Postfix Increment: %d\n", a++);
    printf("Prefix Increment: %d\n", ++a);
    printf("Postfix Decrement: %d\n", b--);
    printf("Prefix Decrement: %d\n", --b);
    return 0;
}
```

Output

```
Tambah: 17
Kurang: 3
Kali: 70
Bagi: 1
Modulus: 3
Postfix Increment: 10
Prefix Increment: 12
Postfix Decrement: 7
Prefix Decrement: 5
```

#3 Relational and Comparison Operators

Operator relasi akan banyak kita gunakan dalam statement bersyarat yang selalu menghasilkan nilai *true*(1) atau *false*(0). Terdapat beberapa macam operasi relasi, contohnya adalah:

Simbol	Fungsi	Contoh
==	Sama dengan	x == y
!=	Tidak sama dengan	x != y
>	Lebih besar	x > y
<	Lebih kecil	x < y
>=	Lebih besar atau sama dengan	x >= y
<=	Lebih kecil atau sama dengan	x <= y

Source Code

```
#include <stdio.h>

int main(){

    printf("18 > 5 : %d\n", 18 > 5);
    printf("3 < 12 : %d\n", 3 < 12);
    printf("6 >= 6 : %d\n", 6 >= 6);
    printf("5 <= 4 : %d\n", 5 <= 4);
    printf("5 == 5 : %d\n", 5 == 5);
    printf("7 != 7 : %d\n", 7 != 7);
    return 0;
}
```

Output

```
18 > 5 : 1
3 < 12 : 1
6 >= 6 : 1
5 <= 4 : 0
5 == 5 : 1
7 != 7 : 0
```

Simbol = (single equal) dan == (double equal) memiliki kegunaan yang berbeda. Simbol = digunakan sebagai assignment operator, yaitu untuk memberikan nilai yang ada di kanan kepada variabel yang ada di kiri, seperti x = 5, y = 7. Sedangkan simbol == digunakan sebagai relational operator untuk membuat perbandingan antar dua nilai dan menghasilkan output *true* atau *false*.

Perlu diketahui bahwa komparasi antara dua bilangan yang tidak bulat(float atau double) dengan operator == sangat tidak dianjurkan karena adanya efek pembulatan akibat bits yang terbatas. Untuk melakukan komparasi antara dua bilangan tidak bulat, dapat dilakukan seperti berikut.

Source Code

```
#include <stdio.h>
#include <math.h> //fungsi fabs ada di header file ini

int main(){
	double EPS = 1e-9;
	double a = 0.5000;
	double b = 1.0/2.0;
	if(fabs(a-b) < EPS){
		printf("Sama\n");
	}
	else{
		printf("Berbeda\n");
	}
}
```

fabs adalah fungsi dari library math.h yang bertujuan untuk meng-absolutkan sebuah nilai yang tidak bulat. EPS disini bernilai 1e-9, sehingga apabila selisih dari a dan b lebih kecil dari EPS, kita anggap 2 nilai itu sama.

#4 Logical Operators

Operator logika adalah operator yang digunakan untuk membandingkan nilai dua variabel atau lebih. Hasil dari operasi ini sama dengan operator relasi, *true*(1) atau *false*(0).

Simbol	Fungsi	Contoh
&&	AND – Jika semua operand bernilai *true*, maka output juga *true*.	x < 5 && y > 7
||	OR – Jika salah satu operand bernilai *true*, maka output juga *true*.	x > 0 || x < 20
!	NOT – Digunakan untuk membalik kondisi, jika kondisi awal adalah *true* maka akan berubah menjadi *false*, dan begitu pula sebaliknya.	!x

Source Code

```
#include <stdio.h>

int main(){

    int x = 5;
    int y = 7;

    printf("%d\n", (x == 5) && (y < 7));
    printf("%d\n", (x == 5) || (y < 7));
    printf("%d\n", !(x==5));
    return 0;
}
```

Output

```
0
1
0
```

Exercise 3

1. Apa perbedaan antara postfix(x++ atau x--) dan prefix(++x atau --x) operator?
2. Adakah cara lain untuk menambahkan nilai variabel a dengan 2 selain a = a+2 ?
3. Manakah yang dikerjakan terlebih dahulu, perkalian atau penambahan?
4. Berapakah nilai x jika x = 3*2+5/3 ?
5. Apa hasil dari !((P && Q) || (!P || (P && !Q))) jika P = *true* dan Q = *false* ?

Control Statements

Struktur kontrol memungkinkan programmer untuk menentukan alur program. Salah satu struktur kontrol yang ada di bahasa pemrograman C adalah percabangan. Percabangan merupakan salah satu pernyataan yang digunakan untuk mengambil keputusan terhadap satu atau beberapa kemungkinan. Ada dua tipe percabangan di bahasa pemrograman C, yaitu:

- If Else
- Switch

#1 If Else

Secara sederhana, fungsi if dapat ditulis seperti berikut:

Source Code

```
if(kondisi){
      //kode yang dijalankan jika bernilai true
}
else{
      //kode yang dijalankan jika bernilai false
}
```

Source Code

```
#include <stdio.h>

int main(){

    int angka;
    printf("Inputkan sebuah angka: ");
    scanf("%d", &angka);

    if(angka > 10){
      printf("Angka lebih besar dari 10\n");
    }
    else if(angka > 5){
      printf("Angka lebih besar dari 5 namun lebih kecil atau sama dengan
10\n");
    }
    else{
      printf("Angka lebih kecil dari atau sama dengan 5\n");
    }
    return 0;
}
```

Output

```
Inputkan sebuah angka: 12
Angka lebih besar dari 10
```

Output

```
Inputkan sebuah angka: 7
Angka lebih besar dari 5 namun lebih kecil atau sama dengan 10
```

Output

```
Inputkan sebuah angka: -3
Angka lebih kecil dari atau sama dengan 5
```

Dalam program diatas, jika angka yang diinput oleh user lebih dari 10, maka program akan mencetak "Angka lebih besar dari 10". Namun jika angka yang diinput oleh user tidak lebih dari 10, maka pengecekan (if) pertama akan menghasilkan nilai *false* dan program akan mengecek lagi apakah nilai variabel angka lebih besar dari 5. Jika pengecekan (if) masih menghasilkan nilai *false*, maka program secara default akan menjalankan instruksi yang ada di dalam statement else, yaitu print "Angka lebih kecil dari atau sama dengan 5".

Di dalam sebuah statement if, boleh juga terdapat lebih dari satu kondisi, contohnya:

Source Code

```
#include <stdio.h>

int main(){

	int x = 3;
	int y = 5;
	if(x == 3 && y == 5){
		printf("x sama dengan %d, y sama dengan %d\n", x, y);
	}
	return 0;
}
```

Output

```
x sama dengan 3, y sama dengan 5
```

Nested if atau if bersarang adalah sebuah statement if di dalam statement if lainnya. Terkadang, kita akan dihadapkan pada kondisi yang sedikit lebih rumit dan statement if di dalam statement if lainnya dapat mempermudah menentukan solusi untuk kondisi tersebut.

Source Code

```
#include <stdio.h>

int main(){

	int burger = 1;
	int cheese = 2;

	if(burger >= 1){
		if(cheese == 0){
			printf("Saya memesan burger biasa\n");
		}
		else if(cheese == 1){
			printf("Saya memesan cheeseburger\n");
		}
		else if(cheese == 2){
			printf("Saya memesan double-cheeseburger\n");
		}
		else{
			printf("Saya memesan triple-cheeseburger\n");
		}
	}
	else{
		printf("Saya tidak memesan burger\n");
	}
	return 0;
}
```

Output

```
Saya memesan double-cheeseburger
```

#2 Switch & Case

Switch pada dasarnya memiliki fungsi yang sama dengan percabangan if else. Pernyataan switch dapat ditulis sebagai berikut:

Source Code

```
switch(operand atau kondisi){
	case konstan1:
		//lakukan sesuatu
		break;
	case konstan2:
		//lakukan sesuatu
		break;
	default:
		//lakukan sesuatu
}
```

Pernyataan switch mengevaluasi operand dan memeriksa apakah operand/kondisi memiliki nilai sama dengan konstan1, jika iya maka perintah yang di bawahnya akan di-eksekusi hingga ia menemukan perintah break. Jika tidak ada perintah break, maka switch akan terus memeriksa operand/kondisi kepada case yang ada di bawahnya hingga kondisi terpenuhi. Namun, jika nilai operand/kondisi tidak sesuai dengan konstanta yang ditentukan sebelumnya, maka program akan mengeksekusi pernyataan default.

Source Code

```
#include <stdio.h>

int main(){

    printf("Daftar menu:\n");
    printf("1. Bakmie\n2. Ayam Goreng\n3. Steak\n");

    int pilihan;
    printf("Masukkan pilihan anda (1-3): ");
    scanf("%d", &pilihan);

    switch(pilihan){
      case 1:
            printf("Saya Suka Bakmie\n");
            break;
      case 2:
            printf("Saya Suka Ayam Goreng\n");
            break;
      case 3:
            printf("Saya Suka Steak\n");
            break;
      default:
            printf("Saya tidak suka ketiganya");
    }
    return 0;
}
```

Output

```
Daftar menu:
1. Bakmie
2. Ayam Goreng
3. Steak
Masukkan pilihan anda (1-3): 2
Saya Suka Ayam Goreng
```

Exercise 4

1. Apakah sebuah pernyataan if harus selalu diikuti dengan pernyataan else?
2. Apakah sebuah pernyataan if harus selalu diikuti dengan pernyataan else if?
3. Apa perbedaan antara switch dan if?
4. Apa yang terjadi jika tidak ada perintah break di dalam switch?
5. Bolehkah default pada switch dihilangkan?

Looping

Looping atau perulangan adalah instruksi program yang bertujuan untuk mengulang beberapa baris perintah. Ada 3 cara untuk melakukan perulangan di bahasa pemrograman C, yaitu:

- For
- While
- Do-While

#1 For

Perulangan dalam bentuk for memiliki struktur sebagai berikut:

Source Code

```
for(deklarasi; kondisi; perubahan){
   //lakukan sesuatu
}
```

Perulangan for biasa digunakan ketika kita tahu berapa kali perulangan perlu dilakukan. Deklarasi dapat diisi dengan deklarasi variabel untuk perulangan. Kondisi biasanya berupa ekspresi yang menghasilkan boolean, untuk menandakan apakah perulangan sudah patut diberhentikan. Perubahan merupakan bagian yang di-eksekusi pada akhir setiap siklus perulangan. Perubahan biasanya adalah tahap dari deklarasi menuju kondisi. Berikut adalah contoh program yang menggunakan perulangan for:

Source Code

```
#include <stdio.h>

int main() {
	int n;
	printf("Masukkan sebuah angka: ");
	scanf("%d", &n);
	for (int i = 0; i < n; i++) {
		printf("tulisan ini dicetak saat i = %d\n", i);
	}

	return 0;
}
```

Output

```
Masukkan sebuah angka: 3
tulisan ini dicetak saat i = 0
tulisan ini dicetak saat i = 1
tulisan ini dicetak saat i = 2
```

#2 While

Dalam perulangan while, program akan terus melakukan perulangan selama kondisi tertentu bernilai benar. Perulangan while dalam bahasa pemrograman C memiliki struktur sebagai berikut:

Source Code

```
while(kondisi){
	//lakukan sesuatu
}
```

Perulangan while biasa digunakan ketika tidak diketahui harus berapa kali serangkaian perintah dilaksanakan, tetapi diketahui perintah-perintah itu perlu dilaksanakan selama suatu kondisi terpenuhi. Selama nilai dari kondisi adalah *true*, seluruh perintah di dalamnya akan di-eksekusi secara berurutan. Berikut adalah contoh program yang menggunakan perulangan while:

Source Code

```
#include <stdio.h>

int main() {
	int n;
	printf("Masukkan sebuah angka: ");
	scanf("%d", &n);
	int i = 0;
	while (i < n) {
		printf("tulisan ini dicetak saat i = %d\n", i);
		i++;
	}

	return 0;
}
```

Output

```
Masukkan sebuah angka: 4
tulisan ini dicetak saat i = 0
tulisan ini dicetak saat i = 1
tulisan ini dicetak saat i = 2
tulisan ini dicetak saat i = 3
```

#3 Do-While

Perulangan do-while dalam bahasa pemrograman C memiliki struktur sebagai berikut:

Source Code

```
do{
      //lakukan sesuatu
}while(kondisi);
```

Perulangan do-while biasa digunakan ketika tidak diketahui harus berapa kali serangkaian perintah dilaksanakan, tetapi diketahui perintah-perintah itu perlu dilaksanakan selama suatu kondisi terpenuhi. Selama nilai dari kondisi adalah *true*, seluruh perintah di dalamnya akan di-eksekusi secara berurutan.

Perulangan while dan do-while pada dasarnya hampir sama. Namun perbedaan terletak pada lokasi pengecekan kondisi perulangan. Dalam struktur while, pengecekan dilakukan di awal sebelum perulangan berjalan, sehingga jika kondisi tidak terpenuhi, maka perulangan tidak akan dijalankan. Sedangkan pada perulangan do-while, pengecekan kondisi akan dilakukan di akhir perulangan, sehingga seluruh perintah akan dilakukan terlebih dahulu, baru diperiksa apakah kondisi masih terpenuhi. Bila ya, maka seluruh perintah akan diulang. Hal ini menjamin seluruh perintah dijalankan paling sedikit satu kali. Berikut adalah contoh program yang menggunakan perulangan do-while:

Source Code

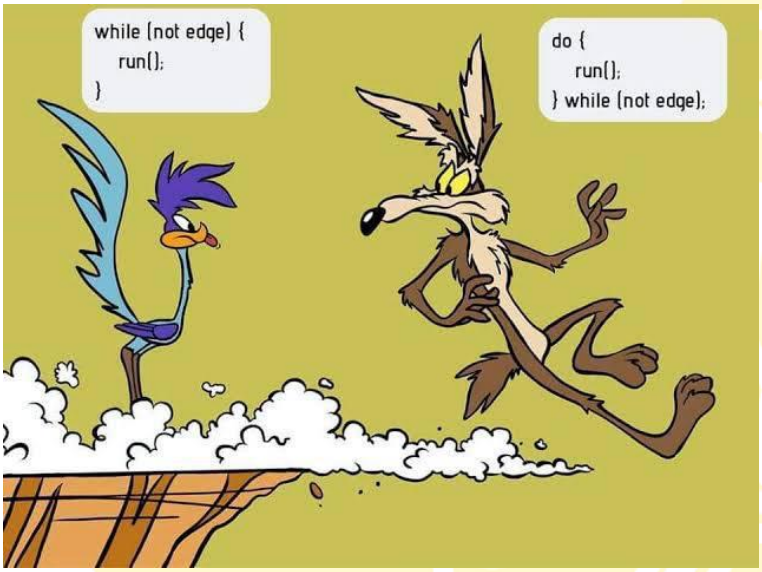
```
#include <stdio.h>

int main() {
      int n;
      printf("Masukkan sebuah angka: ");
      scanf("%d", &n);
      int i = 0;
      do{
            printf("tulisan ini dicetak saat i = %d\n", i);
            i++;
      }while(i < n);

      return 0;
}
```

Output

```
Masukkan sebuah angka: 3
tulisan ini dicetak saat i = 0
tulisan ini dicetak saat i = 1
tulisan ini dicetak saat i = 2
```

Output

```
Masukkan sebuah angka: 0
tulisan ini dicetak saat i = 0
```



Terdapat juga dua perintah yang dapat dipakai di looping, yaitu:

Source Code

```
break; //keluar dari pengulangan
continue; //melewati iterasi sekarang, melanjutkan di iterasi selanjutnya
```

Exercise 5

1. Apa perbedaan looping while dan looping do-while?
2. Apa yang terjadi jika perubahan pada looping tidak dapat mencapai kondisi looping yang false?
3. Apakah perubahan pada looping harus selalu increment/decrement (i++ atau i--)?
4. Kapan pengecekan kondisi looping: do-while dilakukan?
5. Apa perbedaan break dan continue?
6. Apakah bagian kondisi pada perulangan for dapat dikosongkan?
7. Apakah bagian deklarasi pada perulangan for dapat dikosongkan?
8. Apakah bagian perubahan pada perulangan for dapat dikosongkan?

Array

Array adalah variabel dengan satu nama, tetapi mengandung banyak nilai yang bertipe data sama. Array juga bisa disebut sebagai koleksi atau tabel nilai-nilai. Elemen-elemen array tersusun secara berurutan di memori. Penggunaan array adalah sebagai berikut:

Source Code

```
#include <stdio.h>

int main(){

    int arr[100];
    //format: tipe_data nama[ukuran];
    //mendeklarasikan array bernama arr yang dapat mengandung 100 nilai
bertipe data int
    arr[0] = 3;
    //nilai pertama di array diakses dengan indeks 0
    //memberikan nilai 3 di indeks pertama array tersebut
    arr[1] = 2;
    arr[2] = 100;
    printf("%d\n", arr[1] + arr[2]);
    return 0;
}
```

Output

```
102
```

Elemen-elemen di array di-akses menggunakan indeks. Di bahasa pemrograman C, indeks array dimulai dari 0. Jika kita memiliki array A yang di-deklarasikan seperti di bawah, berikut indeks dan nilainya:

Source Code

```
int A[10] = {8, 7, 4, 2, 1, 4, 6, 5, 3, 25};
```

Indeks	0	1	2	3	4	5	6	7	8	9
Nilai	8	7	4	2	1	4	6	5	3	25

Perhatikan bahwa pada tabel di atas, indeks hanya mencapai angka 9 ketika ukuran array tersebut 10. Karena kita mulai berhitung dari angka 0, maka nilai array terakhir yang bisa di-akses adalah pada indeks ukuran-1. Mengakses array dengan indeks < 0 atau lebih dari ukuran-1 dapat menyebabkan runtime error. Untuk itu, tentukan rentang indeks yang akan kalian gunakan saat deklarasi dengan tepat (sesuai kebutuhan).

Source Code

```
#include <stdio.h>

int main(){

	int tabel[5] = {2, 5, 3, 1, 4};
	for(int i = 4; i >= 0; i--){
		printf("%d ", tabel[i]);
	}

	return 0;
}
```

Output

```
4 1 3 5 2
```

Array juga bisa dibuat multi-dimensional, contohnya adalah sebagai berikut:

Source Code

```
//array 2 dimensi
int tabel[105][105];
int siswa[3][2] = {{0, 1}, {2, 3}, {3, 4}};

//array 3 dimensi
int tabel[105][105][105];
char apapun[105][105][105];
```

Setiap elemen pada array membutuhkan memori tergantung pada tipe data yang digunakan. Total memori yang dibutuhkan untuk sebuah array sama dengan ukurannya dikali ukuran memori satu elemennya. Sebagai contoh, array dengan 100 elemen bertipe data integer membutuhkan memori sebesar 100 x 4 byte = 400 byte.

Exercise 6

1. Dalam C, saat menginilsialisasi array, array dimulai dari index berapa?
2. Jika ingin mengakses array dengan index [100] maka berapa minimal ukuran array yang harus dibuat?
3. Jika ingin membuat array dengan tipe integer dengan ukuran 101, maka berapa ukuran memori yang akan dipakai?
4. Apa yang terjadi jika mengakses array di-luar limit?
5. Apakah bisa mengakses array dengan index negatif?

Strings

String sebenarnya merupakan penggunaan tipe data char secara berkelompok. Secara sederhana, string dapat diartikan sebagai array dari tipe data char. Deklarasi sebuah string di bahasa pemrograman C sama seperti cara mendeklarasikan sebuah array dari tipe data char. Berikut contohnya:

Source Code

```
#include <stdio.h>

int main() {

	char kalimat[100];
	//deklarasi string dengan nama kalimat dan panjang maksimum 99
	char menu[3][10] = {"Bakmie", "Ayam Goreng", "Steak"};
	//deklarasi array char multi-dimensi untuk menyimpan string secara
berkelompok

	char slogan[23] = "World Class University";
	printf("%s\n", slogan);
	printf("%c%c%c", menu[0][0], menu[1][2], menu[2][1]);
	return 0;
}
```

Output

```
World Class University
Bat
```

Perhatikan source code program diatas. Tertera di comment bahwa panjang maksimum string tersebut adalah 99 padahal array tersebut di-deklarasikan dengan ukuran 100. Mengapa demikian? Setiap string di bahasa pemrograman C berakhir dengan satu karakter khusus, \0, atau null terminating character. Karakter ini berfungsi untuk memberi tahu komputer dimana string tersebut berakhir. Contohnya seperti berikut:

Source Code

```
char kalimat[9] = "Surabaya";
```

Indeks	0	1	2	3	4	5	6	7	8
Karakter	S	u	r	a	b	a	y	a	\0

Cara input string dari console sedikit berbeda dengan tipe data lain. Pada tipe data string, simbol & sebelum nama variabel tidak diperlukan. Berikut contohnya:

Source Code

```
#include <stdio.h>

int main() {

	char input[1005];
	printf("Masukkan nama kalian: ");
	scanf("%s", input);
	printf("Nama kalian adalah: %s", input);
	return 0;
}
```

Output

```
Masukkan nama kalian: seponsbob
Nama kalian adalah: seponsbob
```

Output

```
Masukkan nama kalian: petrik setar
Nama kalian adalah: petrik
```

Perhatikan output kedua pada contoh di atas. Jika input berupa string yang mengandung spasi, maka scanf tidak akan mengambil seluruh string tersebut. Maka dari itu, diperlukan perlakuan khusus. Berikut contohnya:

Source Code

```
#include <stdio.h>

int main() {

	char input[1005];
	printf("Masukkan nama kalian: ");
	scanf("%[^\n]", input);
	//terus menerima input sampai program menemui karakter \n atau enter
	//Karakter \n tidak akan diambil
	printf("Nama kalian adalah: %s", input);
	return 0;
}
```

Output

```
Masukkan nama kalian: petrik setar
Nama kalian adalah: petrik setar
```

Ada beberapa fungsi bawaan dari bahasa pemrograman C untuk memanipulasi tipe data string, berikut adalah beberapa fungsi tersebut:

Fungsi	Penggunaan	Keterangan	Header File
strlen	`strlen(string)`	Memperoleh jumlah karakter dari suatu string	string.h
strcmp	`strcmp(string1, string2)`	Berfungsi untuk membandingkan dua buah string	
strcpy	`strcpy(string1, string2)`	Berfungsi untuk menyalin suatu string asal ke variabel string tujuan	
strcat	`strcat(string1, string2)`	Berfungsi untuk menambahkan string asal ke bagian akhir dari variabel string tujuan	
isupper	`isupper(string[0])`	Berfungsi untuk mengecek apakah karakter tersebut merupakan huruf kapital	ctype.h
islower	`islower(string[1])`	Berfungsi untuk mengecek apakah karakter tersebut merupakan huruf kecil	
isdigit	`isdigit(string[2])`	Berfungsi untuk mengecek apakah karakter tersebut merupakan sebuah angka	
isalpha	`isalpha(string[3])`	Berfungsi untuk mengecek apakah karakter tersebut merupakan huruf yang ada dalam alfabet	
tolower	`tolower(string[4])`	Berfungsi untuk merubah huruf kapital menjadi huruf kecil. (sepertinya tidak bisa dipakai di algo.binus)	
toupper	`toupper(string[5])`	Berfungsi untuk merubah huruf kecil menjadi huruf kapital. (sepertinya tidak bisa dipakai di algo.binus)	

ASCII

ASCII atau dikenal juga sebagai American Standard Code for Information Interchange merupakan standar internasional untuk kode huruf dan simbol. Setiap kode ASCII mengadung 8 bit dengan nilai maksimal 255. Sebagai contoh, kode ASCII 97 berarti huruf 'a'. Untuk selengkapnya bisa dilihat di tabel di samping.

	048 0	064 @	080 P	096 `	112 p
033 !	049 1	065 A	081 Q	097 a	113 q
034 "	050 2	066 B	082 R	098 b	114 r
035 #	051 3	067 C	083 S	099 c	115 s
036 $	052 4	068 D	084 T	100 d	116 t
037 %	053 5	069 E	085 U	101 e	117 u
038 &	054 6	070 F	086 V	102 f	118 v
039 '	055 7	071 G	087 W	103 g	119 w
040 (	056 8	072 H	088 X	104 h	120 x
041)	057 9	073 I	089 Y	105 i	121 y
042 *	058 :	074 J	090 Z	106 j	122 z
043 +	059 ;	075 K	091 [	107 k	123 {
044 ,	060 <	076 L	092 \	108 l	124 |
045 -	061 =	077 M	093]	109 m	125 }
046 .	062 >	078 N	094 ^	110 n	126 ~
047 /	063 ?	079 O	095 _	111 o	127 ⌂

Sisa kode ASCII yang tidak tercantum pada table lebih diperuntukan untuk symbol-simbol yang jarang digunakan. Apabila tertarik untuk mengetaui lebih lanjut mengenai sisanya dapat melakukan searching sendiri.

Untuk melihat kode ASCII sebuah symbol/ huruf yang berbentuk char, kita hanya perlu meng-outputkan char tersebut dalam format integer. Begitu sebaliknya apabila ingin menampilkan char menurut kode ASCII-nya.

Source Code

```
printf("%d\n", 'a');
printf("%c\n", 97);
```

Toupper & Tolower

Diketahui bahwa char 'a' memiliki kode ASCII 97 dan kita ingin mengubah char tersebut menjadi 'A' yang memiliki kode ASCII 65. Selisih dari kedua huruf tersebut adalah 32, sehingga kita hanya perlu menambahkan atau mengurangi sebanyak selisih tersebut ke char yang ingin diubah.

Source Code

```
char x = 'd';
char y = 'D';
printf("%c\n", x - ('a' - 'A')); //mengubah huruf kecil menjadi besar
printf("%c\n", y + ('a' - 'A')); //mengubah huruf besar menjadi kecil
```

Exercise 7

1. Apa yang terjadi jika kita memasukkan string dengan panjang 100 ke array dengan ukuran 100?
2. Apa yang terjadi jika kita memasukkan string dengan panjang 100 ke array dengan ukuran 99?
3. Apa yang terjadi jika kita memasukkan string dengan panjang 100 ke array dengan ukuran 101?
4. Apa output yang dihasilkan cuplikan program di bawah?

Source Code
```
printf("%c %c\n", 'c' - 32, 'C' + 32);
```

5. Apa output yang dihasilkan cuplikan program di bawah?

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

int main() {

	char kalimat[105] = "Hai\0Apa Kabar";
	printf("%s\n", kalimat);
	return 0;
}
```

6. Apa yang salah pada cuplikan program di bawah?

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

int main() {

	char input[1000];
	scanf("%s", &input);
	printf("%s %d\n", input[0], strlen(input));
	return 0;
}
```

Clean Code

Dari dua cuplikan kode dibawah, manakah yang lebih enak dilihat?

#1

Source Code

```
#include <stdio.h>
int main(){
      printf("Program menghitung faktorial\n");
long long int a, aa;
do{printf("Masukkan sebuah angka(1-10): ");
      scanf("%lld", &a);
      }
      while(a<1 || a>10);
          aa = 1;
      for(int i=1;i<=a;i++)
      aa = aa * i;
      printf("%lld faktorial adalah %lld", a, aa); return 0;
}
```

#2

Source Code

```
#include <stdio.h>

int main(){

      printf("Program menghitung faktorial\n");
      int n;
      do{
            printf("Masukkan sebuah angka(1-10): ");
            scanf("%d", &n);
      }while(n < 1 || n > 10);

      long long answer = 1;
      for(int i = 1; i <= n; i++){
            answer = answer * i;
      }
      printf("%d faktorial adalah %lld\n", n, answer);
      return 0;
}
```

Verdicts

Selama masa perkuliahan, kalian akan tidak asing dengan yang namanya online judge. Setiap program yang di-kumpul ke online judge akan mendapat salah satu dari respons berikut:

Respons	Keterangan
Accepted (AC)	Program berjalan dalam batasan waktu dan memori yang ditetapkan, serta setiap data uji yang diberikan menghasilkan keluaran yang sesuai (benar).
Wrong Answer (WA)	Program berjalan dalam batasan waktu dan memori yang ditetapkan, namun hasil keluaran program tidak sesuai dengan data uji yang diberikan (salah).
Runtime Error (RTE)	Program mengalami *crash* ketika dijalankan dengan data uji.
Time Limit Exceeded (TLE)	Program berjalan melebihi batas waktu yang ditetapkan sehingga program dihentikan secara paksa.
Memory Limit Exceeded (MLE)	Program berjalan melebihi batas memori yang ditetapkan sehingga program dihentikan secara paksa.
Compile Error (CE)	Program tidak dapat dikompilasi.

Code It Yourself!

Latihan sendiri di rumah dengan soal-soal yang tersedia di link di bawah ini.

http://bit.ly/latihankoding

FAQ

Q : “Itu jawabannya harus sama ya dengan outputnya?”

A : “Iya, karena perbedaan satu karakter output saja dengan test case juri akan mendapatkan verdict WRONG ANSWER.”

Q : “Codeku udah sama persis dengan sample tapi kok masih WA?”

A : “OUTPUT SAMA PERSIS DENGAN SAMPLE BELUM TENTU CODE ANDA BENAR. Baca kembali deskripsi soal, bisa jadi anda yang salah memahami maksud soal. Jika sudah yakin telah memahami soal, maka kemungkinan besar itu adalah Logical Error (cara berpikir anda yang masih salah). Untuk mendeteksi adanya logical error, biasakan jangan hanya memasukkan input dari sample, buatlah test-data sendiri. Kemungkinan lain penyebab WA adalah presentation error, dimana format output anda tidak sesuai dengan format yang diminta oleh soal.”

Q : “Apakah harus selalu memakai newline pada akhir output”?

A : “Iya, kecuali ada instruksi spesial dari soal.”

Q : “Kok bisa compiler error?”

A : “Mungkin anda salah memilih compiler pada online judge, atau bisa mencoba untuk compile terlebih dahulu sebelum submit.”

Q : “Kenapa saya bisa Runtime Error?”

A : “Runtime error disebabkan karena terjadi error pada saat program dijalankan, contohnya : Division by zero, mengakses index array diluar batas yang telah dideklarasikan.”

Q : “Kenapa saya terkena Time Limit Exceeded?”

A : “Cara anda kurang efisien, atau bisa saja terjadi infinite looping (looping tidak berhenti).”

Online Resources

- Atcoder.jp
- Codechef.com
- Codeforces.com
- Cplusplus.com
- Cppreference.com
- edx.org
- Geeksforgeeks.org
- Hackerrank.com
- Learn-c.org
- Onlinejudge.org
- Spoj.com
- Topcoder.com
- Udemy.com
- Vjudge.net
- And many more…