



Struktur Data

Pendidikan Teknik Informatika

Universitas Negeri Padang

TIM DOSEN ©2022

JOBSHEET (JS-01)

Array dan String

Fakultas	Proram Studi	Kode MK	Waktu
Teknik	Pendidikan Teknik Informatika	TIK1.61.2317	4 x 50 Menit

TUJUAN PRAKTIKUM

1. Mahasiswa mampu mengaplikasikan konsep array satu dimensi dalam menyelesaikan kasus.
2. Mahasiswa mampu mengaplikasikan konsep array dua dimensi dalam menyelesaikan kasus.
3. Mahasiswa mampu mengaplikasikan konsep string dalam menyelesaikan kasus.

HARDWARE & SOFTWARE

1. Personal Computer
2. IDE: Dev C++

TEORI SINGKAT

A. Array 1 Dimensi

Supaya mudah dipahami, perhatikanlah contoh kasus berikut: suatu universitas ingin mendata nilai mahasiswa di suatu kelas dengan banyak mahasiswa 10 orang. Setelah diinputkan, nilai-nilai tersebut ditampilkan ke layar. Untuk membuat program dengan ketentuan seperti diatas, ada beberapa cara untuk menyelesaiannya :

Program 1: tanpa menggunakan array

```
#include <stdio.h>
#include <conio.h>
void main()
{
    int n1,n2,n3,n4,n5,n6,n7,n8,n9,n10;
    clrscr();
    // Pembacaan semua nilai dari keyboard
    printf("Pemasukan data nilai mahasiswa : \n");
    printf("Nilai mahasiswa Ke-1 : ");
    scanf("%d",&n1);
    printf("Nilai mahasiswa Ke-2 : ");
    scanf("%d",&n2);
    printf("Nilai mahasiswa Ke-3 : ");
    scanf("%d",&n3);
    printf("Nilai mahasiswa Ke-4 : ");
    scanf("%d",&n4);
    printf("Nilai mahasiswa Ke-5 : ");
    scanf("%d",&n5);
    printf("Nilai mahasiswa Ke-6 : ");
    scanf("%d",&n6);
    printf("Nilai mahasiswa Ke-7 : ");
    scanf("%d",&n7);
    printf("Nilai mahasiswa Ke-8 : ");
    scanf("%d",&n8);
    printf("Nilai mahasiswa Ke-9 : ");
    scanf("%d",&n9);
    printf("Nilai mahasiswa Ke-10: ");
    scanf("%d",&n10);

    // Menampilkan data nilai yang telah dimasukan
    printf("Nilai mahasiswa Ke-1 : %d\n",n1);
    printf("Nilai mahasiswa Ke-2 : %d\n",n2);
    printf("Nilai mahasiswa Ke-3 : %d\n",n3);
    printf("Nilai mahasiswa Ke-4 : %d\n",n4);
    printf("Nilai mahasiswa Ke-5 : %d\n",n5);
    printf("Nilai mahasiswa Ke-6 : %d\n",n6);
    printf("Nilai mahasiswa Ke-7 : %d\n",n7);
    printf("Nilai mahasiswa Ke-8 : %d\n",n8);
    printf("Nilai mahasiswa Ke-9 : %d\n",n9);
    printf("Nilai mahasiswa Ke-10: %d\n",n10);
    getch();
}
```

Dengan menggunakan cara diatas, sebenarnya programnya telah mencukupi, tetapi kalau nilai yang akan diolah menjadi lebih banyak, maka pendeklarasian variabel n harus dilakukan sebanyak yang diperlukan. Jadi kalau data yang akan diolah sebanyak 100 buah, maka pendeklarasian dan pembacaan datanya pun dilakukan sebanyak 100 kali. Dan perhitungannya juga. Rumus perhitungan total pun menjadi berubah. Pemrograman di atas sebenarnya sederhana tetapi bisa sangat merepotkan.

Solusi dari permasalahan di atas adalah dengan menggunakan array. **Array adalah suatu variabel yang dapat menampung lebih dari satu data dengan tipe data yang sama dan dibedakan berdasarkan nomor indeksnya.** Dalam bahasa C, array selalu dimulai dari indeks ke-0 (nol). Bentuk deklarasi (pada kamus) sebuah variabel berbentuk array adalah:

```
tipe_data nama_variabel_array [ukuran_array];
```

Contoh :

```
int N[10];
```

Deklarasi diatas berarti pendeklarasian variabel array bernama N yang mempunyai elemen sebanyak 10 buah dengan indeks dimulai dengan nomor 0 sampai 9. Dalam memori deklarasi tersebut dapat digambarkan seperti berikut:

N[0]	N[1]	N[2]	N[3]	N[4]	N[5]	N[6]	N[7]	N[8]	N[9]
------	------	------	------	------	------	------	------	------	------

Untuk memasukan suatu elemen data dalam array, perintah yang dilakukan ditulis seperti pembacaan data variabel biasa (scanf()), hanya perbedaannya pada nama_variabel harus ditulis dengan jelas nama array-nya beserta nomer indeks.

Contohnya:

```
scanf ("%d", &N[3]);
```

Perintah diatas berarti pembacaan data dari keyboard untuk data bertipe integer ("%d") dan dimasukan ke variabel array indeks ke-3 (urutan ke-4). Contoh-contoh lain pengisian ke suatu elemen array:

```
I=5; // variabel I diisi dengan nilai 5  
N[I] = 7; // data ke-I dari variabel N diisi dengan nilai 7
```

Karena nomor elemen dari array bisa diisi dengan variabel, berarti kita bisa melakukan perulangan (loop) untuk melakukan pembacaan data dari elemen pertama sampai elemen terakhir.

Untuk menuliskan data yang ada dalam array ke layar, perintah printf() tetap digunakan, dengan menambahkan nomer indeks array pada nama variabel array yang akan dituliskan.

```
printf("Isi = %d", N[3]);
```

Perintah diatas berarti menuliskan data bertipe integer (%d) yang ada dalam variabel array indeks ke-3 (urutan ke-4) ke layar.

B. Array 2 Dimensi

Array 2 dimensi biasanya digunakan untuk menyimpan data dalam bentuk matrik. Index Array 2 dimensi terdiri dari index baris dan kolom. Pendeklarasian array 2 dimensi adalah :

```
tipedata namaarray [b] [k];
```

Dimana : b adalah banyak baris dan k adalah banyak kolom. Contoh:

```
int matrik[5][5];
```

Perintah di atas akan membuat sebuah array 2 dimensi yang kalau digambarkan adalah sebagai berikut :

index	0	1	2	3	4
0	0,0	0,1	0,2	0,3	0,4
1	1,0	1,1	1,2	1,3	1,4
2	2,0	2,1	2,2	2,3	2,4
3	3,0	3,1	3,2	3,3	3,4
4	4,0	4,1	4,2	4,3	3,3

Cara pengaksesan elemen array 2 dimensi dapat dilihat pada contoh di bawah ini:

```
mat[0][0]=7;  
printf("Masukan data : ");scanf("%d",&mat[2][1]);  
printf("Data yang dimasukan : %d\n",mat[2][1]);
```

Keterangan :

1. Baris pertama adalah mengisikan nilai 7 ke array mat pada baris 0 kolom 0.
2. Baris kedua adalah perintah untuk membaca data elemen matrik pada baris 2 kolom ke 1.
3. Baris ketiga adalah perintah untuk menampilkan data elemen matrik/array pada baris 2 dan kolom ke-1.

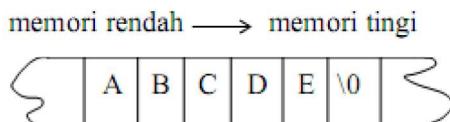
C. String

1. Konstanta dan Variabel String

String merupakan bentuk data yang biasa dipakai dalam bahasa pemrograman untuk keperluan menampung dan memanipulasi data teks, misalnya untuk menampung (menyimpan) suatu kalimat. **Pada bahasa C, string bukanlah merupakan tipe data tersendiri, melainkan hanyalah kumpulan dari nilai-nilai karakter yang berurutan dalam bentuk array berdimensi satu.**

1.1 Konstanta String

Suatu konstanta string ditulis dengan diawali dan diakhiri tanda petik ganda, misalnya: “ABCDE”. Nilai string ini disimpan dalam memori secara berurutan dengan komposisi sebagai berikut:



Gambar 1. Komposisi penyimpanan string dalam memori

Setiap karakter akan menempati memori sebesar 1 byte. Byte terakhir otomatis akan berisi karakter NULL (\0). Dengan mengetahui bahwa suatu string diakhiri nilai NULL, maka akhir dari nilai suatu string dapat dideteksi. Sebagai sebuah array karakter, karakter pertama dari nilai string mempunyai indeks ke-0. Karakter kedua mempunyai indeks ke-, dan seterusnya.

1.2 Variabel String

Variabel string adalah variabel yang dipakai untuk menyimpan nilai string. Misalnya:

```
char name [15];
```

merupakan instruksi untuk mendeklarasikan variabel string dengan panjang maksimal 15 karakter (termasuk NULL). Deklarasi tersebut sebenarnya tidak lain merupakan deklarasi array bertipe *char*.

2. Inisialisasi String

Suatu variabel string dapat diinisialisasi seperti halnya array yang lain.

Namun elemen terakhirnya harus berupa karakter NULL. Contoh:

```
char name [] = { 'E', 'L', 'K', 'A', '\0' };
```

yang menyatakan bahwa **name** adalah variabel string dengan nilai awal berupa string : “ELKA”. Bentuk inisialisasi yang lebih singkat:

```
char name [] = "ELKA";
```

Pada bentuk ini, karakter NULL tidak pernah ditulis. Secara implisit akan disisip oleh kompiler. Perlu diperhatikan, bila **name** dideklarasikan sebagai string, penugasan (*assignment*) suatu string ke variabel string seperti:

```
name = "ELKA";
```

adalah **tidak diperkenankan**.

3. Input Output Data String

3.1 Memasukkan Data String

Pemasukan data string ke dalam suatu variabel biasa dilakukan dengan fungsi *gets()* atau *scanf()*. Bentuk umum pemakaianya adalah sebagai berikut:

```
#include <stdio.h>
gets (nama_array);
```

atau

```
#include <stdio.h>
scanf ("%s", nama_array);
```

Perhatikan :

- a. **nama_array** adalah variabel bertipe *array of char* yang akan digunakan untuk menyimpan sting masukan.
- b. Di depan **nama_array** tidak perlu ada operator & (operator alamat), karena **nama_array** tanpa kurung siku sudah menyatakan alamat yang ditempati oleh elemen pertama dari array tersebut.
- c. Jika menggunakan *scanf()*, data string masukan tidak boleh ada spasi.

3.2 Menampilkan Isi Variabel String

Untuk menampilkan isi variabel string, fungsi yang digunakan adalah *puts()* atau *printf()*. Bentuk umum pemakaiannya adalah sebagai berikut:

```
#include <stdio.h>
puts (var_string);
```

atau

```
printf ("%s", var_string);
```

Dalam hal ini **var_string** adalah sebuah variabel yang berupa sebuah *array of char*. Fungsi *puts()* akan menampilkan isi dari **var_string** dan secara otomatis menambahkan karakter ‘\n’ di akhir string. Sedangkan fungsi *printf()* akan menampilkan isi variabel string tanpa memberikan tambahan ‘\n’. Sehingga, agar kedua pernyataan di atas memberikan keluaran yang sama, maka pernyataan *printf()* diubah menjadi:

```
printf ("%s\n", var_string);
```

4. Mengakses Elemen String

Variabel string merupakan bentuk khusus dari array bertipe *char*. Oleh karena itu, elemen dari variabel string dapat diakses seperti halnya pengaksesan elemen pada array. Contoh program mengakses elemen string adalah cara mengakses elemen array untuk menghitung total karakter dari string yang dimasukkan melalui keyboard.

5. Fungsi-Fungsi Mengenai String

Berikut ini akan dibahas beberapa fungsi pustaka yang umumnya disediakan oleh kompiler C untuk mengoperasikan suatu nilai string. Fungsi-fungsi pustaka untuk operasi string, prototype-prototype nya berada di file header **string.h**. beberapa diantara fungsi pustaka untuk operasi string adalah sebagai berikut:

5.1 Fungsi *strcpy()* untuk Menyalin Nilai String

Bentuk pemakaian :

```
#include <string.h>
strcpy (tujuan, asal)
```

Fungsi ini digunakan untuk menyalin string **asal** ke variabel string **tujuan** termasuk karakter ‘\0’. Keluaran dari fungsi ini (*return value*) adalah string **tujuan**. Dalam hal ini, variabel **tujuan** harus mempunyai ukuran yang dapat digunakan untuk menampung seluruh karakter dari string **asal**.

5.2 Fungsi *strlen()* untuk Mengetahui Panjang Nilai String

Bentuk pemakaian:

```
#include <string.h>
strlen(var_string);
```

Fungsi ini digunakan untuk memperoleh banyaknya karakter di dalam string yang menjadi argumen (**var_string**). Keluaran dari fungsi ini adalah panjang dari **var_string**. karakter NULL tidak ikut dihitung.

5.3 Fungsi *strcat()* untuk Menggabung Nilai String

Bentuk pemakaian:

```
#include <string.h>
strcat(tujuan, sumber);
```

Menggabungkan dua buah nilai string tidak dapat dilakukan dengan operator ‘+’, karena operator ini bukan operator untuk operasi string. Penggabungan dua buah nilai string dapat dilakukan dengan fungsi pustaka *strcat* dengan menambahkan string **sumber** ke bagian akhir dari string **tujuan**. Keluaran dari fungsi ini adalah string **tujuan**.

5.4 Fungsi *strcmp()* untuk Membandingkan Dua Nilai String

Bentuk pemakaian:

```
#include <string.h>
strcmp(Str1, Str2);
```

Fungsi ini digunakan untuk membandingkan string **str1** dengan string **str2**. Keluaran dari fungsi ini bertipe *int* yang berupa nilai”

- a. – 1, jika **str1** kurang dari **str2**

- b. 0, jika **srt1** sama dengan **str2**
- c. 1, jika **srt1** lebih dari **str2**

Pembandingan dilakukan untuk karakter pada posisi yang sama dari **str1** dan **str2**, dimulai dari karakter terkiri. Acuan pembandingan dari dua buah karakter didasarkan oleh nilai ASCII-nya. Misal, karakter ‘A’ lebih kecil daripada karakter ‘B’ dan karakter ‘B’ lebih kecil daripada karakter ‘C’.

5.5 Fungsi *strchr()* untuk Mencari Nilai Karakter dalam String

Bentuk pemakaian:

```
#include <string.h>
strchr (var_string, kar);
```

Fungsi ini dapat digunakan untuk mencari suatu nilai karakter yang berada dalam suatu nilai string. dalam hal ini adalah mencari karakter **kar** dalam string **var_string**. keluaran dari fungsi ini adalah alamat posisi dari karakter pertama pada nilai string, yang sama dengan karakter yang dicari. Jika karakter yang dicari tidak ada dalam string, maka fungsi ini akan memberikan hasil nilai kosong (*null*).

LATIHAN

1. Program Input Output Array

```
/* Nama File : array1
Pembuat : tuliskan nama dan NIM anda
Tgl pembuatan : tuliskan tanggal hari ini
Deskripsi : input dan output array */

#include <stdio.h>
#include <conio.h>

void main()
{
int Nilai[10];
int indeks;
float total,ratarata;
clrscr();
// Pembacaan data dari keyboard
printf("Pembacaan data nilai \n");
```

```
for (indeks=0;indeks<10;indeks++)
{
printf("Nilai mahasiswa ke-%d = ",indeks+1);
scanf("%d",&Nilai[indeks]);
}
// Menampilkan data yang telah dimasukan dan rata-rata.
printf("\n\nMenampilkan Data Nilai \n");
for (indeks=0;indeks<10;indeks++)
printf("Nilai mahasiswa ke-%d = %3d\n",
indeks+1,Nilai[indeks]);
getch();
}
```

2. Program Proses Isi Array (Jumlah Dan Rata-Rata)

```
/* Nama File : array2
Pembuat : tuliskan nama dan NIM anda
Tgl pembuatan : tuliskan tanggal hari ini
Deskripsi : menghitung total dan rata-rata nilai
10 orang mahasiswa yang disimpan dalam array */

#include <stdio.h>
#include <conio.h>

void main()
{
    int Nilai[10];
    int indeks;
    float total,ratarata;
    clrscr();
    // Pembacaan data dari keyboard
    printf("Pembacaan data nilai \n");
    for (indeks=0;indeks<10;indeks++)
    {
        printf("Nilai mahasiswa ke-%d = ",indeks+1);
        scanf("%d",&Nilai[indeks]);
    }
    // Perhitungan total dan rata-rata total=0;
    for (indeks=0;indeks<10;indeks++)
        total=total+Nilai[indeks];
    //endfor
    ratarata=total/10.0;

    // Menampilkan data yang telah dimasukan dan rata-
    rata. printf("\n\nMenampilkan Data Nilai \n");
    for (indeks=0;indeks<10;indeks++)
```

```

        printf("Nilai mahasiswa ke-%d = %3d\n",
               indeks+1,Nilai[indeks]);
    printf("\nTotal      = %0.2f\n",total);
    printf("\nRata-rata = %0.2f\n",ratarata);
    getch();
}

```

3. Mengisi dan menampilkan isi array dua dimensi (matriks)

```

/* Nama File  : matriks1
Pembuat      : tuliskan nama dan NIM anda
Tgl pembuatan : tuliskan tanggal hari ini
Deskripsi     : menuliskan isi matriks */

#include <stdio.h>
#include <conio.h>

void main()
{
int A[3][3],i,j;
clrscr();
A[0][0] = 3; A[0][1] = 7; A[0][2] = 5;
A[1][0] = 4; A[1][1] = 2; A[1][2] = 1;
A[2][0] = 8; A[2][1] = 3; A[2][2] = 1;

for (i=0;i<=2;i++)
{
for(j=0;j<=2;j++)
printf("%d",A[i][j]);
printf("\n");
}
getch();
}

```

4. Menjumlahkan isi elemen matriks (perbaris)

```

/* Nama File  : matriks2
Pembuat      : tuliskan nama dan NIM anda
Tgl pembuatan : tuliskan tanggal hari ini
Deskripsi     : menjumlahkan isi matriks tiap baris*/

#include <stdio.h>
#include <conio.h>

void main()
{

```

```
int A[3][3],i,j,jum;
clrscr();
printf("Masukkan angka untuk matriks ukuran 3 x 3 \n");
for (i=0;i<3;i++)
{
for(j=0;j<3;j++)
scanf("%d",&A[i][j]);
}

printf("\nMatriks anda adalah : \n");
for (i=0;i<=2;i++)
{
jum=0;
for(j=0;j<=2;j++)
{
jum=jum+A[i][j];
printf("%d",A[i][j]);
}
printf("\t= %d",jum);
printf("\n");
}
getch();
```

5. Input data string

```
/* Nama File  : yourname
Pembuat      : tuliskan nama dan NIM anda
Tgl pembuatan : tuliskan tanggal hari ini
Deskripsi     : memasukkan data string dari keyboard */

#include <stdio.h>

main ()
{
    char name [15];

    printf ("Masukkan nama Anda: ");
    //gets (name);
    scanf ("%s", name);

    printf ("\nHalo, %s. Selamat belajar string/", name);
}
```

6. Inisialisasi string

```
/* Nama File : inistr
Pembuat      : tuliskan nama dan NIM anda
Tgl pembuatan : tuliskan tanggal hari ini
Deskripsi     : inisialisasi string */

#include <stdio.h>

void bentuk1 (void);
void bentuk2 (void);

main ()
{
    bentuk1 ();
    bentuk2 ();
}

void bentuk1 (void)
{
    char kompiler_c[]=
{'V','i','s','u','a','l',' ', 'C','+', '+','\0'};

    puts (kompile_c);
}

void bentuk2 (void)
{
    char kompile_c[] = "Visula C++";
    printf ("%s\n", kompile_c);
}
```

7. Memperoleh panjang suatu string

```
/* Nama File : panjangstr
Pembuat      : tuliskan nama dan NIM anda
Tgl pembuatan : tuliskan tanggal hari ini
Deskripsi     : memperoleh panjang suatu string */

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

main ()
{
    char salam [] = "Hallo";
    printf("Panjang string = %d karakter\n",
    strlen(salam));
}
```

8. Menggabungkan isi string1 dengan string2

```
/* Nama File : gabungstr
Pembuat : tuliskan nama dan NIM anda
Tgl pembuatan : tuliskan tanggal hari ini
Deskripsi : menggabungkan isi string1 dengan string2
*/
#include <stdio.h>
#include <string.h>
#define PJG 15

main ()
{
    char str1 [PJG], str2[PJG];

    strcpy (str1, "bukit"); //str1 diisi "bukit"
    strcpy (str2, "tinggi"); //str2 diisi "tinggi"

    strcat(str1, str2); //tambahkan str2 ke akhir str1

    printf ("str1 --> %s str2-->%s\n", str1, str2);
}
```

9. Menghitung banyaknya karakter dari suatu string

```
/* Nama File : hitkar
Pembuat : tuliskan nama dan NIM anda
Tgl pembuatan : tuliskan tanggal hari ini
Deskripsi : menghitung banyaknya karakter dari suatu
string */
#include <stdio.h>
#define MAKS 256

main ()
{
    int i, jumkar = 0;
    char teks [MAKS];

    puts ("Masukkan suatu kalimat (maks 255 karakter).");
    puts ("Saya akan menghitung jumlah karakter.\n");
    fgets (teks, sizeof teks, stdin); //masukkan dari
    keyboard
```

```

        for (i=0; teks [i]; i++)
            jumkar++;

        printf ("\nJumlah karakter = %d\n", jumkar);
    }

```

TUGAS

1. Buatlah sebuah program yang menentukan dan menghitung banyaknya isi array yang bilangan ganjil dan bilangan genap.

Contoh :

array A : 1 2 3 4 5 6 7 8 9 10, maka :

Genap : 2 4 6 8 10 banyak=5

Ganjil : 1 3 5 7 9 banyak =5

2. Buatlah sebuah program yang melakukan pengurangan dua buah matriks.
3. Ketikkan sebuah kalimat melalui keyboard dengan menggunakan *gets ()* (atau *fgets ()*) kemudian didapatkan keluaran berupa laporan tentang jumlah huruf kecil dan huruf kapital dalam kalimat tersebut.

DAFTAR PUSTAKA

Kelley, Al and Pohl, Ira. 2003. *C by Dissection: The Essentials of C Programming*. Addison-Weasley.

Khannedy, Eko Kurniawan. 2007. *Diktat Pemrograman C*. Bandung : Unikom.

Liem, Ingriani. 2003. *Catatan Singkat Bahasa C*. Jurusan Teknik Informatika. Institut Teknologi Bandung.

Reema Thareja. 2014. *Data Structures Using C*. India : Oxford University Press.

Solichin, Achmad. 2003. *Pemrograman Bahasa C dengan Turbo C*. Kuliah Berseri ilmukomputer.com