Bab 4

Array dan Fungsi

4.1 Array

4.1.1 Definisi Array

Array adalah variabel khusus yang dapat menyimpan lebih dari satu nilai dari variabel dengan tipe data yang sama.

4.1.2 Deklarasi Array

Deklarasi dari array menggunakan sintaks berikut:

```
/* deklarasi dari sebuah array yang berisi 10 integer */
int numbers[10];
```

4.1.3 Mengakses Array

Untuk mengakses bilangan yang ada pada array dilakukan dengan sintaks yang sama. Per- hatikan bahwa array didalam C mulai dari 0, sehingga jika dideklarasikan ukuran arraynya 10, berarti elemen array dimulai dari indeks ke 0 sampai 9.

```
int numbers[10];

numbers[0] = 10;
numbers[1] = 20;
numbers[2] = 30;
numbers[3] = 40;
numbers[4] = 50;
numbers[5] = 60;
numbers[6] = 70;

// menampilkan elemen ke 7 dari array, berarti digunakan indeks 6 //
printf("Elemen ke 7 dari array adalah %d",numbers[6]);
```

Array hanya dapat terdiri dari satu tipe varibel saja, karena array sebenamya mengimplemen- tasikan sekuen dari nilai didalam memori Computer. Karena itu, mengakses elemen array secara acak sangatlah efisien (tidak mengurangi performa dari program).

4.1.4 Dimensi Array

- **Array Satu Dimensi**

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

```
int arr[5];
arr[0] = 4;
arr[1] = 2;
arr[2] = 3;
```

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 multidimensi 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 berikut:

		j					
		0	1	2	3	4	5
i	0	?	?	?	?	?	?
	1	?	?	?	?	200	?
	2	?	?	?	100	?	?
	3	?	?	?	?	?	?
	4	?	?	?	?	?	?

matriks[i][j]

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

4.2 String

4.2.1 Definisi String

String di dalam bahasa C sebenarnya adalah array dari karakter. Dalam definisi string di bahasa C, ada beberapa yang menggunakan pointer. Walaupun pointer akan dibahas lebih lanjut di bab berikutnya, dalam bab ini tetap digunakan pointer untuk menunjukkan array dari karakter sebagai representasi dari string. Dapat ditulis dengan sintaks sebagai berikut:

```
char* name= "Bung Karno";
```

Metode ini membuat sebuah string yang hanya dapat digunakan untuk dibaca, tidak dimo- difikasi. Jika kita ingin membuat sebuah string yang dapat dimodifikasi, deklarasi string dapatmenggunakan array dari karakter, seperti sintaks berikut:

```
char name[]="Bung Karno";
```

Sintaks tersebut berbeda dari sintaks sebelumnya, karena pada sintaks tersebut dialokasikan secara statis array dari variabel, sehingga bisa kita manipulasi. Kurung siku kosong[] menyatakan bahwa ukuran dari array otomatis akan dihitung oleh compiler.

```
char name[] = "Bung Karno";
//sama dengan//
char name[11] = "Bung Karno" ;
```

Dalam hal ini, penulisan ukuran array yang eksplisit juga akan menghasilkan tipe array yang sama. Perhatikan ukuran yang ditulis adalah jumlah karakter ditambah satu.

Alasan penambahan satu itu, walaupun panjang dari string Bung Karno adalah 10 karakter, adalah untuk terminasi dari string (karakter spesial dengan nilai 0) yang menandakan akhir dari string.

Input dalam string bisa dilakukan menggunakan scanf seperti berikut:

```
char str[10];
scanf("%s", str);
```

Untuk outputnya tetap menggunakan printf seperti berikut.

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

4.2.2 String formating

Kita dapat menggunakan fungsi printf untuk memformat string dengan string yang lain. Berikut contohnya:

```
char * name=”Bung Karno”;
int age = 27;

/* menampilkan ‘ Bung Karno berumur 27 tahun.’*/
printf(“%s berumur %d tahun.\n”, name, age);
```

Perhatikan bahwa saat menampilkan string, kita bisa menambahkan karakter newline (\n) yang menandakan pindah ke baris baru.

4.3 Fungsi

4.3.1 Pengertian Fungsi

Fungsi adalah suatu bagian dari program yang dimaksudkan untuk mengerjakan suatu tugas tertentu dan letaknya dipisahkan dari bagian program yang menggunakannya.

Berdasarkan keluaran dan masukannya fungsi dapat dibagi menjadi 4. Yang dimaksud keluaran adalah fungsi yang mengembalikan suatu nilai tertentu dan yang dimaksud dengan masukan adalah parameter yang merupakan inputan pada fungsi.

1. Fungsi tanpa keluaran dan masukan Contoh: custom_function ();
2. Fungsi tanpa keluaran dan dengan masukan Contoh: printf("Hello”);
3. Fungsi dengan keluaran dan tanpa masukan Contoh: time ();
4. Fungsi dengan keluaran dan masukan Contoh: atoi(str);

4.3.2 Pendefinisain Fungsi

Sebelum fungsi dapat digunakan dan bisa dipanggil, perlu dilakukan pendefinisan terlebih dahulu. Pendefinisian ditujukan untuk mendefinisikan apa yang fungsi tersebut lakukan ketika fungsi tersebut dipanggil. Berikut adalah sintaks untuk melakukan pendefinisan fungsi.

```
<return_type> <nama_fungsi>(<parameter1>, <parameter2>, ...)
{
    statement;
    statement;
    ...
    ...
    ...
}
```

Berikut adalah contoh fungsi untuk mencetak string **"Ini Fungsi."**

```
void cetak()
{
    printf("Ini Fungsi\n");
}
```

4.3.3 Prototipe Fungsi

Selain menggunakan pendefisian langsung seperti cara sebelumnya, fungsi juga dapat dibuat dengan prototipe. Prototipe Fungsi (atau biasa disebut Interface Fungsi) adalah deklarasi dari sebuah fungsi tanpa definisinya. Deklarasi sebuah fungsi berisi return type, nama fungsi, dan parameter yang terlibat. Untuk menuliskan prototipe fungsi, sintaksnya sebagai berikut.

```
// Deklarasi
<return_type> <nama_fungsi>(<parameter1>, <parameter2>, ...);
```

Contoh kode program menggunakan prototipe fungsi:

```
// Prototipe Fungsi
void cetak();

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

// Definisi Fungsi cetak()
void cetak()
{
    printf("Ini Fungsi\n");
}
```

4.3.4 Fungsi Tanpa Parameter

Fungsi ini merupakan fungsi yang tidak memiliki parameter inputan namun fungsi ini tetap melaksanakan tugas tertentu. Bentuk umumnya adalah:

```
voidnama_fungsi()
{
   <pernyataan>
}
```

Sebagai contoh penggunaan fungsi tanpa masukan dan keluaran, ketikkan kode program berikut dan jalankan melalui Codeblock.

```c
#include <stdio.h>

char nama[50], nrp[20];

void input_data()
{
	//Fungsi Pertama
	printf("Nama: ");
	scanf("%s",&nama);
	printf("NRP: ");
	scanf("%s",&nrp);
}

void cetak_string()
{
	printf("Ini adalah fungsi yang mencetak string\n ");
	printf("Silahkan masukkan data\n");
	input_data();
}

int main(int argc, char** argv)
{
	cetak_string();
	return 0;
}
```

4.3.5 Fungsi Dengan Parameter

Fungsi ini merupakan fungsi yang memiliki sejumlah parameter dengan tugas tertentu. Perlu digaris bawahi bahwa semua parameter bersifat independen dan tidak akan berubah meskipun nilainya diubah di dalam fungsi. Bentuk umumnya adalah:

```c
void nama_fungsi(tipe_data argumen1, tipe_data argumen2 ...)
{
  <pernyataan>
}
```

Jika ingin perubahan nilai argument pada fungsi juga merubah nilai dari variabel tersebut maka dibutuhkan tanda (*) untuk memberi tanda bahwa nilai akan diparsing by reference oleh Compiler. Bentuk umumnya adalah sebagai berikut:

```c
void nama_fungsi(tipe_data*argumen1, tipe_data*argumen2, ...)
{
  <pernyataan>
}
```

Sebagai contoh penggunaan fungsi dengan masukan dan tanpa keluaran, ketikkan kode program berikut dan jalankan melalui Codeblock.

```
#include <stdio.h>

void cetak_string(char str1[], char str2[])
{
	printf("Nama mahasiswa adalah %s\n", str1);
	printf("NRP mahasiswa adalh %s\n", str2);
}
void hitung(int a, int b)
{
	printf("Hasil penjumlahan %d + %d adalah %d\n", a, b,
a+b);
}
int main(int agrc, char** argv)
{
	char nama[50], nrp[20];
	int a, b;
	printf("Nama: ");  scanf("%s",&nama);
	printf("NRP: ");   scanf("%s",&nrp);
	cetak_string(nama, nrp);

	a = 10; b = 12;
	hitung(a,b); return 0;
}
```

Sebagai contoh penggunaan fungsi dengan masukan dan tanpa keluaran dengan menggunakan parsing parameter by reference, ketikkan kode program berikut dan jalankan melalui Codeblock.

4.3.6 Fungsi Dengan Keluaran

Fungsi dengan keluaran merupakan suatu fungsi yang mengembalikan suatu nilai tertentu kepada pemanggil fungsi. Bentuk umumnya adalah

```
<tipe_data>nama_fungsi(tipe_data argumen1, tipe_data
argumen2, ...)
{
  <pernyataan>
  return<nilai
  >;
}
```

Sebagai contoh penggunaan fungsi dengan keluaran, ketikkan kode program berikut dan ja- lankan melalui Codeblock

```
#include <stdio.h>

void cetak_string(char str1[], char str2[])
{
	printf("Nama mahasiswa adalah %s\n", str1);
	printf("NRP mahasiswa adalh %s\n", str2);
}
void hitung(int a, int b)
{
	printf("Hasil penjumlahan %d + %d adalah %d\n", a, b,
a+b);
}
int main(int agrc, char** argv)
{
	char nama[50], nrp[20];
	int a, b;
	printf("Nama: ");  scanf("%s",&nama);
	printf("NRP: ");   scanf("%s",&nrp);
	cetak_string(nama, nrp);

	a = 10; b = 12;
	hitung(a,b); return 0;
}
```

4.3.7 Fungsi Rekursif

Fungsi rekursif adalah fungsi yang memanggil dirinya sendiri secara berulang - ulang. Karena prosesnya dilakukan secara berulang - ulang maka harus ada kondisi yang mengakhiri prosesnya. Sebagai contoh penggunaan fungsi rekursif, ketikkan kode program berikut dan jalankan melalui Codeblock.

```
#include <stdio.h>

void faktorial(int n)
{
	if (n <=1)
	  return 1;
	else
	  return n*faktorial(n-1);
}

int main(int agrc, char** argv)
{
	int n;
	printf("Masukkan integer: "); scanf("%d", &n);
	printf("Faktorial dari %d adalah %d\n ", n, faktorial(n));

	return 0;
}
```