MODUL 2: TIPE DATA DAN ARRAY PADA C

1.1. Deskripsi Singkat

Secara umum, struktur data dikelompokkan menjadi dua kategori, yaitu struktur data primitif (sederhana) dan struktur data non primitif (majemuk). Struktur data primitif adalah tipe data dasar *(primitive data types)* yang didukung oleh bahasa pemrograman (*built-in*), misalnya Integer, floating point, characters, pointer, dan-lain-lain. Struktur data non primitif misalnya arrays, structure, stack, queues, linked lists, dan lain-lain. Struktur data non-primitif diturunkan dari struktur data primitif. Oleh karena itu, pemahaman mengenai tipe data sangat penting untuk memahami dan mengimplementasikan struktur data. Dalam praktikum kita ini akan memahami tipe data dasar dan Array dalam Bahasa Pemrograman C serta mempraktekkan penggunaannya.

1.2. Tujuan Praktikum

Setelah praktikum pada modul 2 ini diharapkan mahasiswa mampu:

1) Memahami tipe data dasar dalam Bahasa Pemrograman C
2) Mendeklarasikan deklarasi data, serta mengoperasikan dan menampilkannya dalam format yang sesuai
3) Menggunakan array, khususnya array multidimensi, dalam penyimpanan data

1.3. Material Praktikum

Kegiatan pada modul 2 ini memerlukan material berupa software editor dan compiler (atau IDE) untuk bahasa pemrograman C.

1.4. Kegiatan Praktikum

A. Tipe Data

Dalam bahasa C terdapat beberapa jenis tipe data yang bisa digunakan untuk mendeklarasikan sebuah variabel, konstansta, atau fungsi pada program. Jenis variabel menentukan berapa banyak ruang yang ditempati dalam penyimpanan dan bagaimana pola bit yang disimpan ditafsirkan. Variabel biasa digunakan di dalam program dengan tujuan untuk menampung data. Nilai yang terdapat pada variabel sewaktu-waktu dapat diubah. Jumlah variabel yang dibuat dapat tidak terbatas, namun masing-masing variabel tersebut harus bersifat unik dan tidak boleh ada nama variabel yang sama. Agar dapat digunakan, sebelumnya variabel dideklarasikan dahulu berserta tipe data yang akan disimpan dalam variabel tersebut. variabel dideklarasikan dengan format:

```
tipe_data nama_variabel
```

Adapun aturan-aturan penamaan variabel dalam bahasa C adalah sebagai berikut :

1. Harus diawali dengan huruf (tidak boleh diawali dengan angka), selanjutnya dapat diikuti dengan angka atau underscore (_).
2. Tidak boleh mengandung spasi, simbol, atau karakter khusus lain selain huruf, angka, dan underscore.
3. Bersifat case sensitive.
4. Tidak boleh memakai kata kunci (keywords) yang telah digunakan oleh Bahasa C.

Selain variabel terdapat konstanta yang juga dapat menampung data. Hanya saja dalam kostanta nilai yang ada tidak dapat diubah atau bernilai pasti. Konstanta dideklarasikan dengan menggunakan preprocessor **#define**:

```
#define nama_konstanta
```

atau dengan dengan singkatan **const:**

```
const tipe_data dan nama_konstanta.
```

Dalam pembuatan fungsi, tipe data digunakan untuk mendeklarasikan tipe data kembalian dari fungsi. Berikut format dasar cara penulisan fungsi dalam bahasa C:

```
tipeDataKembalian namaFunction() {
  // Isi function disini...
  // Isi function disini...
  return nilai;
}
```

Jika suatu fungsi tidak mengembalikan nilai, **tipeDataKembalian** ditulis sebagai **void**. Sebuah fungsi yang tidak mengembalikan nilai kadang disebut juga sebagai **procedure**.

C menyediakan lima macam tipe data dasar, yaitu:

- tipe data integer (nilai numerik bulat, yang dideklarasikan dengan **int**)
- floating-point (nilai numerik pecahan ketetapan tunggal, yang dideklarasikan dengan **float**)
- double-precision (nilai numerik pecahan ketetapan ganda, yang dideklarasikan dengan **double**)
- karakter (dideklarasikan dengan **char**)
- **void** (tidak bertipe, tidak menyimpan apapun, biasanya untuk tipe fungsi yang tidak return value apapun)

1. a. Berikut cara mendeklarasikan variable menggunakan **int**. Ketik ulang kode program berikut ke dalam Code::Blocks atau IDE C lainnya yang Anda gunakan.

```
#include <stdio.h>
int main() {
    int tanggal = 17;
    printf("Tanggal %i", tanggal);
    return 0;
}
```

Simpan potongan program tersebut dengan nama **praktikum2_1a.c**. Jalankan program tersebut. Output yang didapatkan adalah sebagai berikut:

```
Tanggal 17
```

b. Kita juga bisa mendeklarasikan dua variable sekaligus jika tipe datanya sama. Modifikasi program **praktikum2_1a.c** menjadi seperti berikut ini.

```
#include <stdio.h>
int main() {
    int tanggal = 17;
    int bulan = 8;
    printf("Tanggal %i, Bulan %d", tanggal, bulan);
    return 0;
}
```

Simpan ulang dan jalankan program. Hasilnya adalah sebagai berikut:

```
Tanggal 17, Bulan 8
```

c. Bisa kita perhatikan, pada variabel tanggal kita menggunakan format specifier **%i** sedangkan pada variabel bulan kita menggunakan format specifier **%d**. Pada **printf()** tidak ada perbedaan antara keduanya. Namun ketika menggunakan **scanf()** dan menginput nilai dengan 0 didepan angka (011, 012, 013), maka akan terlihat perbedaannya. Pada **%d** diasumsikan sebagai basis 10 (desimal) sedangkan **%i** diasumsikan sebagai basis 8 (oktal). Silakan dicoba, dengan mengetikkan ulang potongan program di bawah ini, lalu simpan dengan nama **praktikum2_1c.c**.

```
#include <stdio.h>

int main()
{
```

```
    int a, b;
    printf("Angka pertama: ");
    scanf("%d", &a);
    printf("Angka kedua: ");
    scanf("%i", &b);
    printf("\nNilai desimal dari angka pertama adalah: %i",
a);
    printf("\nNilai oktal dari angka kedua adalah: %i", b);

    return 0;
}
```

Jika kita berikan **011** sebagai input, maka ouput yang didapatkan adalah sebagai berikut:

```
Angka pertama: 011
Angka kedua: 011

Nilai desimal dari angka pertama adalah: 11
Nilai oktal dari angka kedua adalah: 9
```

2. Jika kita menggunakan **float** atau **double** untuk bilangan desimal, double akan dua kali lebih lebih teliti daripada float. Float memiliki ketepatan 7 digit desimal, sedangkan double memiliki ketepatan hingga 15 digit desimal. Contoh kode penggunaan kedua tipe data ini adalah sebagai berikut:

```
#include <stdio.h>

void pifloat()
{
   float pi = 22.0/7.0;
   printf("=== FLOAT ===\n");
   printf("%.15lf\n", pi);
   printf("%lf\n", pi);
}

void pidouble()
{
   double pi = 22.0/7.0;
```

```
    printf("=== DOUBLE ===\n");
    printf("%.15lf\n", pi);
    printf("%lf\n", pi);
}

int main()
{
    pifloat();
    printf("\n");
    pidouble();

    return 0;
}
```

Ketik ulang program tersebut, lalu simpan dengan nama **praktikum2_2.c**. Bila dijalankan, outputnya adalah:

```
=== FLOAT ===
3.142857074737549
3.142857

=== DOUBLE ===
3.142857142857143
3.142857
```

Bisa kita lihat, ada perbedaan antara penggunaan float dan double pada desimal ke-tujuh.

3. Keyword **char** digunakan untuk mendeklarasikan tipe data yang memuat nilai berupa karakter, contohnya adalah sebagai berikut:

```
#include <stdio.h>

int main() {
    char huruf = 'Y';

    printf("%c", huruf);

return 0;
}
```

Ketik ulang program di atas, lalu simpan dengan nama **praktikum2_3a.c.** Output:

```
Y
```

Dengan menggunakan kode diatas hanya akan mengeluarkan satu huruf saja. Untuk menyimpan kalimat atau **string**, maka harus kita ubah terlebih dahulu kodenya. Simpan potongan program di bawah ini dengan nama **praktikum2_3b.c**.

```
#include <stdio.h>

int main() {
    char kalimat[10] = "AnbiDev";
    printf("%s", kalimat);

return 0;
}
```

Output yang dihasilkan:

```
AnbiDev
```

Terlihat perbedaan pada kode pada **praktikum2_3a.c** dan **praktikum2_3b.c**. Pada **praktikum2_3a.c** kita menggunakan tanda petik satu untuk nilainya dan menggunakan format specifier **%c**. Pada **praktikum2_3b.c**, kita menggunakan tanda petik dua dan menggunakan format specifier **%s** sebagai ganti **%c**.

B. Array

Array merupakan suatu tipe data yang terstruktur dan dapat digunakan untuk menyimpan data yang memiliki tipe data yang sama. Array biasa juga disebut larik. Penggunaan array dapat mengurangi kerumitan dalam proses penyimpanan data dalam jumlah yang besar. Jika kita hendak menyimpan variabel nama yang berjumlah sepuluh nama, kita tidak harus membuat 10 variabel untuk nama tersebut. Dengan menggunakan array kita tidak perlu membuat banyak variabel untuk data yang memiliki tipe yang sama. Selain itu, dalam alokasi memori penyimpanan, tipe data array melakukan pemesanan tempat terlebih dahulu sesuai dengan kebutuhan yang ada.

Terdapat 2 jenis array, yaitu array 1 dimensi dan array 2 dimensi. array dengan 1 dimensi merupakan array yang dapat digambarkan sebagai sebuah baris. Dalam array 1 dimensi, elemen yang ada di dalamnya dapat diakses hanya dengan menggunakan 1 indeks saja. Sedangkan array 2 dimensi merupakan array yang dapat digambarkan seperti sebuah matrik. Selain itu elemen yang ada dalam array 2 dimensi dapat diakses dengan menggunakan 2 indeks, yaitu indeks kolom dan juga indeks baris. Berikut format pendeklarasian array:

```
tipe_data nama_array[jumlah_elemen];
```

Contoh:

```
int Number[10];
```

Eelemen pada array dapat diinisialisasi sebagai berikut:

```
int Number[10]={1,3,2,4,5,7,6,9,8,0};
```

4. Salinlah program di bawah ini untuk mempraktrekkan pendeklarasian dan pengaksesan array 1 dimensi. Simpan dengan nama **praktikum2_4.c**, lalu jalankan.

```
#include<stdio.h>
int main()
{
   char vokal[5];
   int i;
   vokal[0]='A';
   vokal[1]='I';
   vokal[2]='U';
   vokal[3]='E';
   vokal[4]='O';

   for(i=0;i<5;i++)
   {
      printf("%c\n", vokal[i]);
   }
   return 0;
}
```

Setelah program dijalankan, diketahui bahwa pendeklarasian array dapat dilakukan dengan langsung mengisikan nilai dari tiap indeksnya.

5. Contoh dibawah ini adalah penggunaan array untuk menyimpan sejumlah angka dan menghitung rata-ratanya. Ketik program pada IDE C Anda, lalu simpan dengan nama **praktikum2_5.c.**

```
#include <stdio.h>
int main(void)
{
int numbers[10];
int count = 10;
```

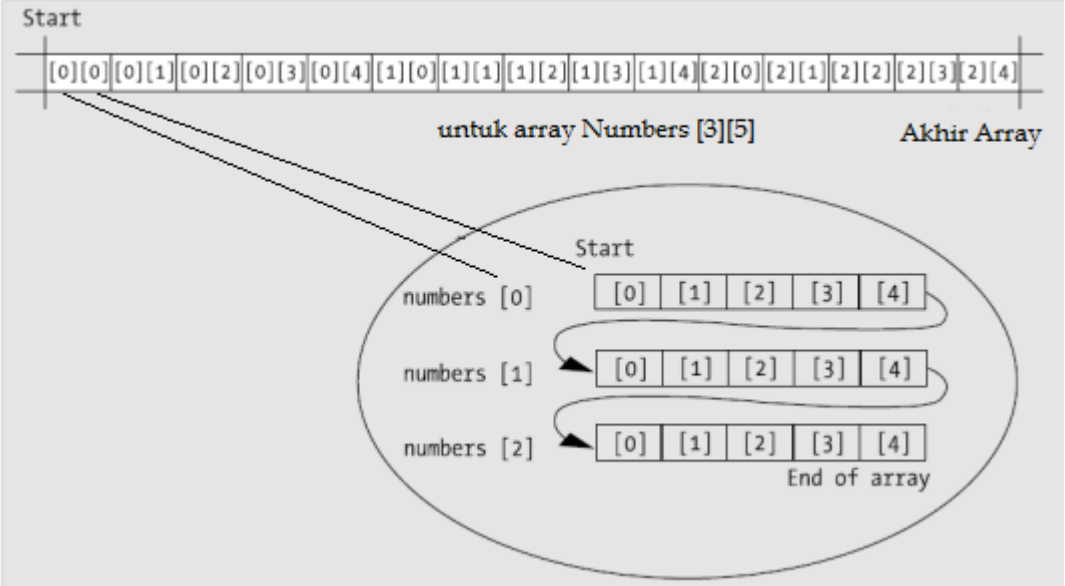
```
long sum = 0L;
float average = 0.0f;
printf("\n Masukkanlah 10 Angka:\n");
for(int i = 0; i < count; i ++)
{
printf("%2d> ",i+1);
scanf("%d", &numbers[i]); /* Read a number */
sum += numbers[i]; /* Jumlahkan setiap elemen */
}
average = (float)sum/count; /* Hitung rata-rata */
printf("\n Rata-rata dari sepuluh Angka yang dimasukkan:
%f\n", average);
return 0;
}
```

Jika program dijalankan, hasilnya adalah:

```
Masukkanlah 10 Angka:
1> 450
2> 765
3> 562
4> 700
5> 598
6> 635
7> 501
8> 720
9> 689
10> 527
Rata-rata dari sepuluh Angka yang dimasukkan: 614.700000
```

Pada contoh sebelumnya, array yang digunakan adalah array 1 dimensi, yang dicirikan dengan menggunakan 1 tanda “[ ]”. Array multi dimensi akan menggunakan lebih dari satu “[ ]” untu kmenyatakan elemen yang ada pada variabel tersebut. Contoh:

```
float Numbers[3][5];
```



Contoh inisiasi array multidimensi seperti yang ditampilkan dibawah ini:

```
int numbers[3][4] = {
                      { 10, 20, 30, 40 },
                      { 15, 25, 35, 45 },
                      { 47, 48, 49, 50 }
                    };

int numbers[2][3][4] = {
                         {
                            { 10, 20, 30, 40 },
                            { 15, 25, 35, 45 },
                            { 47, 48, 49, 50 }
                         },
                         {
                            { 10, 20, 30, 40 },
                            { 15, 25, 35, 45 },
                            { 47, 48, 49, 50 }
                         }
                       };
```

6. Contoh di bawah ini menunjukkan penggunaan array 2 dimensi dalam hal penjumlahan dua matriks.

```
#include <stdio.h>
#define ROW 2
#define COL 3

int main(void)
{
int Matriks_A[ROW][COL]={{4,2,3},{5,7,6}};
```

```
int Matriks_B[ROW][COL]={{1,8,9},{3,5,4}};
int Matriks_C[ROW][COL];

printf("Matriks_C adalah : \n");
for(int i = 0; i < ROW; i++)
{
for(int j = 0; j < COL; j++)
{
Matriks_C[i][j] = Matriks_A[i][j]+ Matriks_B[i][j];
}
printf("%d %d %d\n", Matriks_C[i][0], Matriks_C[i][1],
Matriks_C[i][2]);
}
return 0;
}
```

Simpan program tersebut dengan nama **prektikum2_6.c.** Cobalah untuk memodifikasi program tersebut untuk penjumlahan matriks berukuran lainnya.

1.5. Penugasan

1. **`sizeof()`** adalah sebuah operator untuk mengetahui jumlah memori (byte) yang diperlukan oleh suatu tipe data pada bahasa C. Gunakan **`sizeof()`** untuk mengetahui ukuran memori pada berbagai tipe data pada bahasa pemrograman C, seperti **char, int, float, double.** Catat hasil yang Anda dapatkan. Bandingkan dengan hasil yang didapatkan oleh teman-teman Anda. Diskusikan apakah yang menyebabkan hasil yang didapatkan berbeda-beda.

2. Ketik program berikut pada IDE Anda, lalu simpan.

```
/* Aturan Scope pada Bahasa C */
#include<stdio.h>

int main()
{
  {
    int x = 10, y = 20;
    {
       printf("x = %d, y = %d\n", x, y);
```

```
        {
           int y = 40;
           x++;
           y++;
           printf("x = %d, y = %d\n", x, y);
        }
        printf("x = %d, y = %d\n", x, y);
     }
   }
 return 0;
}
```

Amati output yang dihasilkan. Jelaskan mengapa bisa terjadi perubahan demikian pada nilai `x` maupun `y`.

3. Buatlah program untuk menginput nilai elemen-elemen Matriks A berukuran 3x4 dan mencetak Matriks A tersebut. Contoh ouput sebagai berikut.

```
1 3 4 5
2 4 6 8
3 5 7 9
```