

LAPORAN FUNGSI & STRUKTUR



Oleh:

Nama : L HAFIDL ALKHAIR
NIM : 2023903430060
Kelas : TRKJ 1.C
Jurusan : Teknologi Informasi dan Komputer
Program Studi : Teknologi Rekayasa Komputer Jaringan
Dosen Pembimbing : Indrawati, SST. MT



**JURUSAN TEKNOLOGI, KOMPUTER, DAN INFORMASI
PRODI TEKNOLOGI REKAYASA KOMPUTER DAN JARINGAN
POLITEKNIK NEGERI LHOKEUMAWA
TAHUN AJARAN 2023/2024**

LEMBAR PENGESAHAN

No. Praktikum : 06 /TIK/TRKJ-1C/ Data Structure And Algorithms Practice

Judul : Laporan Fungsi dan Struktur

Nama : L HAFIDL ALKHAIR

NIM : 2023903430060

Kelas : TRKJ-1C

Jurusan : Teknologi Informasi Dan Komputer

Prodi : Teknologi Rekayasa Komputer Jaringan

Tanggal Praktikum : 16 Oktober 2023

Tanggal Penyerahan : 23 Oktober 2023

Buketrata, 23 Oktober 2023

Dosen Pembimbing,

Indrawati, SST.MT

Nip. 19740815 200112 2 001

A. Tujuan

- Memahami fungsi dan struktur bahasa C membantu mahasiswa memahami dasar-dasar pemrograman, termasuk konsep variabel, tipe data, operator, dan kontrol alur program.
- Mahasiswa belajar tentang deklarasi dan pemanggilan fungsi, serta konsep pengembalian nilai dari fungsi dan pengiriman argumen ke dalam fungsi.
- Mahasiswa mempelajari struktur kontrol seperti percabangan (if-else statements) dan pengulangan (loops) dalam bahasa C. Ini membantu mereka memahami cara program membuat keputusan dan mengulangi tugas tertentu berdasarkan kondisi-kondisi tertentu
- Mahasiswa memahami struktur data dasar seperti array dan struktur dalam bahasa C. Ini penting karena memungkinkan mereka menyimpan dan mengelola data secara efisien dalam program mereka.

B. Dasar Teori

Pengertian Fungsi:

Fungsi adalah blok kode yang dapat dipanggil oleh program untuk menjalankan tugas tertentu. Fungsi memiliki nama, tipe kembalian, dan daftar parameter (jika ada). Mereka dapat digunakan untuk mengelompokkan dan mengatur kode, serta memungkinkan pemrogram untuk menjalankan tugas tertentu berulang kali.

Contoh sederhana fungsi dalam Bahasa C:

```
1 int tambah(int a, int b) {  
2     return a + b;  
3 }
```

Pengertian Struktur

Pengertian struktur adalah kumpulan variabel dengan tipe data yang berbeda yang digabungkan dalam satu unit. Ini memungkinkan programmer untuk membuat entitas data kompleks yang dapat mengandung data dengan jenis yang berbeda.

```
[*] Untitled1 |
1 struct Mahasiswa {
2     char nama[100];
3     int usia;
4     long int nomorIdentitas;
5     float nilai;
6 };
```

Dalam definisi ini, `struct` adalah kata kunci yang digunakan untuk mendefinisikan struktur baru dengan nama "Mahasiswa". Struktur ini memiliki empat variabel anggota: `nama` dengan tipe data char array, `usia` dengan tipe data integer, `nomorIdentitas` dengan tipe data long integer, dan `nilai` dengan tipe data float.

C. Alat dan Bahan

Di dalam pembuatan sebuah Bahasa pemrograman tentu saja memerlukan alat dan bahan-bahan supaya program tersebut dapat di jalankan dengan baik dan tidak mengalami yang namanya error saat program atau kode tersebut di jalankan. Adapun alat dan bahan sebagai berikut:

1. Laptop atau device



2. Tools atau aplikasi pemrograman (disini kita memakai aplikasi Dev-C++)



D. Program

A. Fungsi

A. Program Pertama (fungsi 1)

```
1  #include <stdio.h>
2
3  void cetak_pesan(void);
4
5  int main()
6  {
7      int i;
8
9      for ( i = 1; i<=5; i++) {
10         printf("Pesan ke-%d : ", i);
11         cetak_pesan();
12     }
13 }
14
15 void cetak_pesan()
16 {
17     printf("ini dulu! baru itu!\n\n");
18 }
```

Hasil Output

```
Pesan ke-1 : ini dulu! baru itu!  
Pesan ke-2 : ini dulu! baru itu!  
Pesan ke-3 : ini dulu! baru itu!  
Pesan ke-4 : ini dulu! baru itu!  
Pesan ke-5 : ini dulu! baru itu!  
-----
```

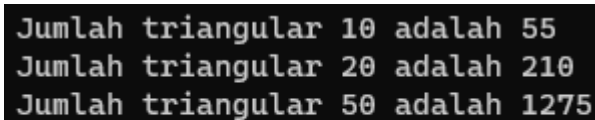
Analisa Program :

- Header File: Program ini menggunakan header file `stdio.h` yang berisi fungsi-fungsi standar input/output dalam bahasa C.
- Fungsi `cetak_pesan()`: Fungsi ini didefinisikan di bawah fungsi `main()`. Fungsi ini bertujuan mencetak pesan "ini dulu! baru itu!\n\n" ke layar. \n digunakan untuk membuat baris baru. Fungsi ini dideklarasikan di awal program menggunakan prototipe `void cetak_pesan(void);`.
- Fungsi `main()`: Fungsi `main()` adalah titik awal eksekusi program. Di dalamnya terdapat loop `for` yang akan berjalan dari 1 hingga 5 (inklusif). Pada setiap iterasi loop, pesan "Pesan ke-%d : " dicetak bersama nilai `i`, kemudian fungsi `cetak_pesan()` dipanggil.
- Loop `for`: Loop ini berjalan sebanyak lima kali karena variabel `i` dimulai dari 1 dan berakhir ketika `i` kurang dari atau sama dengan 5. Selama setiap iterasi loop, pesan "Pesan ke-%d : " dicetak, diikuti oleh pesan "ini dulu! baru itu!\n\n" dari fungsi `cetak_pesan()`.

B. Program kedua (fungsi 2)

```
1  #include <stdio.h>
2
3  void hitung_triangular(int n);
4
5  int main() {
6      hitung_triangular(10);
7      hitung_triangular(20);
8      hitung_triangular(50);
9      return 0;
10 }
11
12 void hitung_triangular(int n) {
13     int i, jumlah = 0;
14     for (i = 1; i <= n; ++i) {
15         jumlah = jumlah + i;
16     }
17     printf("Jumlah triangular %d adalah %d\n", n, jumlah);
18 }
```

Hasil Output



```
Jumlah triangular 10 adalah 55
Jumlah triangular 20 adalah 210
Jumlah triangular 50 adalah 1275
-----
```

Analisa Program :

- Header File: Program ini menggunakan header file `stdio.h` yang berisi fungsi-fungsi standar input/output dalam bahasa C.
- Fungsi `hitung_triangular(int n)`: Fungsi ini didefinisikan untuk menghitung jumlah bilangan triangular hingga angka `n`. Bilangan triangular adalah deret bilangan yang ditemukan dengan menambahkan semua bilangan bulat dari 1 hingga `n`. Fungsi ini memiliki satu parameter `n` yang menentukan batas atas dari deret bilangan triangular yang akan dihitung. Di dalam fungsi ini, terdapat loop `for` yang menjumlahkan semua bilangan bulat dari 1 hingga `n`. Hasil penjumlahan disimpan dalam variabel `jumlah`. Setelah loop selesai, fungsi mencetak hasilnya menggunakan `printf` dengan format "Jumlah triangular %d adalah %d\n".

- Fungsi main(): Fungsi main() adalah titik awal eksekusi program. Di dalamnya, fungsi hitung_triangular() dipanggil tiga kali dengan argumen 10, 20, dan 50.

C. Program ketiga (fungsi 3)

```

1  #include <stdio.h>
2  #include <conio.h>
3
4  void fpb(int, int);
5  void main()
6  {
7      fpb(150,35);
8      fpb(1026,405);
9      fpb(83,240);
10     //gecth();
11 }
12 void fpb(int u, int v)
13 {
14     int tampung;
15     printf("FPB dari %d dan %d adalah ",u,v);
16     while(v != 0){
17         tampung = u % v ;
18         u = v;
19         v = tampung;
20     }
21     printf("%d\n",u);
22 }

```

Hasil program

```

FPB dari 150 dan 35 adalah 5
FPB dari 1026 dan 405 adalah 27
FPB dari 83 dan 240 adalah 1
-----

```

Analisa Program :

- Header File: Program ini menggunakan dua header file, yaitu stdio.h untuk fungsi input/output standar dan conio.h untuk fungsi-fungsi pengontrolan konsol. Perlu diperhatikan bahwa header file conio.h tidak standar dan mungkin tidak didukung oleh semua kompiler.

- Fungsi `fpb(int u, int v)`: Fungsi ini didefinisikan untuk mencari FPB dari dua bilangan bulat `u` dan `v`. Fungsi ini menggunakan algoritma Euclidean, di mana dua bilangan tersebut dibagi terus-menerus sampai ditemukan sisa yang nol. Algoritma ini memanfaatkan sifat bahwa FPB dari dua bilangan sama dengan FPB dari bilangan yang lebih kecil dan sisa pembagian bilangan yang lebih besar dengan bilangan yang lebih kecil. Hasil FPB akhirnya adalah bilangan yang lebih kecil ketika sisa pembagian menjadi nol. Hasil FPB dicetak ke layar.
- Fungsi `main()`: Fungsi `main()` adalah titik awal eksekusi program. Di dalamnya, fungsi `fpb()` dipanggil tiga kali dengan argumen yang berbeda untuk menghitung FPB dari pasangan bilangan yang berbeda.
- Loop `while`: Di dalam fungsi `fpb()`, terdapat loop `while` yang terus berjalan sampai `v` menjadi 0. Di setiap iterasi, nilai `u` dan `v` diperbarui berdasarkan sisa pembagian, dan proses ini diulang sampai sisa pembagian menjadi 0. Saat sisa pembagian adalah 0, nilai `u` adalah FPB dari dua bilangan yang diinputkan.

D. Program Keempat (Fungsi 4)

```
1  #include <stdio.h>
2  #include <conio.h>
3
4  int fpb(int u, int v);
5
6  int main() {
7      int hasil;
8
9      hasil = fpb(150, 35);
10     printf("FPB dari 150 dan 35 adalah %d\n", hasil);
11
12     hasil = fpb(1026, 405);
13     printf("FPB dari 1026 dan 405 adalah %d\n", hasil);
14
15     printf("FPB dari 83 dan 240 adalah %d\n", fpb(83,240));
16 }
17
18 int fpb(int u, int v)
19 {
20     int tampung;
21     while (v != 0) {
22         tampung = u % v;
23         u = v;
24         v = tampung;
25     }
26     return(u);
27 }
```

Hasil Output

```
FPB dari 150 dan 35 adalah 5
FPB dari 1026 dan 405 adalah 27
FPB dari 83 dan 240 adalah 1
-----
```

Analisa Program :

- Header File: Program ini menggunakan dua header file, yaitu `stdio.h` untuk fungsi input/output standar dan `conio.h` untuk fungsi-fungsi pengontrolan konsol. Perlu dicatat bahwa header file `conio.h` tidak standar dan mungkin tidak didukung oleh semua kompiler.
- Fungsi `fpb(int u, int v)`: Fungsi ini didefinisikan di bagian bawah program. Fungsi ini mengambil dua parameter `u` dan `v` yang merupakan dua bilangan

yang akan dicari FPB-nya. Di dalam fungsi, terdapat loop while yang terus berjalan sampai v menjadi 0. Di setiap iterasi, nilai u dan v diperbarui berdasarkan sisa pembagian, dan proses ini diulang sampai sisa pembagian menjadi 0. Saat sisa pembagian adalah 0, nilai u adalah FPB dari dua bilangan tersebut. Fungsi mengembalikan nilai FPB tersebut.

- Fungsi main(): Fungsi main() adalah titik awal eksekusi program. Di dalamnya, fungsi fpb() dipanggil tiga kali dengan berbagai pasangan bilangan. Hasil dari pemanggilan fungsi tersebut disimpan dalam variabel hasil dan dicetak ke layar menggunakan printf().

E. Program Kelima (Fungsi 5).

```
1  #include <stdio.h>
2  #include <math.h>
3  #include <conio.h>
4
5  float absolut(float x) {
6      if (x < 0)
7          x = -x;
8      return x;
9  }
10
11 float akar(float x) {
12     float epsilon = 0.00001;
13     float guess = 1.0;
14
15     if (x < 0) {
16         printf("Argumen Negative!\n");
17         return -1.0;
18     }
19
20     while (absolut(guess * guess - x) >= epsilon)
21         guess = (x / guess + guess) / 2.0;
22
23     return guess;
24 }
25
26 int main() {
27     printf("Akar (4.0)      = %f\n", akar(4.0));
28     printf("Akar (625.0)   = %f\n", akar(625.0));
29     printf("Akar (-39.5)   = %f\n", akar(-39.5));
30     getch();
31 }
```

Hasil Output

```
Akar (4.0)      = 2.000000  
Akar (625.0)    = 25.000000  
Argumen Negative!  
Akar (-39.5)    = -1.000000
```

Analisa Program

- Fungsi “absolut” mengembalikan nilai absolut dari bilangan real x dengan cara mengubahnya menjadi positif jika x kurang dari nol.
- Fungsi "akar" menghitung akar kuadrat dari bilangan real x menggunakan metode Newton-Raphson.
- Jika x kurang dari nol, program mencetak pesan "Argumen Negative!" dan mengembalikan -1.0 sebagai indikasi kesalahan.
- Program akan terus melakukan iterasi metode Newton-Raphson hingga selisih antara perkiraan akar kuadrat dan nilai sebenarnya kurang dari epsilon (dalam hal ini, 0.00001).
- Fungsi “main” adalah fungsi utama program.
Ini memanggil fungsi akar untuk menghitung akar kuadrat dari beberapa bilangan dan mencetak hasilnya.
- Terakhir, program menggunakan getch() untuk menahan jendela konsol agar tidak langsung tertutup setelah menampilkan hasil. Ini adalah fungsi yang mungkin khusus untuk beberapa lingkungan kompilasi.

F. Program Keenam (Fungsi 6).

```
1  #include <stdio.h>
2  #include <conio.h>
3
4  float minimum(float x, float y) {
5      if (x < y) {
6          return x;
7      } else {
8          return y;
9      }
10 }
11 int main() {
12     float a, b;
13
14     printf("Masukkan nilai a: ");
15     scanf("%g", &a);
16     printf("Masukkan nilai b: ");
17     scanf("%g", &b);
18
19     printf("\nBilangan terkecil antara %g dan %g adalah ", a, b);
20     printf("%g\n\n", minimum(a, b));
21
22     getch();
23 }
```

Hasil Output :

```
Masukkan nilai a: 8
Masukkan nilai b: 2

Bilangan terkecil antara 8 dan 2 adalah 2
```

Analisa :

- Fungsi "minimum" mengambil dua parameter bertipe float, x dan y, dan mengembalikan nilai terkecil di antara keduanya.
- Jika x kurang dari y, maka x dikembalikan; jika tidak, y dikembalikan.
- Fungsi main adalah fungsi utama program.
- Dua variabel a dan b bertipe float digunakan untuk menyimpan input dari pengguna.

- Pengguna diminta memasukkan dua nilai float.
- Fungsi minimum kemudian dipanggil dengan dua nilai yang dimasukkan pengguna sebagai argumen, dan hasilnya dicetak menggunakan printf.
- Program menggunakan getch() untuk menahan jendela konsol sehingga tidak langsung tertutup setelah menampilkan hasil.

G. Program Ketujuh (Fungsi 7).

```

1  #include <stdio.h>
2
3  void demo(void)
4  {
5      auto int var_auto = 0;
6      static int var_static = 0;
7
8      printf("auto = %d, static = %d\n", var_auto, var_static);
9      ++var_auto;
10     ++var_static;
11 }
12
13 int main()
14 {
15     int i = 0;
16     while (i < 3 ) {
17         demo();
18         i++;
19     }
20 }

```

Hasil Ouput

```

auto = 0, static = 0
auto = 0, static = 1
auto = 0, static = 2
-----

```

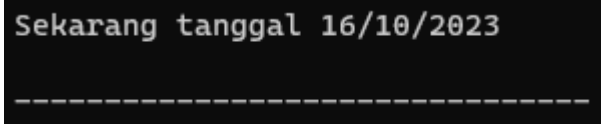
Analisa :

- Fungsi "demo" adalah fungsi void tanpa parameter.
fungsi ini memiliki 2 variable lokal :
- var_auto: Variabel dengan penyimpanan otomatis (auto). Variabel ini akan ditempatkan dalam stack memory dan nilainya akan diinisialisasi ulang setiap kali fungsi dipanggil.
- var_static: Variabel dengan penyimpanan statis (static). Variabel ini akan ditempatkan dalam data segment memory dan nilainya akan disimpan di antara pemanggilan fungsi.
- terdapat loop while yang akan memanggil fungsi demo sebanyak 3 kali.
- Setiap kali demo dipanggil, variabel var_auto akan diinisialisasi ulang menjadi 0 karena merupakan variabel lokal dengan penyimpanan otomatis. Sementara itu, variabel var_static akan menyimpan nilai sebelumnya di antara pemanggilan fungsi karena memiliki penyimpanan statis.

H. Program Kedelapan (Struktur 1)

```
1  #include <stdio.h>
2
3  struct tanggal {
4      int tgl;
5      int bulan;
6      int tahun;
7  };
8
9  int main() {
10     struct tanggal now;
11
12     now.tgl = 16;
13     now.bulan = 10;
14     now.tahun = 2023;
15
16     printf("Sekarang tanggal ");
17     printf("%d/%d/%d\n", now.tgl, now.bulan, now.tahun);
18
19     return(0);
20 }
```

Hasil Output



```
Sekarang tanggal 16/10/2023
-----
```

Analisa

Program ini menggunakan struktur (struct) dalam bahasa pemrograman C untuk menyimpan data tanggal, bulan, dan tahun. Dalam program ini, sebuah struct bernama tanggal didefinisikan, yang memiliki tiga variabel anggota: tgl, bulan, dan tahun.

- Program ini mendeklarasikan sebuah struktur dengan nama tanggal, yang memiliki tiga variabel anggota bertipe data integer: tgl, bulan, dan tahun
- Variabel now dari tipe struct tanggal digunakan untuk menyimpan tanggal saat ini. Nilai-nilai tgl, bulan, dan tahun diisi dengan 16, 10, dan 2023 secara berturut-turut.
- Kemudian, program mencetak pesan "Sekarang tanggal " diikuti oleh nilai-nilai tgl, bulan, dan tahun yang disimpan dalam variabel now.

I. Program Kesembilan (Struktur 2)

```
1  #include <stdio.h>
2  #include <conio.h>
3
4  struct tanggal {
5      int tgl;
6      int bln;
7      int thn;
8  };
9
10 void cetak_tgl(struct tanggal now) {
11     static char *bulan[] = {
12         "Kode bulan salah",
13         "Januari", "Februari", "Maret", "April", "Mei", "Juni",
14         "Juli", "Agustus", "September", "Oktober", "November", "Desember"
15     };
16
17     printf("\nTanggal sekarang adalah: ");
18     printf("%d %s %d\n", now.tgl, bulan[now.bln], now.thn);
19 }
20
21 int main() {
22     struct tanggal skr;
23
24     printf("Masukkan tanggal hari ini (tgl/bulan/tahun): ");
25     scanf("%d/%d/%d", &skr.tgl, &skr.bln, &skr.thn);
26
27     cetak_tgl(skr);
28
29     getch();
30     return 0;
31 }
```

Hasil Output

```
Masukkan tanggal hari ini (tgl/bulan/tahun): 16/10/2023
Tanggal sekarang adalah: 16 Oktober 2023
-----
```

Analisa

Program ini meminta pengguna memasukkan tanggal, bulan, dan tahun, kemudian mencetak tanggal tersebut menggunakan fungsi cetak_tgl.

- Membuat struktur tanggal dengan tiga anggota: tgl (tanggal), bln (bulan), dan thn (tahun).
- Fungsi ini mencetak tanggal yang diberikan dalam format "dd bulan yyyy", menggunakan indeks bulan untuk mengambil nama bulan dari array bulan.
- Fungsi main() membuat variabel struktur tanggal bernama skr.
- Meminta pengguna memasukkan tanggal, bulan, dan tahun menggunakan scanf().
- Memanggil fungsi cetak_tgl() untuk mencetak tanggal yang dimasukkan pengguna.
- Menggunakan getch() untuk menahan tampilan layar sampai pengguna menekan sebuah tombol.
- Mengembalikan nilai 0 untuk menandakan bahwa program telah berakhir dengan sukses

J. Program Kesepuluh (Struktur 3)

```
1  #include <stdio.h>
2  #include <string.h>
3  #include <conio.h>
4
5  struct tanggal {
6      int tgl;
7      int bln;
8      int tahun;
9  };
10
11 struct orang {
12     char nama[30];
13     struct tanggal lahir;
14 };
15
16 struct orang siswa;
17
18 int main() {
19     strcpy(siswa.nama, "L HAFIDL ALKHAIR");
20     siswa.lahir.tgl = 27;
21     siswa.lahir.bln = 01;
22     siswa.lahir.tahun = 2006;
23     printf("Nama      : %s\n", siswa.nama);
24     printf("Tanggal    : %d-%d-%d\n",
25           siswa.lahir.tgl, siswa.lahir.bln, siswa.lahir.tahun);
26     getch();
27 }
```

Hasil Output

```
Nama      : L HAFIDL ALKHAIR
Tanggal    : 27-1-2006
-----
```

Analisa

- Dua struktur data didefinisikan: tanggal yang memiliki tiga anggota (tgl, bln, tahun) untuk menyimpan tanggal, bulan, dan tahun, dan orang yang memiliki dua anggota (nama dan struktur tanggal untuk tanggal lahir).
- Sebuah variabel struktur siswa dari jenis orang dideklarasikan. Ini akan digunakan untuk menyimpan data siswa.

- Informasi tentang siswa (nama dan tanggal lahir) diinisialisasi dalam fungsi main() menggunakan fungsi strcpy() untuk menyalin string nama ke variabel siswa.nama, dan kemudian nilai-nilai untuk siswa.lahir.tgl, siswa.lahir.bln, dan siswa.lahir.tahun diatur.
- Informasi yang telah diinisialisasi dicetak ke layar menggunakan printf(). Nama siswa dan tanggal lahirnya dicetak ke layar dengan format tertentu.
- Program menggunakan fungsi getch() dari <conio.h> untuk menunggu sampai pengguna menekan tombol sebelum menutup jendela program. Namun, perlu diingat bahwa <conio.h> adalah bagian dari beberapa kompiler C/C++ kuno dan mungkin tidak didukung oleh kompiler modern.

K. Program Kesebelas (Struktur 4)

```

1  #include <stdio.h>
2  #include <string.h>
3  #include <conio.h>
4
5  struct bulan {
6      int jumlah;
7      char nama[4];
8  };
9
10 int main() {
11     int i;
12     struct bulan bulan[12] = {
13         {31, {'J', 'a', 'n', '\0'}}, {28, {'F', 'e', 'b', '\0'}},
14         {31, {'M', 'a', 'r', '\0'}}, {30, {'A', 'p', 'r', '\0'}},
15         {31, {'M', 'e', 'i', '\0'}}, {30, {'J', 'u', 'n', '\0'}},
16         {31, {'J', 'u', 'l', '\0'}}, {31, {'A', 'g', 's', '\0'}},
17         {30, {'S', 'e', 'p', '\0'}}, {31, {'O', 'k', 't', '\0'}},
18         {30, {'N', 'o', 'v', '\0'}}, {31, {'D', 'e', 's', '\0'}}
19     };
20
21     printf("Bulan    Jumlah hari\n");
22
23     for (i = 0; i < 12; ++i) {
24         printf("%s %d\n", bulan[i].nama, bulan[i].jumlah);
25     }
26
27     getch();
28 }
29

```

Hasil Output :

```
Bulan    Jumlah hari
Jan 31
Feb 28
Mar 31
Apr 30
Mei 31
Jun 30
Jul 31
Ags 31
Sep 30
Okt 31
Nov 30
Des 31
-----
```

Analisa :

- Struktur bulan memiliki dua anggota: jumlah untuk menyimpan jumlah hari dalam bulan, dan nama untuk menyimpan nama bulan. Ukuran array nama adalah 4 untuk menampung nama bulan bersama dengan karakter null ('\0') di akhir string.
- Array bulan mengandung 12 elemen dari tipe data struct bulan, mewakili informasi tentang jumlah hari dalam masing-masing bulan dan nama bulannya.
- Program mencetak header "Bulan Jumlah hari" ke layar.
- Program kemudian melakukan iterasi melalui array bulan, mencetak nama bulan dan jumlah hari dalam bulan tersebut menggunakan fungsi printf().

L. Program Keduabelas (Struktur 5)

```
1  #include <stdio.h>
2  #include <string.h>
3
4  struct waktu {
5      int jam;
6      int menit;
7      int detik;
8  };
9
10 struct waktu time_update(struct waktu now)
11 {
12     struct waktu new_time;
13     new_time = now;
14     ++new_time.detik;
15
16     if (new_time.detik == 60) {
17         new_time.detik = 0;
18         ++new_time.menit;
19
20         if (new_time.menit == 60) {
21             new_time.menit = 0;
22             ++new_time.jam;
23
24             if (new_time.jam == 24) {
25                 new_time.jam = 0;
26             }
27         }
28     }
29
30     return new_time;
31 }
32
33 int main() {
34     int i;
35
36     struct waktu tes[5] = {
37         {11, 59, 59}, {12, 0, 0}, {1, 29, 59},
38         {23, 59, 59}, {19, 12, 27}
39     };
40
41     for (i = 0; i < 5; ++i) {
42         printf("Waktu adalah %.2d:%.2d:%.2d", tes[i].jam, tes[i].menit, tes[i].detik);
43         tes[i] = time_update(tes[i]);
44         printf("... satu detik berikutnya %.2d:%.2d:%.2d\n", tes[i].jam, tes[i].menit, tes[i].detik);
45     }
46
47     return 0;
48 }
```

Hasil Output :

```
Waktu adalah 11:59:59... satu detik berikutnya 12:00:00
Waktu adalah 12:00:00... satu detik berikutnya 12:00:01
Waktu adalah 01:29:59... satu detik berikutnya 01:30:00
Waktu adalah 23:59:59... satu detik berikutnya 00:00:00
Waktu adalah 19:12:27... satu detik berikutnya 19:12:28
```

Analisa :

- Struktur waktu memiliki tiga anggota: jam, menit, dan detik untuk menyimpan informasi waktu.
- Fungsi `time_update` digunakan untuk memperbarui waktu. Fungsi ini menerima objek waktu `now`, menambahkan satu detik ke waktu tersebut, dan mengembalikan waktu yang sudah diperbarui.
- Fungsi `main` adalah fungsi utama yang akan dieksekusi ketika program dijalankan.
- Program menginisialisasi array `tes` dengan lima objek waktu yang berbeda.
- Program melakukan loop melalui array `tes` dan memanggil fungsi `time_update` untuk memperbarui waktu pada setiap iterasi.
- Program mencetak waktu sebelum dan sesudah diperbarui ke layar.
- Program menggunakan fungsi `printf` untuk mencetak waktu dalam format "HH:MM:SS" sebelum dan sesudah diperbarui.
- Fungsi `time_update` digunakan untuk memperbarui waktu dengan menambahkan satu detik. Fungsi ini menangani peningkatan jam, menit, dan detik secara benar dengan memeriksa apakah setiap unit waktu mencapai batasnya (misalnya, 60 detik menjadi 0 detik, dan 59 menit menjadi 0 menit).
- Penggunaan `printf` dengan format specifier `%.2d` memastikan bahwa waktu selalu dicetak dengan dua digit, termasuk angka nol di depan jika nilainya kurang dari 10.

M. Program Ketigabelas (Struktur 6)

```
1  #include <stdio.h>
2  #include <string.h>
3  #include <conio.h>
4
5  struct entry {
6      char kata[10];
7      char definisi[50];
8  };
9  int sama_str(char str1[], char str2[]) {
10     return strcmp(str1, str2) == 0;
11 }
12 int lihat(struct entry kamus[], char cari[], int jumlah) {
13     int i;
14     for (i = 0; i < jumlah; i++) {
15         if (sama_str(cari, kamus[i].kata)) {
16             return i;
17         }
18     }
19     return -1;
20 }
21 int main() {
22     struct entry kamus[] = {
23         {"komodo", "Kayaknya di Sulawesi ada tuh"},
24         {"unta", "Lahirnya di Mesir tapi tidak bisa bahasa Arab"},
25         {"kangguru", "Suka melompat di Australia"},
26         {"kingkong", "Lawannya Superman kali"},
27         {"harimau", "Belang-belang dan ada di Kalimantan"},
28         {"kobra", "Ular dari India"},
29         {"kancil", "Suka nyolong timun"},
30         {"iguana", "Mahal harganya loh"},
31         {"nyamuk", "Nakal suka sedot sana sini"},
32         {"tikus", "Suka menggoda cewek cakep"},
33         {"buaya", "wah ini sih istrinya pakai"}
34     };
35
36     int banyaknya = sizeof(kamus) / sizeof(kamus[0]);
37     char kata[10];
38     int angka_masukan;
39     printf("Masukkan kata: ");
40     scanf("%s", kata);
41     angka_masukan = lihat(kamus, kata, banyaknya);
42
43     if (angka_masukan != -1)
44         printf("%s\n", kamus[angka_masukan].definisi);
45     else
46         printf("Maaf, kata itu tidak ada dalam kamus.\n");
47     getch();
48 }
```


Hasil Output

```
Masukkan kata: buaya
wah ini sih istrinya pakai
-----
```

Analisa ;

- Program menggunakan struktur entry untuk menyimpan pasangan kata dan definisi yang terkait.
- sama_str(): Fungsi ini membandingkan dua string dan mengembalikan 0 jika string tersebut sama.
- Fungsi lihat(): mencari kata dalam kamus menggunakan fungsi sama_str().
- Jika kata ditemukan, fungsi mengembalikan indeks kata dalam kamus.
- Jika kata tidak ditemukan, fungsi mengembalikan -1.
- Fungsi main(): Fungsi utama program.
- Mendeklarasikan array kamus yang berisi kata dan definisinya.
- Menggunakan fungsi lihat() untuk mencari kata yang dimasukkan pengguna dalam kamus.
- Menampilkan definisi kata jika ditemukan, dan pesan kesalahan jika tidak ditemukan.

PENUTUP

KESEIMPULAN

Dalam praktikum yang disebutkan di atas, kami telah mempelajari dan menerapkan berbagai konsep dasar pemrograman C/C++, terutama yang berkaitan dengan struktur dan fungsi. Praktikum ini sangat penting untuk meningkatkan pemahaman dasar kami tentang aspek penting pemrograman. Pertama, kami mempelajari cara struktur digunakan dalam C/C++ untuk mengorganisasi dan mengelompokkan berbagai jenis data menjadi satu entitas. Kita menciptakan struktur yang disebut sebagai "entry", yang terdiri dari dua komponen: "kata" dan "definisi." Struktur ini sangat penting untuk menyimpan dan mengatur pasangan kata dan definisinya dalam kamus. Kami juga memahami pentingnya fungsi dalam pemrograman. Fungsi adalah blok kode yang memiliki nama dan dapat digunakan kembali dalam berbagai bagian program. Kami menetapkan fungsi "lihat" dalam program ini untuk mencari kata dalam kamus dengan tiga alasan: kamus, kata yang dicari, dan jumlah elemen kamus. Oleh karena itu, kami memiliki pemahaman tentang cara mengatur kode secara terstruktur dan membaginya menjadi bagian-bagian yang berbeda untuk meningkatkan keterbacaan dan modularitas program.

DAFTAR PUSTAKA

<https://www.duniaikom.com/tutorial-belajar-c-cara-membuat-fungsi-function-bahasa-c/>

<https://nulis-ilmu.com/fungsi-dalam-bahasa-c/>

<https://www.petanikode.com/c-struct/>