Modul Praktikum

Pemrograman Lanjut

Laboratium Digital

Tim Penyusun Modul

Tim Asisten Lab Digital 2021

Program Studi S1 Teknik Komputer

Fakultas Teknik

Universitas Indonesia

2021

KATA PENGANTAR

Praktikum ini adalah bagian dari mata kuliah Pemrograman Lanjut dan Praktikum yang diberikan untuk mahasiswa Semester 2 Program Studi Teknik Komputer. Mata kuliah ini merupakan mata kuliah lanjutan yang diberikan kepada mahasiswa dari serangkaian mata kuliah untuk mendukung 2 CP Prodi, yaitu Mampu merancang algoritma untuk masalah tertentu dan mengimplementasikannya ke dalam pemrograman (C6) dan Mampu memanfaatkan teknologi informasi dan komunikasi (C3).

Capaian Pembelajaran Mata Kuliah (CPMK) dari mata kuliah ini adalah:

1. Mampu merancang program komputer prosedural kompleks dengan struktur data dinamis (C6)
2. Mampu menunjukkan sikap kritis, kreatif, dan inovatif dan menghargai orang lain dalam kelompok untuk memecahkan masalah bersama dan tugas kelompok Pemrograman Lanjut (C3, A3)
3. Mampu menggunakan software pemrogram komputer dengan mahir (C3)

Sedangkan Sub-CPMK yang akan dicapai adalah:

1.1. Mampu mengimplementasikan algoritma rekursif ke dalam pemrograman

1.2. Mampu mengimplementasikan algoritma searching dan sorting ke dalam pemrograman

1.3. Mampu membuat program komputer prosedural kompleks dengan linked list, stack dan queue

1.4. Mampu mengimplementasikan multi-threading dan parallel programming

1.5. Mampu merancang perangkat lunak sederhana dengan struktur data dinamis

2.1. Mampu menunjukkan proses berpikir kritis, kreatif dan inovatif dalam menyelesaikan permasalahan kelompok

2.2. Mampu berkomunikasi dengan sopan

2.3. Mampu menghargai pendapat orang lain

3.1. Mampu menggunakan software pemrogram komputer untuk program kompleks dengan mahir

Pada mata kuliah ini mahasiswa akan mengasah kemampuan cara berpikir dan penyelesaian masalah dengan membuat algoritma, kemudian menerjemahkan algoritma tersebut ke dalam bahasa pemrograman yang dapat dijalankan oleh komputer. Pada mata kuliah ini mahasiswa akan mempelajari Rekursif, Searching, Sorting, Linked list, Stack, Queue, Multi-threading, Parallel programming,, dan pada bagian akhir akan ditutup dengan proyek akhir pemrograman yang dibuat oleh mahasiswa.

Bahasa C merupakan bahasa pemrograman terstruktur, yang membagi program dalam sejumlah blok. Tujuannya adalah untuk mempermudah dalam pembuatan dan pengembangan program. Bahasa C menggunakan standarisasi ANSI (American National Standardization Institute) yang dijadikan acuan oleh para pembuat compiler C. Bahasa C terdiri dari fungsi-fungsi dan setiap program C memiliki fungsi utama yang disebut main. Program akan dieksekusi dimulai dari statement pertama pada fungsi main tersebut.

Akhir kata, diharapkan modul praktikum ini akan dapat menjadi referensi untuk membuat program dalam Bahasa C secara umum, dan menjadi panduan dalam menjalankan praktikum mata kuliah Pemrograman Lanjut, secara khusus.

Depok, Februari 2021

Tim Penyusun Modul

MODUL 6: FILE HANDLING

File merupakan suatu container pada sistem penyimpanan komputer yang digunakan untuk menyimpan data.

Definisi

File Handling dibutuhkan karena,

1. Ketika program di terminate, semua data akan hilang. Dengan menyimpan data dalam bentuk file, maka data akan tetap utuh walaupun program sudah determinate
2. Jika data yang dimasukkan dalam jumlah yang besar, maka akan membutuhkan waktu untuk memasukkannya satu persatu. Maka dari itu dapat menggunakan file handling untuk proses inputnya.
3. Memudahkan untuk melakukan pemindahan data dari suatu program komputer ke program komputer lain tanpa adanya perubahan

Tipe File,

1. **Text Files**
   Memiliki ekstensi file dalam bentuk **.txt** dan berisikan plain text sehingga mudah untuk di read ataupun di write.
2. **Binary Files**
   Memiliki ekstensi file dalam bentuk **.bin**, menyimpan data dalam bentuk binary (0 dan 1). Dapat berisikan lebih banyak data, tidak mudah dibaca, namun memiliki tingkat keamanan yang lebih tinggi

Read and Write File

Declaration

```
FILE *fptr
```

Opening a file

```
ptr =  fopen("fileopen", "mode");
```

Example

```
fopen("D:\\daskom\data.txt", "w");
fopen("D:\\daskom\data.bin", "rb");
```

Apabila file **data.txt** tidak tersedia, maka fungsi fopen() akan membuat file dengan nama **data.txt** dan membukanya dalam mode writing **w**. Apabila file **data.bin** tersedia, maka file tersebut akan dibuka dalam mode read only **rb.**

Opening Mode

Mode	Meaning of Mode	During Inexistence of file
r	Open for reading.	If the file does not exist, fopen() returns NULL.
rb	Open for reading in binary mode.	If the file does not exist, fopen() returns NULL.
w	Open for writing.	If the file exists, its contents are overwritten. If the file does not exist, it will be created.
wb	Open for writing in binary mode.	If the file exists, its contents are overwritten. If the file does not exist, it will be created.
a	Open for append. Data is added to the end of the file.	If the file does not exist, it will be created.
ab	Open for append in binary mode. Data is added to the end of the file.	If the file does not exist, it will be created.

`r+`	Open for both reading and writing.	If the file does not exist, `fopen()` returns NULL.
`rb+`	Open for both reading and writing in binary mode.	If the file does not exist, `fopen()` returns NULL.
`w+`	Open for both reading and writing.	If the file exists, its contents are overwritten. If the file does not exist, it will be created.
`wb+`	Open for both reading and writing in binary mode.	If the file exists, its contents are overwritten. If the file does not exist, it will be created.
`a+`	Open for both reading and appending.	If the file does not exist, it will be created.
`ab+`	Open for both reading and appending in binary mode.	If the file does not exist, it will be created.

Closing a File

```
fclose(fptr);
```

Reading and Writing to a text file

```
fprintf();
fscanf();
```

Example Write to a text File

```
#include <stdio.h>
#include <stdlib.h>
```

```c
int main()
{
   int num;
   FILE *fptr;

   // use appropriate location if you are using MacOS or Linux
   fptr = fopen("D:\\daskom.txt","w");

   if(fptr == NULL)
   {
      printf("Error!");
      exit(1);
   }

   printf("Enter num: ");
   scanf("%d",&num);

   fprintf(fptr,"%d",num);
   fclose(fptr);

   return 0;
}
```

Example Read from a text file

```c
#include <stdio.h>
#include <stdlib.h>

int main()
{
   int num;
   FILE *fptr;

   if ((fptr = fopen("C:\\daskom.txt","r")) == NULL){
       printf("Error! opening file");

       // Program exits if the file pointer returns NULL.
       exit(1);
   }

   fscanf(fptr,"%d", &num);
```

```
   printf("Value of n=%d", num);
   fclose(fptr);

   return 0;
}
```

Reading and Writing to a binary file

```
fread(addressData, sizeData, numbersData, pointerToFile);
fwrite(addressData, sizeData, numbersData, pointerToFile);
```

Example Write to a binary file

```
#include <stdio.h>
#include <stdlib.h>

struct threeNum
{
   int n1, n2, n3;
};

int main()
{
   int n;
   struct threeNum num;
   FILE *fptr;

   if ((fptr = fopen("C:\\daskom.bin","wb")) == NULL){
       printf("Error! opening file");

       // Program exits if the file pointer returns NULL.
       exit(1);
   }

   for(n = 1; n < 5; ++n)
   {
      num.n1 = n;
      num.n2 = 5*n;
      num.n3 = 5*n + 1;
      fwrite(&num, sizeof(struct threeNum), 1, fptr);
   }
```

```
   fclose(fptr);

   return 0;
}
```

Example Read from a binary file

```
#include <stdio.h>
#include <stdlib.h>

struct threeNum
{
   int n1, n2, n3;
};

int main()
{
   int n;
   struct threeNum num;
   FILE *fptr;

   if ((fptr = fopen("C:\\program.bin","rb")) == NULL){
       printf("Error! opening file");

       // Program exits if the file pointer returns NULL.
       exit(1);
   }

   for(n = 1; n < 5; ++n)
   {
      fread(&num, sizeof(struct threeNum), 1, fptr);
      printf("n1: %d\tn2: %d\tn3: %d", num.n1, num.n2, num.n3);
   }
   fclose(fptr);

   return 0;
}
```

Additional : Getting data using fseek()

```
fseek(FILE * stream, long int offset, int whence);
```

Whence	Meaning
SEEK_SET	Starts the offset from the beginning of the file.
SEEK_END	Starts the offset from the end of the file.
SEEK_CUR	Starts the offset from the current location of the cursor in the file.

Example

```c
#include <stdio.h>
#include <stdlib.h>

struct threeNum
{
   int n1, n2, n3;
};

int main()
{
   int n;
   struct threeNum num;
   FILE *fptr;

   if ((fptr = fopen("C:\\daskom.bin","rb")) == NULL){
       printf("Error! opening file");

       // Program exits if the file pointer returns NULL.
       exit(1);
   }

   // Moves the cursor to the end of the file
   fseek(fptr, -sizeof(struct threeNum), SEEK_END);

   for(n = 1; n < 5; ++n)
   {
      fread(&num, sizeof(struct threeNum), 1, fptr);
      printf("n1: %d\tn2: %d\tn3: %d\n", num.n1, num.n2, num.n3);
```

```
        fseek(fptr, -2*sizeof(struct threeNum), SEEK_CUR);
    }
    fclose(fptr);

    return 0;
}
```

Program ini akan melakukan reading pada file **daskom.bin** dari baris paling belakang terlebih dahulu, kemudian mencetaknya.