PLAN 396
Lecture 7

Dr. Hossen Asiful Mustafa

https://hossenmustafa.buet.ac.bd

Global and Local Variable

- Global Variable
  - A variable that is declared outside the function
  - It can be initialized during declaration
  - Any function in the program can use the variable
- Local Variable
  - A variable that is declared inside a function
  - It can be initialized during declaration
  - Only the function where it is declared can use the variable

Global and Local Variable

```
#include<stdio.h>
int size = 10; // this is global and initialized
int n;       // this is global but not initialized

int add(){
        n =10;
        size++;
}
int main(){
        int x;            // this is local
        int n = 5;     // this is local
}
```

Variable Scope

- Local variable can be used within the function it is declared
- Global variable can be used in any function within the program
- There can be a local variable and a global variable with the same name
  - In such case, the local variable will be used not the global one

Variable Scope

```
#include<stdio.h>
int n = 100; // this is global
int add(){
        printf(“%d”. n); // this will print 100 (global)
}
int main(){
        int n = 5;  // this is local
        printf(“%d”. n);   // this will print 5 (local)
}
```

Input and Output

- Standard I/O
  - printf
  - scanf
  - getchar
  - putchar
  - etc.
- File I/O
  - Read from a file from the filesystem
  - Write a new file
  - Append to an existing file

Reading File

- Variable
  - FILE *fp;
- Function
  - FILE *fopen(char *name, char *mode);
- Code Example:
  - FILE * fp; // variable declaration for file pointer
  - fp = fopen(“input.txt”, “r”); // open the file in read mode
  - fscanf(fp, “%d”, &n); // read a value from file
  - fclose(fp);

Reading File Example

```
int main(){
        int n;
        FILE * fp;
        fp = fopen(“input.txt”, "r");
        if (fp == NULL) {
                printf(“can't open\n,);
                return 1;
        }
        fscanf(fp, “%d”, &n);
        printf(“n = %d\n”, n);
        fclose(fp);
        return 0;
}
```

Reading File

- Functions to read
  - fscanf(FILE *fp, char* format)
  - fgetc(FILE *fp)
  - fgets(char *array, int size, FILE *fp)
- These functions return EOF (End of File) when no data is available

Writing File

- Variable
  - FILE *fp;
- Function
  - FILE *fopen(char *name, char *mode);
- Code Example:
  - FILE * fp; // variable declaration for file pointer
  - fp = fopen(“output.txt”, “w”); // open file in write mode
  - fprintf(fp, “The value of n is %d”, n); // write to file
  - fclose(fp);

Writing File Example

```
int main(){
        int n = 500;
        FILE * fp;
        fp = fopen(“output.txt”, “w");
        if (fp == NULL) {
                printf(“can't open\n,);
                return 1;
        }
        fprintf(“n = %d\n”, n);
        fclose(fp);
        return 0;
}
```

Writing File

- Functions to read
  - fprintf(FILE *fp, char* format)
  - fputc(FILE *fp)
  - fputs(char *array, FILE *fp)

Appending File

- Variable
  - FILE *fp;
- Function
  - FILE *fopen(char *name, char *mode);
- Code Example:
  - FILE * fp; // variable declaration for file pointer
  - fp = fopen(“output.txt”, “a”); // open file in write mode
  - fprintf(fp, “The value of n is %d”, n); // write to file
  - fclose(fp);

Appending File Example

```
int main(){
        int n = 500;
        FILE * fp;
        fp = fopen(“output.txt”, “a");
        if (fp == NULL) {
                printf(“can't open\n,);
                return 1;
        }
        fprintf(“n = %d\n”, n);
        fclose(fp);
        return 0;
}
```

** New write will be appended at the end of the file

More about fopen()

- For “r” mode,
  - fopen returns NULL if the file cannot be found
- For “w” mode,
  - fopen will create a new file with the filename
  - If file exists, that will be replaced
- For “a” mode,
  - fopen will open the file if exists
  - If the file doesn’t exist, a new file will be created
- C allows to open the same file in “r”, “w” mode
  - But, it should be avoided if possible

fflush and fclose

- When we do a write to a file, e.g., using fprintf
  - The OS doesn’t write it instantly in the file
  - The OS maintains a buffer, and writes to file when the buffer crosses a limit or when fclose is used
  - This is to maintain OS performance
- We can use fflush(FILE *fp) to force the OS to write to a file.
  - fflush function must be used after the fprintf
- We must use fclose when file operations are complete
  - Otherwise, the OS may not write the file at all

Term Project

- Team: 2-3 members
- Development using C or Python
- Due: End of semester
- Topic: Choose a problem related to your program
  - Submit your topic by 22/01/22

Class Assignment

- Write a program named classassignment10.c
- The program should
  - read file password.txt (available in Teams). Each line of the file contains a username and a password separated by space
  - save the username in one array and passwords in another array of strings (2D char array). Save the corresponding username and password in the same index
  - prompt user to enter a username and a password from keyboard
  - match the username with the username in the array (use strcmp function from string.h) and then match corresponding password
  - If user-password matches, show success message
  - If the username doesn’t match, append the username password in the password.txt