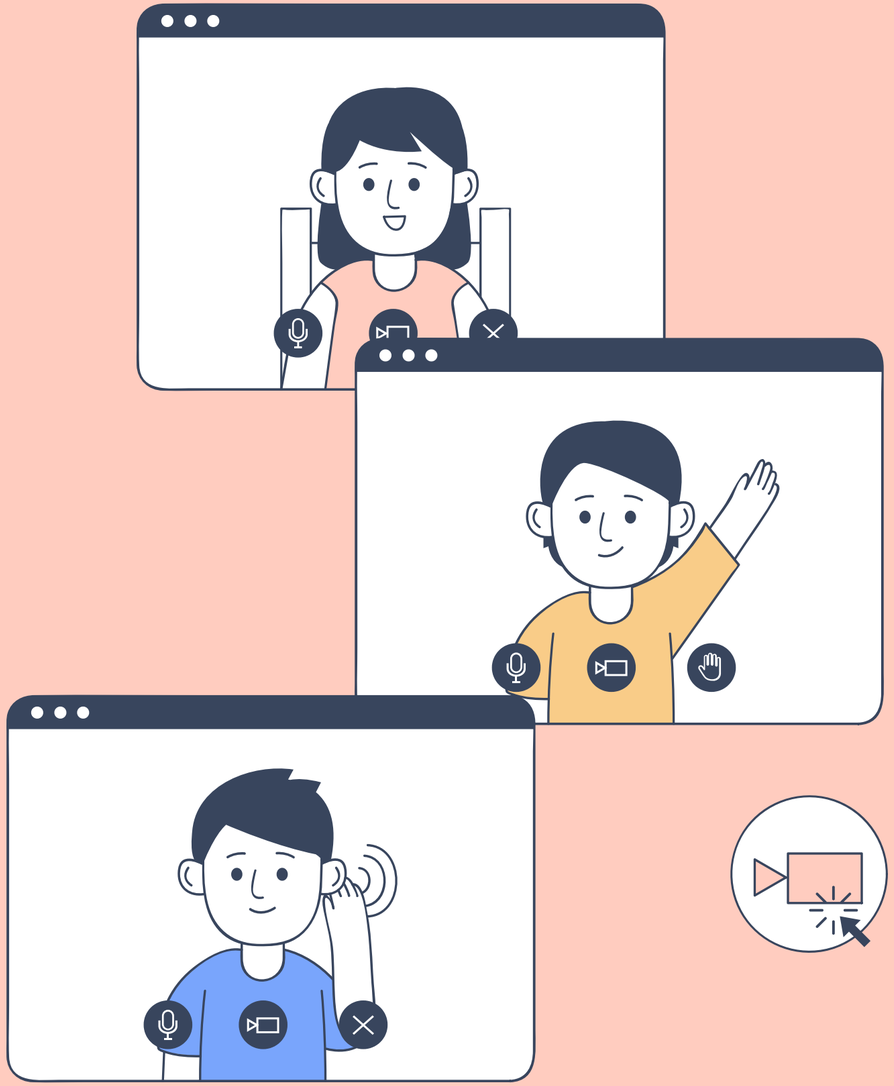
Structures and Files

Dan Victor Gapaz



Learning Outcomes

1

Have a better understanding on declaring and initializing Structures.

2

Learn two ways to access members of a structure and learn the four operations that can be used by structure variables.

3

Review on Files and its operations.

Structures

Structures

collection of **Related Data**
that may of **different types.**

Structures

collection of **Related Data** that may of **different types.**

Like a Dictionary in python.

3 ways to define a structure:

Using Structure Tags

```
struct <structure tag>{
    <member 1>;
    <member 2>;
    ...
    <member n>;
};
```

Using type-definition

```
typedef struct{
    <member 1>;
    <member 2>;
    ...
    <member n>;
}<synonym;
```

Using Both

```
typedef struct <structure_tag>{
    <member 1>;
    <member 2>;
    ...
    <member n>;
}<synonym;
```

3 ways to define a structure:

Using Structure Tags

```c
struct Tutor {
    char fname[50];
    char lname[50];
    int age;
};
```

Using type-definition

```c
typedef struct {
    char fname[50];
    char lname[50];
    int age;
}Tutor;
```

Using Both

```c
typedef struct Tutor{
    char fname[50];
    char lname[50];
    int age;
}PEERTutor;
```

Initializing Structures

```c
typedef struct {
    char fname[50];
    char lname[50];
    int age;
}Tutor;
```

Initializing Structures

```
typedef struct {
    char fname[50];
    char lname[50];
    int age;
}Tutor;
```

```
Tutor firstTutor = {"Dan", "Gapaz", 19};
```

Initializing Structures

```
struct Tutor {
    char fname[50];
    char lname[50];
    int age;
};
```

Initializing Structures

```
struct Tutor {
    char fname[50];
    char lname[50];
    int age;
};
```

```
struct Tutor firstTutor = {"Dan","Gapaz",19};
```

Initializing Structures

```c
typedef struct Tutor{
    char fname[50];
    char lname[50];
    int age;
}PEERTutor;
```

Initializing Structures

```c
typedef struct Tutor{
    char fname[50];
    char lname[50];
    int age;
}PEERTutor;
```

```c
struct Tutor firstTutor = {"Dan", "Gapaz", 19};
```

Initializing Structures

```c
typedef struct Tutor{
    char fname[50];
    char lname[50];
    int age;
}PEERTutor;
```

```c
struct Tutor firstTutor = {"Dan", "Gapaz", 19};
```

or

```c
PEERTutor firstTutor = {"Dan", "Gapaz", 19};
```

Notes in Defining Structures

02

Defining a structure is like creating your own data type.

01

Structures are usually defined outside any function.

03

You cannot use them unless you declare a structure variable.

02

Defining a structure is like creating your own data type.

```
typedef struct {
    char fname[50];
    char lname[50];
    int age;
}Tutor;
```

You are creating a data type that can be called as “Tutor”

03

You cannot use them unless you declare a structure variable.

```c
typedef struct {
    char fname[50];
    char lname[50];
    int age;
}Tutor;
```

```c
Tutor = {"Dan", "Gapaz", 19};
```

```c
int = 7;
```

03

You cannot use them unless you declare a structure variable.

```
typedef struct {
    char fname[50];
    char lname[50];
    int age;
}Tutor;
```

```
Tutor = {"[illegible]n", "[illegible]Z", 19};
```

```
[illegible]t = 7;
```

03

You cannot use them unless you declare a structure variable.

```
typedef struct {
    char fname[50];
    char lname[50];
    int age;
}Tutor;
```

```
Tutor firstTutor = {"Dan", "Gapaz", 19};
```

03

You cannot use them unless you declare a structure variable.

```
typedef struct {
    char fname[50];
    char lname[50];
    int age;
}Tutor;
```

```
Tutor firstTutor     Dan", "Gapaz", 19};
```

Accessing Members of a Structure

```c
typedef struct {
    char fname[50];
    char lname[50];
    int age;
}Tutor;
```

```c
PEERTutor firstTutor;
```

Accessing Members of a Structure

```
typedef struct {
    char fname[50];
    char lname[50];
    int age;
}Tutor;
```

```
PEERTutor firstTutor;
```

DOT OPERATOR

Accessing Members of a Structure

```c
typedef struct {
    char fname[50];
    char lname[50];
    int age;
}Tutor;
```

```c
PEERTutor firstTutor;
```

DOT OPERATOR

SYNTAX: VARNAME.MEMBERNAME

Accessing Members of a Structure

```
typedef struct {
    char fname[50];
    char lname[50];
    int age;
}Tutor;
```

```
PEERTutor firstTutor;
```

DOT OPERATOR

```
strcpy(firstTutor.fname, "Juan");
strcpy(firstTutor.lname, "Dela Cruz");
firstTutor.age = 19;
```

Accessing Members of a Structure

```
typedef struct {
    char fname[50];
    char lname[50];
    int age;
}Tutor;
```

```
PEERTutor firstTutor;
```

DOT OPERATOR

```
strcpy(firstTutor.fname, "Juan");
strcpy(firstTutor.lname, "Dela Cruz");
firstTutor.age = 19;
```

STRUCTURE POINTER

Accessing Members of a Structure

```
typedef struct {
    char fname[50];
    char lname[50];
    int age;
}Tutor;
```

```
PEERTutor firstTutor;
```

DOT OPERATOR

```
strcpy(firstTutor.fname, "Juan");
strcpy(firstTutor.lname, "Dela Cruz");
firstTutor.age = 19;
```

STRUCTURE POINTER

SYNTAX: PTR->MEMBER

Accessing Members of a Structure

```
typedef struct {
    char fname[50];
    char lname[50];
    int age;
}Tutor;
```

```
PEERTutor firstTutor;
```

DOT OPERATOR

```
strcpy(firstTutor.fname, "Juan");
strcpy(firstTutor.lname, "Dela Cruz");
firstTutor.age = 19;
```

STRUCTURE POINTER

```
Tutor secondTutor;
Tutor *ptrToSecondTutor = &secondTutor;
strcpy(ptrToSecondTutor->fname, "Juan");
strcpy(ptrToSecondTutor->lname, "Dela Cruz");
ptrToSecondTutor->age = 19;
```

Nested Structures

```c
typedef struct{
    char subject[50];
    char courseCode[15];
}CoursesOffered;
```

```c
typedef struct {
    char fname[50];
    char lname[50];
    int age;
    CoursesOffered course[3];
}Tutor;
```

Nested Structures

```c
typedef struct{
    char subject[50];
    char courseCode[15];
}CoursesOffered;
```

```c
typedef struct {
    char fname[50];
    char lname[50];
    int age;
    CoursesOffered course[3];
}Tutor;
```

Array of Structures

Files

Used to **load** data before a program ends

Files

Used to **load** data before a program ends

Used to **store** data before a program ends

Files

Used to **load** data before a program ends

Used to **store** data before a program ends

Used for data persistence

Files

Used to **load** data before a program ends

Used to **store** data before a program ends

Used for data persistence

Files

operations are handled by a **file pointer**

Used to **load** data before a program ends

Used to **store** data before a program ends

Used for data persistence

Files

operations are handled by a **file pointer**

```
FILE <var_name>;

FILE *fp;
```

THINK ABOUT THIS!

We are to open a txt file named ‘cmsc21.txt’ and we want to read its content, what should we write in our code?

THINK ABOUT THIS!

We are to open a txt file named ‘cmsc21.txt’ and we want to read its content, what should we write in our code?

```
FILE *fp;
fp = fopen("cmsc21.txt", "r");
```

THINK ABOUT THIS!

We are to open a txt file named ‘cmsc21.txt’ and we want to read its content, what should we write in our code?

```
fopen("cmsc21.txt", "r");
```

File Modes

MODE	DESCRIPTION
r	read only
w	re(create) the file and write into it.
a	append to the file

File Modes

MODE	DESCRIPTION
r+	read or write: file MUST exist
w+	re(create) the file and write into it.
a+	append/read to the file

Closing Files

```
fclose(fp);
```

Closing Files

```
fclose(fp);
```

Why do we need to close a file?

File Writing

fputc vs fprintf

FILE READING

FILE READING

fgets

FILE READING

fgets

```
char fgets(char *str, int
size, FILE *fp);
```

FILE READING

fgets

```
char name[50];
fgets(name, 50,fp );
```

FILE READING

fgetc

fgets

```
char name[50];
fgets(name, 50,fp );
```

FILE READING

fgetc

```
char fgetc(FILE *fp);
```

fgets

```
char name[50];
fgets(name, 50,fp );
```

FILE READING

fgetc

```
fgetc(fp);
```

fgets

```
char name[50];
fgets(name, 50,fp );
```

FILE READING

fgetc

```
fgetc(fp);
```

fgets

```
char name[50];
fgets(name, 50,fp );
```

fscanf

FILE READING

fgetc

```
fgetc(fp);
```

fgets

```
char name[50];
fgets(name, 50,fp );
```

fscanf

```
char fscanf (FILE *fp, char
*format, variables);
```

FILE READING

fgetc

```c
fgetc(fp);
```

fgets

```c
char name[50];
fgets(name, 50,fp );
```

fscanf

```c
char name[50];
fscanf(fp, "%[^\n]s", name);
```

Thank You

Thank You

Do you have any questions of us?