ELEC 278

Tutorial Week 3

2022 Fall
Instructors:
Dr. Jianbing Ni & Dr. Mohammadali Hedayati

Tutorial TAs:
Shayan Noei & AmirHossein Sojoodi

Structures

- Arrays:
  - Variables that can hold **several data items** of the **same kind**.
- Structures:
  - A data type that allows to **combine data items of different kinds**.

Suppose you want to keep track students in a class. You want to track:

- **First name**
- **Last name**
- **Student number**

We use structures to represent each record.

Structures (Defining)

A structures is usually defined in this format:

```
struct tag {
    member-list
    member-list
    member-list
    ...
} name ;
```

```
struct book_t {
    char title[50];
    char author[50];
    char subject[100];
    int book_id;
} book;
```

Structures (Defining)

Sometimes, the struct tag can be omitted:

```
struct {
    int a;
    char b;
    double c;
} s1;
```

We can define the struct first and declare the variable later:

```
struct SIMPLE {
    int a;
    char b;
    double c;
};
struct SIMPLE t1, t2[20], *t3;
```

Structures (Defining Example)

Suppose you want to keep track students in a class. You want to track:

- **First name**
- **Last name**
- **Student number**

We use structures to represent each student.

```
#include <stdio.h>
#include <string.h>
struct student {
	char first_name[50];
	char last_name[50];
	int student_id;
};
```

Structures (initialization Example)

```
#include <stdio.h>
struct book_t {
    char title[50];
    char author[50];
    char subject[100];
    int book_id;
};

struct book_t book = {"C ", "RUNOOB",
"struct", 123456};

int main() {
    printf("title : %s\nauthor: %s\nsubject:
%s\nbook_id: %d\n",  book.title, book.author,
book.subject, book.book_id);
}
```

Structures (Accessing Example)

```
int main( ) {
   struct student student1;
   struct student student2;

   strcpy( student1.first_name, “Dave”);
   strcpy( student1.last_name, “Johnson”);
   student1.student_id = 6495407;

   strcpy( student2.first_name, “Mike”);
   strcpy( student2.last_name, “Coleman”);
   student2.student_id = 6495700;

   printf( “student 1 first name : %s\n", student1.first_name);
   printf( “student 1 last name : %s\n", student1.last_name);
   printf( “student 1 id : %d\n", student1.student_id);
   return 0;
}
```

Output

Structures (As Function Arguments)

```
void printStudent(struct Students Student);

int main( ) {
   struct Students Student1;
   strcpy(Student1.first_name, “Dave”);
   strcpy(Student1.last_name, “Johnson”);
   Student1.student_id = 6495407;
   printStudent(Student1);
   return 0;
}

void printStudent(struct Students Student) {
   printf(“First Name : %s\n", Student.first_name);
   printf(“Last Name : %s\n", Student.last_name);
   printf(“Student ID : %d\n", Student.student_id);
}
```

Structures (Pointers to Structures)

- We define **pointers to structures** in the **same way** as you define pointer to any **other variable.**

```
struct Students *struct_pointer;
```

```
struct_pointer = &Student1;
```

You must use the **→ operator** to access the members of a **structure using a pointer.**

```
struct_pointer->first_name;
```

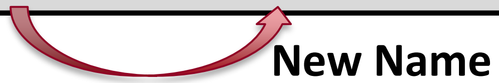
Structures (Pointers to Structures)

```
void printStudent(struct Students *Student);

int main( ) {
   struct Students Student1;

   strcpy(Student1.first_name, “Dave”);
   strcpy(Student1.last_name, “Johnson”);
   Student1.student_id = 6495407;

   printStudent(&Student1);
   return 0;
}
void printStudent(struct Students *Student) {
   printf(“First Name : %s\n", Student->first_name);
   printf(“Last Name : %s\n", Student->last_name);
   printf(“Student ID : %d\n", Student->student_id);
}
```

Typedef

- We use **Typedef** to give a type a new name.

```
typedef unsigned char BYTE;
```



New Name

Now: BYTE can be used as a short version of the type **unsigned char.**

```
BYTE b1, b2;
```

Naming convention

The image part with relationship ID rId2 was not found in the file.

Uppercase letters, but you can use lowercase too!

Typedef (Example)

```
#include <stdio.h>
int main(){
    typedef unsigned int UINT;
    UINT i,j;
    i=10;
    j=20;
    printf("Value of i is :%ud",i);
    printf("Value of j is :%ud",j);
    return 0;
}
```

Typedef (With Structures)

We can use typedef to create the new struct type:

```
typedef struct {
    int a;
    char b;
    double c;
} Simple2;
 Simple2 u1, u2[20], *u3;
```

```
typedef struct simple2{
    int a;
    char b;
    double c;
} Simple2;
 Simple2 u1, u2[20], *u3;
```

The declaration of this struct contains other structs:

```
struct COMPLEX {
    char string[100];
    Simple2 a;
};
```

```
struct NODE {
    char string[100];
    struct NODE *next_node;
};
```

Typedef (With Structures)

- You can use **typedef** to give a name to your strucrure data types as well.

```
typedef struct student_t {
   char first_name[50];
   char last_name[50];
   int  student_id;
} student;

int main( ) {
   student st;
   strcpy( st.first_name, “Dave”);
   strcpy( st.last_name, “Johnson);
   st.student_id = 6495407;
   return 0;
}
```

typedef vs #define

- #**define** is a C-directive which is also used to define the aliases for various data types similar to **typedef.**

- **typedef** is limited to giving symbolic names to types only where as #**define** can be used to define alias for values as well, q., you can define 1 as ONE etc.
- **typedef** interpretation is performed by the compiler whereas #**define** statements are processed by the pre-processor.

Define (Example)

```
#include <stdio.h>

#define TRUE  1
#define FALSE 0

int main( ) {
   printf( "Value of TRUE : %d\n", TRUE);
   printf( "Value of FALSE : %d\n", FALSE);
   return 0;
}
```

File I/O

- You can use the **fopen( )** function to create a new file or to open an existing file. This call will initialize an object of the type **FILE**, which contains all the information necessary to control the stream.

```
FILE *fopen( const char * filename, const char * mode );
```

Closing a file:

```
int fclose( FILE *fp );
```

File I/O

Mode	Description
r	Opens an existing text file for reading purpose.
w	Opens a text file for writing. If it does not exist, then a new file is created. Here your program will start writing content from the beginning of the file.
a	Opens a text file for writing in appending mode. If it does not exist, then a new file is created. Here your program will start appending content in the existing file content.
r+	Opens a text file for both reading and writing.
w+	Opens a text file for both reading and writing. It first truncates the file to zero length if it exists, otherwise creates a file if it does not exist.
a+	Opens a text file for both reading and writing. It creates the file if it does not exist. The reading will start from the beginning but writing can only be appended.

File I/O (Writing)

Writing to a file:

```
int fprintf( FILE *fp, char *st, …);
```

```
int fputc( int c, FILE *fp );
```

The function **fputs()** writes the string **s** to the output stream referenced by fp. It returns a non-negative value on success, otherwise **EOF** is returned in case of any error.

```
int fputs( const char *s, FILE *fp );
```

File I/O (Writing Example)

```c
#include <stdio.h>

void main() {
   FILE *fp;
   fp = fopen("/tmp/test.txt", "w+");
   fprintf(fp, "This is testing for fprintf...\n");
   fputs("This is testing for fputs...\n", fp);
   fclose(fp);
}
```

File I/O (Reading)

below is the simplest function to read a single character from a file:

```
int fscanf( FILE * fp, char *str, … );
```

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

The functions **fgets()** reads up to n-1 characters from the input stream referenced by fp. It copies the read string into the buffer **buf**, appending a **null** character to terminate the string.

```
char *fgets( char *buf, int n, FILE *fp );
```

File I/O (Reading Example)

```
#define COUNT 128
#include <stdio.h>
void main() {
   FILE *fp;
   char buff[COUNT];

   fp = fopen("/tmp/test.txt", "r");
   fscanf(fp, "%s\n", buff);
   printf("1 : %s\n", buff );

   fgets(buff, COUNT, (FILE*)fp);
   printf("2: %s\n", buff );

   fgets(buff, COUNT, (FILE*)fp);
   printf("3: %s\n", buff );
   fclose(fp);
}
```

Example: https://github.com/amirsojoodi/ELEC278