05OGDLP - Algorithms & Programming
Laboratory 3

Learning objectives

- Arrays, multidimensional arrays, strings
- C composite data types (structs)
- Functions

1 Morse Code Translator

Morse code, utilized in telecommunications, encodes text characters into standardized sequences of short and long signals—dots and dashes, respectively. For a deeper understanding, see the detailed description on Wikipedia.

Your task is to write a C program capable of encoding and decoding messages using the International Morse code. This code covers the 26 basic Latin letters (`[A-Z]`), digits (`[0-9]`), and a subset of punctuation symbols. The Morse code representations for these characters is provided in the `alphabet.in` file.

Write a C program that

- Utilizes an efficient data structure, preferably a zero-based array, to map characters to their Morse representations based on the content of `alphabet.in`.
- Presents users with an interactive menu to select: encode, decode, or exit (hint: a switch-case structure can be helpful here).
- Accepts Morse-encoded sentences or cleartext messages directly from `stdin`.
- Outputs the corresponding translation on `stdout`.

Example

```
Morse code translator
(E)ncode (D)ecode e(X)it >>> e
Enter your message: Hello, World!
Encoded message:
.... . .-.. .-.. --- --..-- / .-- --- .-. .-.. -.. -.-.--

(E)ncode (D)ecode e(X)it >>> d
Enter Morse code: .... . .-.. .-.. --- --..-- / .-- --- .-. .-..
-.. -.-.--
```

```
Decoded message:
HELLO, WORLD!

(E)ncode (D)ecode e(X)it >>> x
```

Implementation notes

- In C, character variables inherently store ASCII values, i.e., integers between 0 to 127. Using this ASCII table can inspire an optimal data structure choice. A zero-based array, with ASCII values serving as indices, can efficiently map characters to Morse codes.
- Within a word, Morse symbols are separated by spaces. Different words, on the other hand, are demarcated by the `/` character. Additionally, the newline character (`\n`) is represented as `/`.
- Our Morse includes uppercase letters only. However, your solution should equate both upper and lower-case letters in Morse conversion (e.g., `a` and `A` both translate to `.-`). The functions defined in the `ctype.h` header can help you achieve this goal..
- Design an intuitive user interface using a `switch-case` structure. It is an effective method to process user selections, guiding them through the encoding and decoding stages.

2 Digital Library System

Imagine you are helping your local library digitize their book inventory. Your task is to create a simple Digital Library System to manage books. Each book has certain attributes and operations that can be performed on it.

Attributes of a book

- Title (string)
- Author (string)
- ISBN (string of 10 characters, consider it unique for every book)
- Year of publication (integer)
- Number of copies available (integer)

Operations

- Add a new book
- Update book details (by ISBN)
- Search for a book by title or author
- List all books by a specific author

- Display the total number of distinct books and total number of books (including multiple copies)
- Borrow a book (decrease the number of copies by 1)
- Return a book (increase the number of copies by 1)

Example

```
Welcome to the Digital Library System!
1. Add a new book
2. Update book details
3. Search for a book
4. List books by an author
5. Display total books
6. Borrow a book
7. Return a book
8. Exit
Choice >>> 1
Enter book title: The Great Gatsby
Enter author: F. Scott Fitzgerald
Enter ISBN: 1234567890
Enter year of publication: 1925
Enter number of copies: 5
Book added successfully!
```

Implementation notes

- Use an array of `structs` to represent the collection of books in the library:

  ```
  typedef struct {
      char title[MAX_LENGTH];
      char author[MAX_LENGTH];
      ...
      int copies;
  } book_t;
  ```

- Ensure the system can handle a scenario where a user might try to add a book with an ISBN that already exists.
- When a book is borrowed or returned, update the number of copies available accordingly. Make sure not to allow borrowing a book if no copies are available.
- Keep the user interface simple and interactive, prompting users with clear instructions and feedback.

3 Tic-tac-toe

Tic-tac-toe is a paper-and-pencil game for two players who take turns marking the spaces in a three-by-three grid with X or O. The player who succeeds in placing three of their marks in a horizontal, vertical, or diagonal row is the winner. It is a solved game, with a forced draw assuming best play from both players (source).

Build a program in C to play tic-tac-toe against the computer or another human player.

At the start of each game, the application lets the user choose the type of player (human or computer) and their mark. Then, it shows a sample board with numbered squares (0-9) as a reference for placing the marks.

At each players' turn, the application checks the game status.

- If the player is the winner or there are no available squares, the application prints this information and prompts the user to play a new game or quit.
- If instead the game is still on, the player chooses one of the available squares for placing their mark, and the application prints the updated board.

Example

```
Welcome to Tic Tac Toe!
Player1 automatic gameplay? (y/n): n
Player1 pick a marker ['O', 'X']: x
Player2 automatic gameplay? (y/n): y

Positions in the board:
        -------------------
        |     |     |     |
        |  0  |  1  |  2  |
        |     |     |     |
        |-----------------|
        |     |     |     |
        |  3  |  4  |  5  |
        |     |     |     |
        |-----------------|
        |     |     |     |
        |  6  |  7  |  8  |
        |     |     |     |
        |_________________|
Player1 (X) select position: 4
        ___________________
        |     |     |     |
        |     |     |     |
        |     |     |     |
        |-----------------|
        |     |     |     |
        |     |  X  |     |
        |     |     |     |
        |-----------------|
        |     |     |     |
        |     |     |     |
        |     |     |     |
        |_________________|

Player2 (O) select position: 1
        ___________________
        |     |     |     |
        |     |  O  |     |
        |     |     |     |
        |-----------------|
        |     |     |     |
        |     |  X  |     |
        |     |     |     |
        |-----------------|
        |     |     |     |
        |     |     |     |
        |     |     |     |
        |_________________|

Player1 (X) select position: 1
Position 1 not available.
Player1 (X) select position: 11
Position 11 not allowed.
Player1 (X) select position: 0
        ___________________
        |     |     |     |
        |  X  |  O  |     |
        |     |     |     |
        |-----------------|
        |     |     |     |
        |     |  X  |     |
        |     |     |     |
        |-----------------|
        |     |     |     |
        |     |     |     |
        |     |     |     |
        |_________________|
Player2 (O) select position: 6
        -------------------
        |     |     |     |
        |  X  |  O  |     |
        |     |     |     |
        |-----------------|
        |     |     |     |
        |     |  X  |     |
        |     |     |     |
        |-----------------|
        |     |     |     |
        |  O  |     |     |
        |     |     |     |
        |_________________|

Player1 (X) select position: 8
        -------------------
        |     |     |     |
        |  X  |  O  |     |
        |     |     |     |
        |-----------------|
        |     |     |     |
        |     |  X  |     |
        |     |     |     |
        |-----------------|
        |     |     |     |
        |  O  |     |  X  |
        |     |     |     |
        |_________________|

Player1 wins!
Do you want to play again (y/n)?
```

Implementation notes

- The board can be implemented using either a 1-dimensional array or a 2-dimensional one. You do not need to store the graphical structure of the board.
- Try to keep the board visualization and the gameplay logic decoupled.
- For the automatic gameplay, you can use the `rand` function, scaled to the 0-9 range, to choose one of the available squares.
- Use functions to keep your code clear and organized.