Modular Programming in C

Course: Introduction to Programming and Data Structures

Dr. Laltu Sardar

Institute for Advancing Intelligence (IAI),
TCG Centres for Research and Education in Science and Technology (TCG Crest)

tcg crest
Inventing Harmonious Future

1. Modular Programming in C
   - Project Management in C

tcg crest
Inventing Harmonious Future

Modular Programming

What is Modular Programming?

- Technique of dividing a program into smaller, manageable parts called modules.
- Each module performs a specific task and can be developed, tested, and debugged independently.
- Enhances readability, maintainability, and reusability of code.

tcg crest
Inventing Harmonious Future

Advantages of Modular Programming

- **Separation of Concerns:** Each module handles a specific part of the functionality.
- **Code Reusability:** Modules can be reused in different programs.
- **Improved Maintainability:** Changes to one module can be made independently.
- **Parallel Development:** Multiple developers can work on different modules simultaneously.

tcg crest
Inventing Harmonious Future

Structure of a C Program in Modular Approach

- **Header Files:** Contain function declarations and macros (e.g., '.h' files).
- **Source Files:** Contain function definitions (e.g., '.c' files).
- **Main File:** Orchestrates the program by calling various modules.
- **Compilation:** Each module can be compiled separately and linked later.

tcg crest
Inventing Harmonious Future

Example of a Modular C Program

- Example of a header file: ‘mathutils.h‘

```
1  #ifndef MATHUTILS_H
2  #define MATHUTILS_H
3
4  int add(int, int);
5  int subtract(int, int);
6
7  #endif
```

Example of Implementation File: mathutils.c

```c
#include "mathutils.h"

int add(int a, int b) {
    return a + b;
}

int subtract(int a, int b) {
    return a - b;
}
```

Main Program Using Modular Components

```c
#include <stdio.h>
#include "mathutils.h"

int main() {
    int result = add(10, 5);
    printf("Result: %d\n", result);
    return 0;
}
```

What is Project Management?

- Organizing and managing resources (code, documentation, tools) to develop software efficiently.
- Involves managing timelines, code versions, testing, and debugging.
- Use of tools like Makefiles, version control (Git), and continuous integration systems.

Using Makefiles

- Automate the build process.
- Define rules for compiling and linking different modules.

```
# Example Makefile

all: main

main: main.o mathutils.o
gcc -o main main.o mathutils.o

main.o: main.c mathutils.h
gcc -c main.c

mathutils.o: mathutils.c mathutils.h
gcc -c mathutils.c

clean:
rm -f *.o main
```

tcg crest
Inventing Harmonious Future

What are Header Guards?

- Header guards prevent multiple inclusion of the same header file.
- This avoids issues like redefinition errors.
- Uses preprocessor directives `#ifndef`, `#define`, and `#endif`.

Example of Header Guards

- Modified Header Guard Example:

```
#ifndef MY\_LIBRARY\_H
#define MY\_LIBRARY\_H

// Declarations or definitions go here.

#endif /* MY\_LIBRARY\_H */
```

- #ifndef: Checks if MY_LIBRARY_H is not defined.
- #define: Defines the macro if it's not already defined.
- #endif: Ends the conditional inclusion.

tcg crest
Inventing Harmonious Future

Why Use Header Guards?

- Prevents multiple definition errors.
- Ensures each header file is included only once in a compilation unit.
- Improves compilation efficiency and avoids subtle bugs.

Version Control with Git

- Git helps in tracking changes and collaborating on code with multiple developers.
- Features:
  - **Commit:** Save changes with a description.
  - **Branching:** Work on multiple features simultaneously.
  - **Merging:** Combine changes from different branches.

tcg crest
Inventing Harmonious Future

tcg crest

Inventing Harmonious Future

Dr. Laltu Sardar

laltu.sardar@tcgcrest.org

https://laltu-sardar.github.io.