SDM COLLEGE OF ENGINEERING AND TECHNOLOGY

Dhavalagiri, Dharwad-580002, Karnataka State, India.

Email: cse.sdmcet@gmail.com

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

A Report on

MINOR WORK FOR CTA

COURSE CODE: 22UHUC500

COURSE TITLE: Software Engineering & Project Management

SEMESTER: 5th DIVISION: B

COURSE TEACHER: Dr. U.P.Kulkarni

[Academic Year- 2023-24]

Date of Submission: 18/10/2024

Submitted By

Ms. Shradha S Naik USN: 2SD22CS095

TABLE OF CONTENTS

1. Write a C program to prove that C programming language supports only Call by value.

```
#include <stdio.h>

// Function that modifies the value of an integer using a pointer

void modifyValue(int *ptr) {

    *ptr = 100;

    // Changing the value at the memory address of ptr

}

int main() {

    int a = 10;

    printf("Before calling modifyValue: a = %d\n", a);

    modifyValue(&a);

    // Passing the address of 'a' to modifyValue

    printf("After calling modifyValue: a = %d\n", a);

    // Check the value of 'a' after calling the function

    return 0;

}
```

OUTPUT:
Before calling modifyValue: a = 10
After calling modifyValue: a = 100

C does not support "call by reference" directly as some other languages like C++ do, where you can pass references of variables (not their values).

However, **pointers** in C allow you to simulate call by reference. When you pass the **address** of a variable (a pointer), the function can indirectly modify the original variable. This is sometimes referred to as "call by reference" by programmers, but technically it is still **call by value**, where the value being passed is the memory address of the variable.

Dennis Ritchie's Perspective:

C passes everything by value, but when you pass the address of a variable (i.e., a pointer), the function can manipulate the original data through the pointer. This is often confused with call by reference because the original data can be modified, but it's still a form of "value" being passed (the address itself).

In other programming languages like C++, you can pass arguments by reference by using the **reference operator (&)**. When a function takes a reference to a variable as a parameter, any modifications made to that reference affect the original variable.

2. Study the concept USABILITY. Prepare a report on usability of atleast two UIs of major software products you have seen.

Usability refers to how easy and efficient it is for users to interact with a system. It is a key aspect of User Experience (UX) and focuses on ensuring that users can achieve their goals effectively, efficiently, and satisfactorily.

Usability analysis:

I. Canva

Canva is a popular online design tool that allows users to create a wide range of graphics, including social media posts, presentations, posters, and more. Its appeal lies in its simplicity and ease of use, especially for non-designers.

A. **Easy to Learn:** Even beginners can start designing quickly because Canva is intuitive and provides templates to help get started.

B. **Efficient :** We can create designs fast using its drag-and-drop tools and pre-made elements like fonts, images, and graphics

C. **Accessible :** Canva works on both computers and mobile devices, and there is no need to install anything. Everything is stored online, so we can access our work anywhere.

D. **Error-Friendly :** Canva has undo/redo buttons, so fixing mistakes is easy, and the layout tools help prevent errors like misalignments.

E. **Satisfying to Use :** It's designed to be enjoyable, with plenty of creative options, making it great for both beginners and non-designers.

II. Google Drive

Google Drive is a cloud storage service developed by Google that allows users to store files online and access them from anywhere.

A. **Easy to Learn :** Google Drive is straightforward to use, even for those who aren't tech-savvy. The layout is clean, making it easy to find and organize files.

B. **Efficient :** You can quickly upload, store, and access your files from any device with internet access. It also allows us to create and edit documents, spreadsheets, and presentations directly within the platform.

C. **Accessible :** Since Google Drive is cloud-based, we can access our files from anywhere, whether on a computer, tablet, or smartphone. We don't have to worry about losing your files as they're saved automatically.

D. **Collaboration Friendly :** Google Drive makes it easy to share files with others and work on them together in real-time. We can leave comments, suggest edits, and see changes instantly.

E. **Error Prevention :** Google Drive automatically saves our work, reducing the risk of losing progress. It also allows you to recover previous versions of files if we make mistakes.

3. List all the features programming language and write programs to show how they help to write a robust code.

I. Strong Typing

Programming languages enforces type checking at compile-time or runtime. Variables are bound to a specific data type, and operations on incompatible types result in errors. This feature helps prevent unintended behaviors and bugs by ensuring that data types are handled correctly.

Python code:

```
def add_numbers(a: int, b: int) -> int:

return a + b

print(add_numbers(5, 10))  # Valid

print(add_numbers("5", "10"))  # Raises TypeError
```

II. Error Handling

Error handling mechanisms, such as exceptions, allow programs to deal with unexpected situations gracefully. Instead of crashing, programs can catch errors, log them, and respond appropriately, ensuring a better user experience.

Java code:

```
public class ErrorHandlingExample {
        public static void main(String[] args) {
         try {
                 int result = divide(10, 0);
                System.out.println(result);
              } catch (ArithmeticException e) {
                System.out.println("Cannot divide by zero: " + e.getMessage());
              }
 }
 public static int divide(int a, int b) {
          return a / b;
          }
 }
```

III. Modularity

Modularity refers to organizing code into separate, manageable units such as functions, classes, or modules. This approach enhances code readability and maintainability, making it easier to understand, test, and modify individual parts of a program.

Python code:

```
function calculateArea(radius) {
        return Math.PI * radius * radius;
}
function calculateCircumference(radius) {
        return 2 * Math.PI * radius;
}
const radius = 5;
console.log("Area:", calculateArea(radius));
console.log("Circumference:", calculateCircumference(radius));
```

IV. Comments and Documentation

Comments and documentation are essential for maintaining and understanding code. They provide context and explanations for how and why specific parts of the code work. Good documentation helps other developers (and the original author) quickly grasp the purpose and functionality of the code.

C Code:

```
// Function to calculate the factorial of a number

int factorial(int n) {

    if (n == 0) return 1; // Base case

    return n * factorial(n - 1); // Recursive case

}
```

V. Version Control

Version control systems (like Git) track changes to code over time, allowing developers to collaborate, revert to previous versions, and maintain a history of modifications. This is crucial for team projects and helps manage the complexity of development.

Commands:

```
# Initialize a new Git repository

        git init

# Stage changes

        git add .

# Commit changes

        git commit -m "Initial commit"
```

4. Study the ASSERTIONS in C language and its importance in writing RELIABLE CODE. Study POSIX standard and write a C program under unix to show use of POSIX standard in writing portable code.

Assertions in C are used to ensure that certain conditions hold true while a program is running. They are mainly used for debugging purposes to check for logical errors in the code during development. If an assertion fails (i.e., the condition is false), the program terminates and displays an error message, helping you catch bugs early.

Importance of Assertions in Writing Reliable Code:

I. **Error detection**: Assertions help catch unexpected behaviors or errors during the development phase.
II. **Debugging**: They provide information about which part of the code caused an issue, making debugging easier.
III. **Assumptions validation**: Assertions verify that conditions assumed to be true by the developer are indeed true.
IV. **Prevent serious failures**: By catching problems early, assertions prevent larger issues from occurring later.

Example of Assertion in C :

```
#include <stdio.h>

#include <assert.h>

int divide(int numerator, int denominator) {

        assert(denominator != 0);

        // Assert to check that denominator is not zero

        return numerator / denominator;

}

int main() {

         int a = 10, b = 0;
```

```
    // This will fail because denominator is 0, and assert will terminate the
program

    printf("Result: %d\n", divide(a, b));

    return 0;

}
```

OUTPUT :

Assertion failed: (denominator != 0), function divide, file <filename>, line <line number>.

Abort trap: 6

How is assert different from if-condition :

assert is a **development tool** used to validate assumptions and detect programming errors early. It is typically disabled in production.**if condition** is a **control structure** used for decision-making during the normal execution of the program. It is part of the program's logic and cannot be disabled in production.

POSIX (Portable Operating System Interface) is a set of standards that define how applications should interact with the operating system, making code portable across different Unix-like systems such as Linux and macOS. Using POSIX ensures that programs can run on any operating system that adheres to these standards, without needing platform-specific code.

Features of POSIX :

I. **Portability**: Ensures programs can run on different Unix-like systems.

II. **Standardized interfaces**: Provides a consistent way to perform tasks like process control, file management, and threading.

III. **Compatibility**: Code written with POSIX standards can be compiled and run on any POSIX-compliant system

Code

```
#include <stdio.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/wait.h>
int main() {
    pid_t pid = fork();  // Create a new process using POSIX fork()

    if (pid < 0) {
        // Fork failed
        perror("Fork failed");
        return 1;
    } else if (pid == 0) {
        // Child process
        printf("This is the child process, executing 'ls' command.\n");
        execl("/bin/ls", "ls", NULL);  // POSIX exec to run the 'ls' command
    } else {
        // Parent process
        wait(NULL);  // Wait for the child process to finish
        printf("Child process finished, now returning to parent.\n");
    }
return 0;
}
```

The above C program that uses POSIX functions like fork() to create a new process and execl() to execute another program. This program can be run on any Unix-based system, demonstrating POSIX-compliant code.