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

COURSE CODE : 22UHUC500

COURSE TITLE :Software Engineering and Project Management

SEMISTER : 5th DIVISION : CSE 'B'

COURSE TEACHER : Dr. U.P Kulkarni

[Academic Year- 2024-25]

Date of Submission: 18-10-2024

Submitted

By

Mr. Sumpreetsing S Rajaput USN : 2SD22CS112

CONTENTS

TASK – 1 : Write a C program to show that C programming language supports only call by value.

```
#include <stdio.h>
// Function which is trying to modify pointer itself
void modifyPointer(int *p) {
    int temp = 100;
    p = &temp;  // This will modify the original value
    *p = 200;    // Modifies the value pointed to by the local
pointer (not the original address)
    printf("Inside function modifyPointer: *p = %d (Address of p:
%p)\n", *p, p);
}
int main() {
    int a = 10;
    int *ptr = &a;  // Pointer to the variable a
    printf("Before calling modifyPointer: a = %d, *ptr = %d
(Address of ptr: %p)\n", a, *ptr, ptr);
    modifyPointer(ptr); //function call by passing the value
    //checking the values after function call
    printf("After calling modifyPointer: a = %d, *ptr = %d
(Address of ptr: %p)\n", a, *ptr, ptr);
    return 0;
}
```

The above code demonstrates how 'C' supports only "Call by Value".

Inside main() function:

- 'a' is a variable that holds the value 10.
- 'ptr' is the pointer that points to the address of 'a'.

Inside modifypointer() function:

- 'p' is the copy of the original pointer 'ptr' when we change p=&temp we only modify the copy not the original pointer 'ptr' so 'ptr' in 'main()' is unaffected.
- Now we modify the value of address pointed to by 'p' since 'p' is now points to a local variable(temp) it does not affects the values in the 'main()' function.

After the function call the value of 'a' and 'ptr' remained unchanged this tells that 'C' only supports call by value.

TASK – 2 : Study the concept of "USABILITY".Prepare a report on USABILITY of atleast TWO UIs of major software product you have seen.

INTRODUCTION :

Usability refers to the ease with which users can interact with a product, system, or service. It encompasses various aspects that contribute to a user's experience, including learnability, efficiency, memorability, error frequency and severity, and user satisfaction. Effective usability is crucial for ensuring that products meet user needs and provide a positive experience.

KEY USABILITY PRINCIPLES :

- **Learnability:** The product should be easy for new users to learn and understand. Intuitive design helps reduce the learning curve.
- **Efficiency:** Once users have learned the design, they should be able to perform tasks quickly and with minimal effort.
- **Memorability:** Users should be able to return to the product after a period of not using it and easily regain their proficiency.
- **Error Management:** The product should minimize user errors, provide clear error messages, and offer easy recovery options.
- **Satisfaction:** Users should have a positive experience with the product, feeling that it meets their needs and expectations.

USABILITY ANALYSIS OF TWO MAJOR SOFTWARE PRODUCTS:

1. Google Chrome

Overview :

Google Chrome is a widely used web browser known for its speed, simplicity, and robust performance. It serves millions of users worldwide across various platforms, including Windows, macOS, Linux, Android, and iOS.

Key Usability Features :

1. **Learnability**

- **Intuitive Interface:** Chrome offers a clean and minimalistic design with straightforward navigation. New users can easily understand how to open tabs, bookmark pages, and access settings without extensive guidance.
- **Onboarding Tutorials:** For first-time users, Chrome provides helpful prompts and tooltips that guide them through essential features, enhancing the initial learning experience.

2. **Efficiency**

- **Fast Performance:** Known for its quick page loading times and efficient resource management, Chrome allows users to browse the web seamlessly.
- **Extensions and Shortcuts:** A vast library of extensions enables users to customize their browsing experience, while keyboard shortcuts streamline common tasks, boosting productivity.

3. **Memorability**

- **Consistent Design Across Devices:** Chrome maintains a uniform interface across different platforms, making it easy for users to switch devices without relearning the layout or functionality.

- **Sync Feature:** Users can synchronize bookmarks, history, and settings across multiple devices, ensuring a consistent and memorable user experience.

4. **Error Management**

- **Clear Error Messages:** When issues arise, such as failed page loads or connectivity problems, Chrome provides clear and actionable error messages.
- **Recovery Options:** Features like session restore help users recover their previous browsing sessions after unexpected shutdowns or crashes.

5. **Satisfaction**

- **User-Friendly Experience:** The combination of speed, simplicity, and customization options leads to high user satisfaction.
- **Regular Updates:** Frequent updates ensure that Chrome remains secure, feature-rich, and aligned with user needs, contributing to sustained user trust and approval.

2. Adobe Photoshop

Overview :

Adobe Photoshop is a leading graphic design and photo editing software used by professionals and enthusiasts alike. It offers a comprehensive set of tools for image manipulation, graphic design, and digital art creation.

Key Usability Features:

1. **Learnability**

- **Extensive Tutorials and Documentation**: Adobe provides a wealth of tutorials, guides, and community forums to help new users learn Photoshop's complex features.
- **Customizable Workspace**: Users can customize their workspace to focus on the tools they use most, reducing the initial overwhelm and facilitating easier learning.

2. **Efficiency**

- **Advanced Tools and Shortcuts**: Photoshop offers a wide array of powerful tools and keyboard shortcuts that enable experienced users to perform complex tasks quickly and efficiently.
- **Automation Features**: Features like actions and scripts allow users to automate repetitive tasks, enhancing productivity.

3. **Memorability**

- **Consistent Interface**: The user interface remains consistent across different versions, helping users retain their knowledge and skills over time.
- **Modular Tool Panels**: Tools are organized into panels that can be customized and rearranged, aiding users in remembering where specific tools are located based on their personalized layout.

4. **Error Management**

- **Non-Destructive Editing**: Features like layers and adjustment layers allow users to make changes without permanently altering the original image, making it easy to undo mistakes.
- **History Panel**: The history panel lets users step back through previous actions, providing a safety net for error correction.

5. **Satisfaction**

- **High Customizability and Flexibility**: Photoshop's extensive feature set meets the diverse needs of professionals, leading to high satisfaction among its user base.
- **Community and Support**: A large community of users and ample support resources contribute to a positive user experience, although the software's complexity can be daunting for beginners.

CONCLUSION :

Usability is a critical aspect of any software product, ensuring that users can effectively and efficiently achieve their goals with minimal frustration. By focusing on key features like learnability, efficiency, memorability, error management, and user satisfaction, products become more intuitive and user-friendly.

TASK – 3 : “List the features of programming language and write programs to show how they help to write robust code.”

INTRODUCTION:

A programming language is a formal set of instructions used to communicate with computers to perform specific tasks. It allows developers to write code that the machine can interpret and execute.

Here are some of the Key features of Programming languages

1. Strong Typing
2. Error Handling
3. Modularity
4. Code Reusability
5. Standard Libraries

ILLUSTRATION OF FEATURES :

1. Strong Typing :

Strong typing in programming languages refers to the enforcement of strict rules on how different types of data can interact with one another. In strongly typed languages, each variable, expression, and value has a specific data type (such as integer, float, string, etc.), and operations involving mismatched data types are either disallowed or require explicit conversion. This feature helps catch errors at compile-time or runtime, reducing bugs caused by unintended type conversions or misuse of data types

Here is a java code that demonstrates **Strong Typing**, where operations between mismatched data types require explicit handling, preventing unintended errors.

```
public class Main {
    public static void main(String[] args) {
        int number = 10;
        String text = "Hello";
         // This line will cause a compile-time error because of
type mismatch:
        // String result = number + text; // Error: incompatible
types: int cannot be converted to String
        // Correct approach: explicit conversion of int to String
          String result = text + Integer.toString(number); //
Explicit conversion of int to String
        System.out.println(result);  // Output: Hello10
    }
}
```

2. Error Handling :

Error handling in programming refers to the methods and practices used to manage and respond to errors that occur during the execution of a program. Errors can arise from various sources, such as invalid user input, accessing unavailable resources, or attempting operations that the system cannot handle (like division by zero or accessing out-of-bounds array elements).

Here is a java code that demonstrates **Error Handling** feature

```
public class Main {
    public static void main(String[] args) {
        int[] numbers = {1, 2, 3};
        try {
            // This line will throw an
ArrayIndexOutOfBoundsException
            System.out.println(numbers[5]);  // Trying to access
index 5, which is out of bounds
        } catch (ArrayIndexOutOfBoundsException e) {
            System.out.println("Error: Tried to access an
invalid index.");
        } finally {
            System.out.println("This block always executes, even
if there's an exception.");
        }
        System.out.println("Program continues normally after
handling the error.");
    }
}
```

3. Modularity :

Modularity in programming refers to the design principle of breaking down a large program into smaller, independent, and manageable units or modules. Each module performs a specific part of the overall functionality and can be developed, tested, and maintained separately. Modularity promotes code reuse, ease of debugging, maintainability, and scalability.

Here ia the java code that demonstrates how it supports modular programming through classes and methods.

```
public class Main {
    public static double calculateArea(double radius) {
        return Math.PI * radius * radius;
    }
    public static double calculateCircumference(double radius) {
        return 2 * Math.PI * radius;
    }
    public static void main(String[] args) {
        double radius = 5.0;
        System.out.println("Area: " + calculateArea(radius)); // 
Output: Area: 78.53981633974483
        System.out.println("Circumference: " + 
calculateCircumference(radius)); // Output: Circumference: 
31.41592653589793
    }
}
```

4. Code Reusability :

Code reusability is a fundamental principle in software development that emphasizes the practice of using existing code in multiple applications or parts of a program without modification. It allows developers to avoid redundancy, reduce errors, and save time by leveraging previously written and tested code.

Here is the java code that tells how methods in it are reused across the different parts of code

```
public class Main {
    public static void greetUser(String name) {
        System.out.println("Hello, " + name + "!");
    }
    public static void main(String[] args) {
        greetUser("Alice"); // Output: Hello, Alice!
        greetUser("Bob");   // Output: Hello, Bob! }
}
```

5. Standard Libraries :

Standard libraries are a collection of pre-written code and functions provided by a programming language to perform common tasks. These libraries offer a wide range of functionalities that facilitate development, allowing programmers to write code more efficiently without needing to implement everything from scratch. Standard libraries cover various areas, including data structures, algorithms, file handling, networking, and graphical user interfaces.

Here is the code that demonstrates how java provides a rich set of standard libraries that reduse the need for custome implementation

```
import java.util.ArrayList;
public class Main {
    public static void main(String[] args) {
        ArrayList<String> fruits = new ArrayList<>();
        fruits.add("Apple");
        fruits.add("Banana");
        fruits.add("Cherry");
         System.out.println(fruits); // Output: [Apple, Banana, 
Cherry]
    }
}
```

CONCLUSION :

In conclusion, the features of programming languages, such as **Strong Typing**, **Error Handling**, **Modularity**, **Code Reusability**, and **Standard Libraries**, collectively contribute to the development of robust, efficient, and maintainable software.

TASK – 4: Study the 'ASSERTION' 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.

ASSERTION

INTRODUCTION :

In C programming, assertions are a debugging aid that allows programmers to test assumptions made during program execution. An assertion is a statement that checks a condition, and if the condition evaluates to false, the program displays an error message and halts execution. Assertions are implemented using the assert() macro, which is part of the <assert.h> header file.

Declaration : void assert(expression);

Example 1:

```
#include <stdio.h>

#include <assert.h>

int main()

{

    int a,b; //declare two int variables

    printf("Enter the values of 'a' and 'b' : \n"); //read the
values from user

    scanf("%d%d",&a,&b);

    assert(b!=0); //check the condition b not equals to zero

    float res=(a/b);

    printf("Result of division is : %f",res); //if assertion
successfull print the result

}
```

Sample Output 1:

```
Enter the values of a and b :
4 2
Result of division is : 2.000000
```

Sample Output 2:

```
Enter the values of 'a' and 'b' :
3 0
Assertion failed!
Program: C:\Users\sumpr\OneDrive\Desktop\CSE\c-coding\C\Assertion.exe
File: Assertion.c, Line 8
```

Example 2:

```c
#include <stdio.h>
#include <assert.h>

int factorial(int n){
    assert(n>=0);  //assertion to check non negative number
    if(n==0 || n==1){
        return 1;
    }
    else{
        return n*factorial(n-1);
    }
}
int main(){
    int num;
```

```
     printf("Enter the number for which you want to calculate
factorial :\n");
    scanf("%d",&num);
    int res=factorial(num);
    printf("factorial of given number %d is : %d",num,res);
}
```

Sample Output 1:

```
Enter the number for which you want to calculate factorial :
4
factorial of given number 4 is : 24
```

Sample Output 2:

```
Enter the number for which you want to calculate factorial :
-9
Assertion failed!
Program:C:\Users\sumpr\OneDrive\Desktop\CSE\ccoding\C\Assertion_exampl
e2.exe
File: Assertion_example2.c, Line 5
```

- In **Example1** the `assert(b!=0)` ensures division operation is performed only when denominator not equals to '0'.
- In **Example2** the `assert(n>=0)` ensures factorial function is called only for non-negative numbers.

IMPORTANCE OF ASSERTIONS IN WRITING RELIABLE CODES :

- Assertion can detect the bugs as soon as they occur,preventing them from propagating further into the code.
- When an assertion fails, it provides clear indication of where the problem lies making it easier to identify and fix the root cause.
- It can be used to enforce preconditions and postconditions. This helps to ensure that the code is behaving as expected.
- Using of assertions can encourage programmers to write more robust and maintainable code.

In conclusion we can say that using assertions can significantly improve the quality, reliability and maintainability of code.

POSIX

INTRODUCTION :

The POSIX standard is a family of standard for operating systems that define an API (Application Programming Interface) for programmers to use. It aims to provide a consistent environment for application development across different platforms, making it easier to port software between systems.

Key components of the POSIX standard include:

- **Shell :** A command line interpreter that provides a user interface for interacting with an operating system.
- **Utilities :** In the context of operating system refers to a set of built in programs that provide essential functionalities for managing files, processes and other system resources.

- **C Library :** A collection of functions that provide a standard interface for interacting with the operating system and performing common programming tasks in the C programming language.
- **System Calls :** These are the functions used by a computer program to request services from the operating system kernel. These services can include tasks such as file I/O, process management, memory allocation, and network communication.

C Program under UNIX to show use of POSIX standard :

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <fcntl.h>
int main() {
    // Open a file for reading
    int fd = open("my_file.txt", O_RDONLY);
    if (fd == -1) {
        perror("open");
        exit(EXIT_FAILURE);
    }
    // Read data from the file
    char buffer[100];
    ssize_t bytes_read=read(fd,buffer, sizeof(buffer));
    if (bytes_read == -1) {
        perror("read");
        exit(EXIT_FAILURE);
    }
    // Print the read data
    printf("Read %zd bytes:\n%s\n", bytes_read, buffer);
    // Close the file
    close(fd);
    return 0;
}
```

Sample Output :

```
Read 39 bytes:
sumpreetsing rajaput cse sdmcet dharwad
```