SDM College of Engineering and Technology

Dhavalagiri, Dharwad-580 002. Karnataka State. India.

Email: principal@sdmcet.ac.in, cse.sdmcet@gmail.com

Ph: 0836-2447465/ 2448327 Fax: 0836-2464638 Website: sdmcet.ac.in

Department

of

COMPUTER SCIENCE AND ENGINEERING

MINOR WORK

[21UCSC403-SOFTWARE ENGINEERING]

ODD Semester: Sept-Jan-2024-25

Course Teacher: Dr. U.P. Kulkarni

2024-25

Submitted By

MS. SHREYA V HIREMATH

2SD22CS100

5th Semester B division

Table of Contents

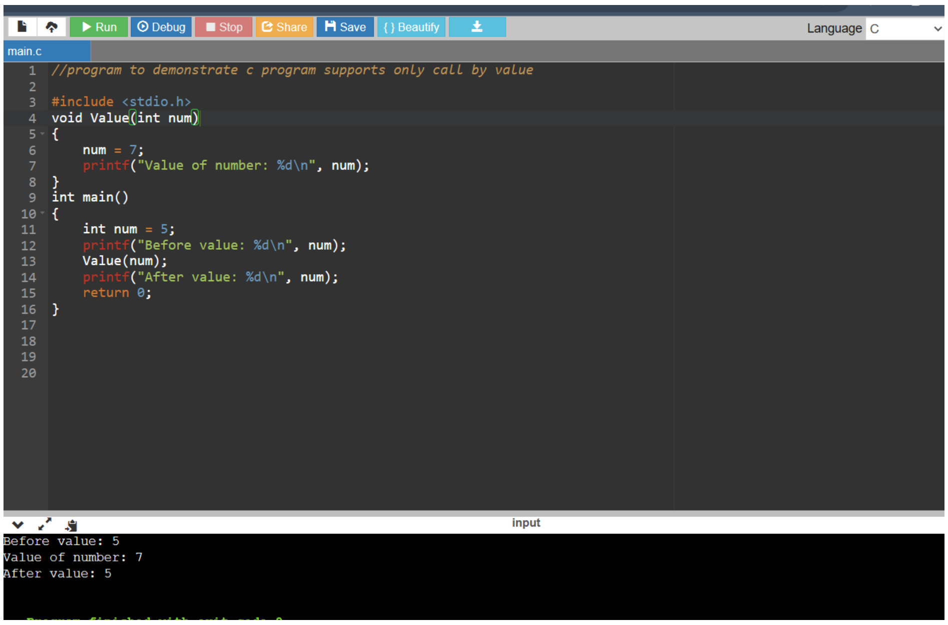
TERMWORK 1- Write a C program to show that C programming Language support only Call by Value.

There are two parameter passing methods passing (or calling) methods to functions. They are Call by Value and Call by Reference. The most significant distinction between call by value and call by reference is that in the call by value, passing an actual value of the parameter. While, in a call by reference, passing the reference of the parameter; hence some change made to formal arguments will also reflect in actual arguments. C Language only supports Call by Value.This means that the called function is given the values of its arguments in temporary variables rather than the originals. This leads to some different properties than are seen with "call by reference". In C, both the calling and called functions don't share any memory they have their copy, along with the called function can't directly change a variable in the calling function; it may only alter its private, temporary copy. Call by value is an asset, however, not a liability. It usually leads to more compact programs with fewer extraneous variables, because parameters can be treated as conveniently initialized local variables in the called routine.

C Program to demonstrate Call by Value:

```
//program to demonstrate c program supports only call by value

#include <stdio.h>
void Value(int num)
{
   num = 7;
   printf("Value of number: %d\n", num);
}
int main()
{
   int num = 5;
   printf("Before value: %d\n", num);
   Value(num);
   printf("After value: %d\n", num);
   return 0;
}
```



```c
//program to demonstrate c program supports only call by value

#include <stdio.h>
void Value(int num)
{
    num = 7;
    printf("Value of number: %d\n", num);
}
int main()
{
    int num = 5;
    printf("Before value: %d\n", num);
    Value(num);
    printf("After value: %d\n", num);
    return 0;
}
```

```
Before value: 5
Value of number: 7
After value: 5
```

This demonstrates that changes made to the parameter inside the function do not affect the original argument in the caller function. When a variable is passed as an argument to a function in C, a copy of its value is created and passed to the function. Therefore, any modifications made to the parameter inside the function are only applied to the local copy, and the original value of the argument remains unaffected. This behavior is known as "Call by Value" in C.

Here the value of num in the main function is declared as 5. A function Value is called. In that function the value of the num is initialized to 7. After the execution of the function is finished the control is given back to the main function. In the main function if we print the value of num, its value will the same as it was in the main function. The changes made in the function will not effect the value in the main function. Only the copy of the variable is sent to the function not the actual variable.

TERMWORK-2: Study the concept “USABILITY”. Prepare a report on USABILITY of at least TWO UIs of major software product you have seen.

Usability is a way to measure how easy a product is to use. It is a concept in design circles to ensure products—whether websites, mobile applications can be used as simply and painlessly as possible. Good usability means users can accomplish their tasks quickly, with minimal stress and errors, and ultimately feel satisfied in their interaction with a product. For websites in particular, usability is crucial. Visitors to a website can easily leave as soon as they encounter difficulty or confusion.

REPORT ON USABILITY OF TWO UIS OF MAJOR SOFTWARE PRODUCTS

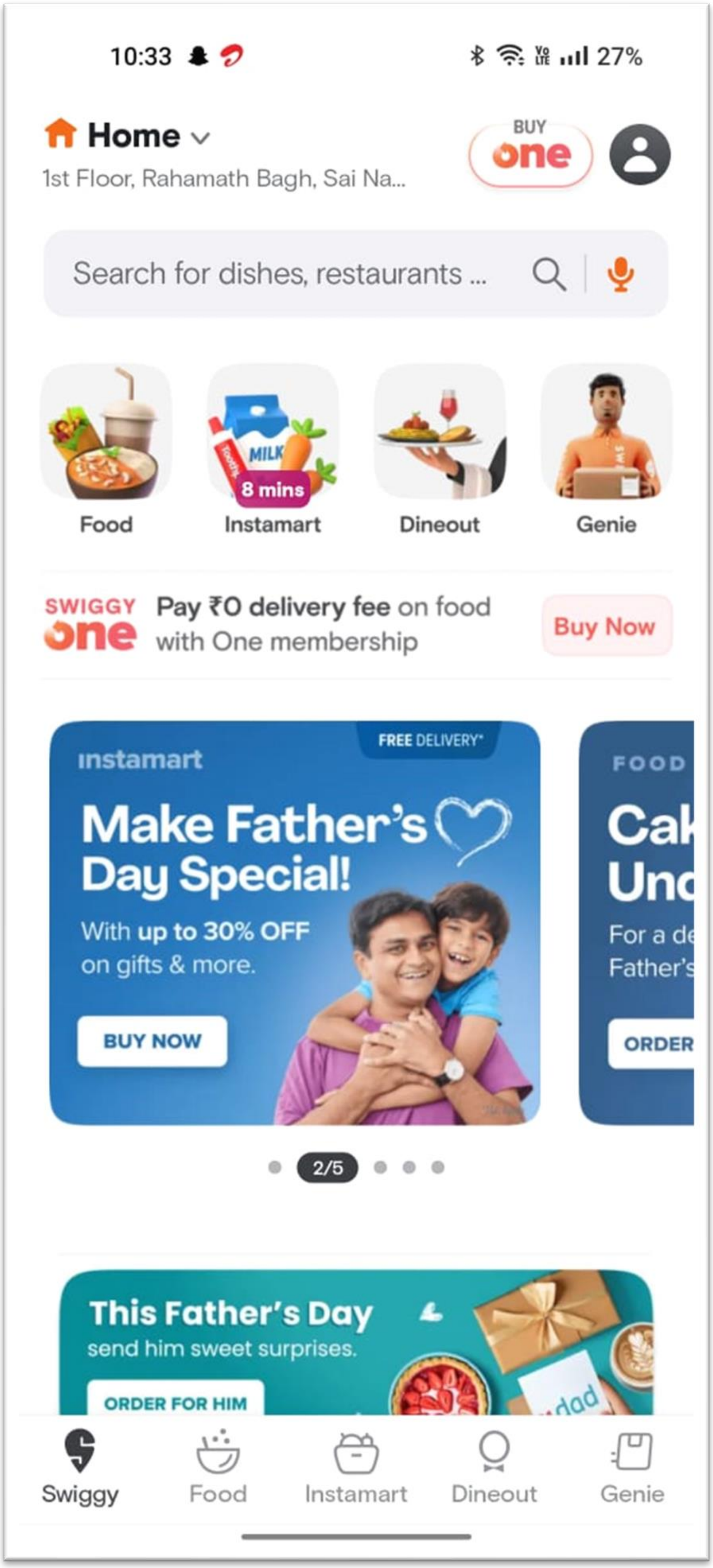
1.SWIGGY MOBILE APP

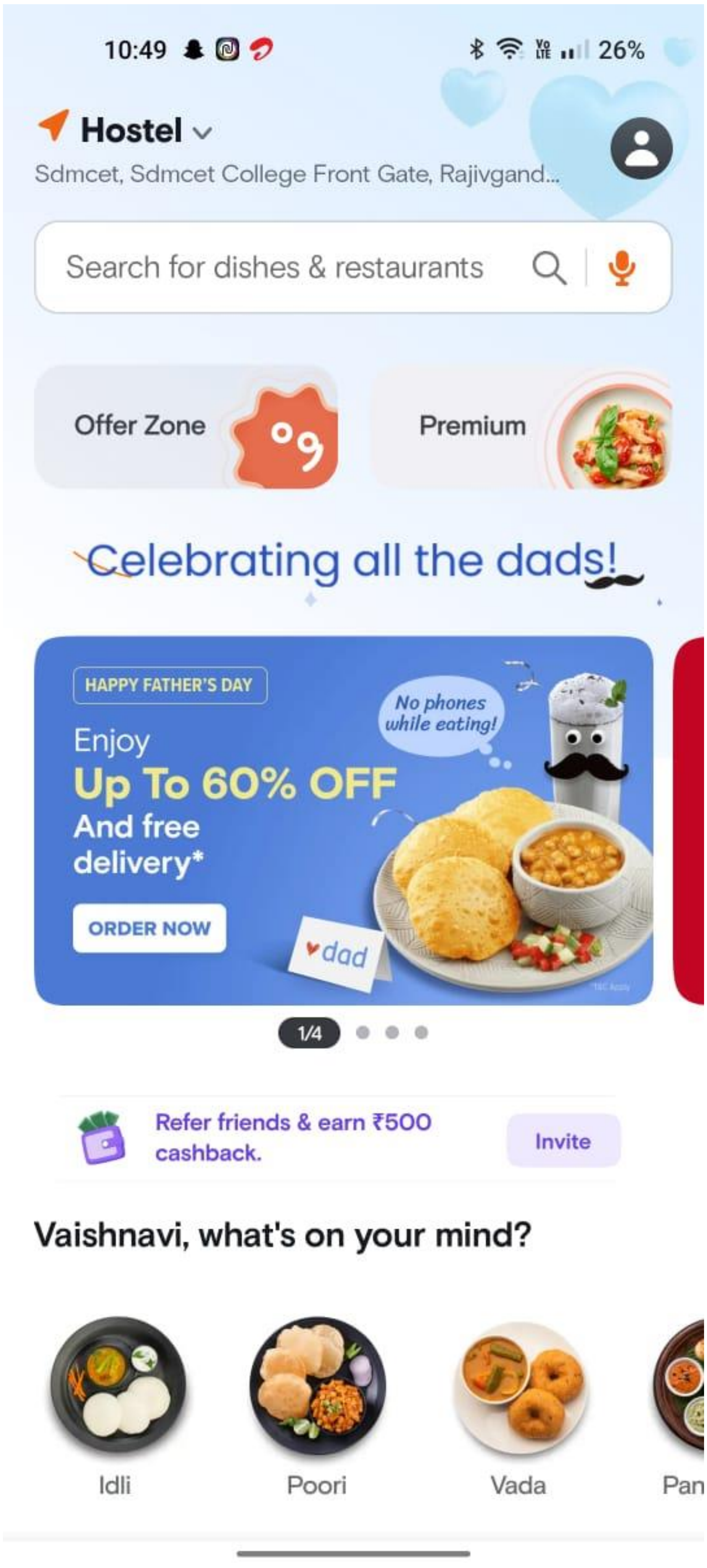
The Swiggy app is a popular food delivery platform that allows users to order food from a wide range of restaurants in their area. The aspects of it’s user interface(UI) are as follows

- Visual Design
  The visual design of the app is very modern and appealing. The design changes according to the location of the user. In cities the app provides extra facilities which are listed clearly in the layout.

- Navigation
  Navigation in the app is user friendly. The search bar is located at the top. The menu option, accounts, address selection etc are all properly labelled making it easy to navigate with one hand. The use of icons alongside the labels enhances recognition and usability.

- Search and Filtering
  The Swiggy app has a robust search and filtering system that helps users to find their desired restaurants and dishes. The search bar is displayed at the top of the screen, allowing users to quickly enter keywords or restaurant names. The app also offers various filters, such as cuisine type, price range, and delivery time, allowing users to refine their search results based on their preferences. It also allows voice enabled search.


10:33
27%
Home
1st Floor, Rahamath Bagh, Sai Na...
BUY one
Search for dishes, restaurants ...
8 mins
Food
Instamart
Dineout
Genie
SWIGGY one
Pay ₹0 delivery fee on food with One membership
Buy Now
instamart
FREE DELIVERY*
Make Father's Day Special!
With up to 30% OFF on gifts & more.
BUY NOW
FOOD
Cak
Unc
For a de
Father's
ORDER
2/5
This Father's Day
send him sweet surprises.
ORDER FOR HIM
Swiggy
Food
Instamart
Dineout
Genie



10:49 26%

Hostel

Sdmcet, Sdmcet College Front Gate, Rajivgand...

Search for dishes & restaurants

Offer Zone

Premium

Celebrating all the dads!

HAPPY FATHER'S DAY

Enjoy

Up To 60% OFF

And free delivery*

ORDER NOW

1/4

Refer friends & earn ₹500 cashback.

Invite

Vaishnavi, what's on your mind?

Idli

Poori

Vada

Pan

2. Google Search Engine

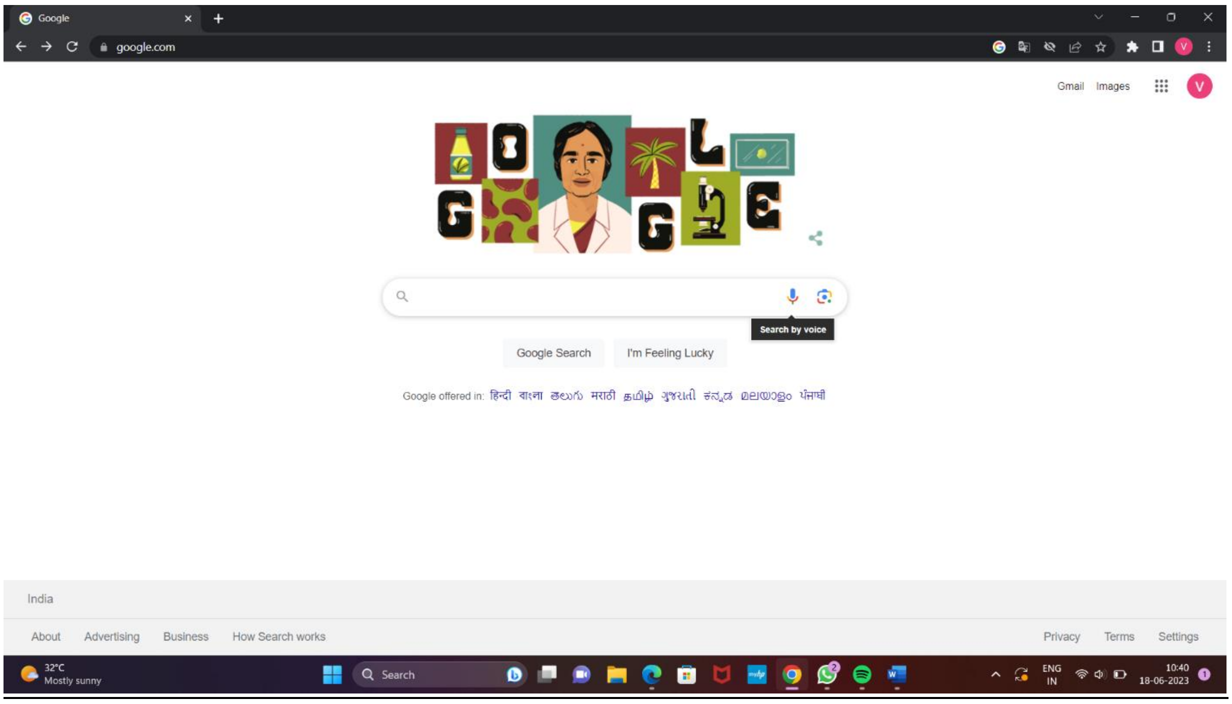
Google Search is one of the most widely used search engines globally, providing users with a vast amount of information and resources. The aspects of it's user interface (UI) are as follows

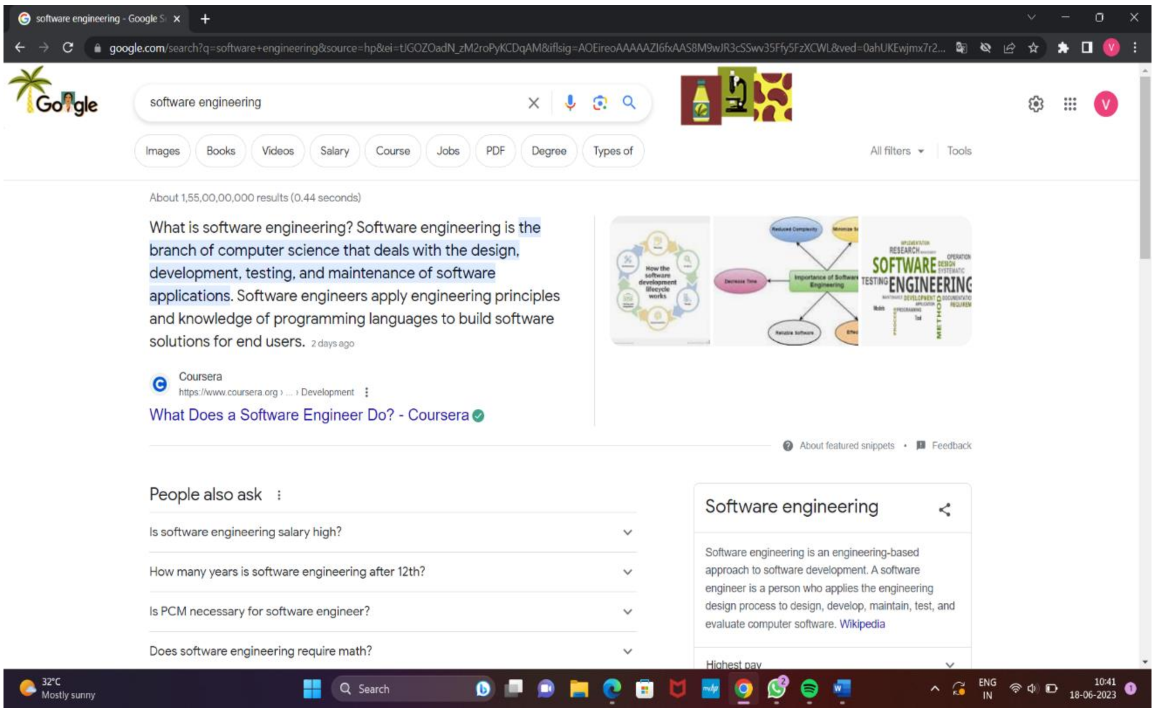
- Search Bar and Query Input
  The UI of Google Search prominently features a search bar at the center of the page. The search bar is highly visible, easily accessible, and provides autocomplete suggestions as users type. This feature enhances usability by assisting users in formulating their queries and saving time in the search process. It also enables image based search and voice enabled search.

- Search Results Page
  The search results page of Google Search is designed to be highly user-friendly and informative, presenting search results in a structured and organized manner. Google's search algorithm prioritizes the most relevant and authoritative results at the top of the search results page. The title of the webpage and its URL are displayed below the heading, giving users a better understanding of the source. Google displays a section of related searches at the bottom of the search results page. These related searches help users explore alternative or related topics.

- Image and Video Search
  Google Search has separate tabs for image and video searches which provides a user-friendly experience for visual content discovery. The UI for image and video search allows users to quickly assess the relevant results. The inclusion of filters, such as image size and color, enhances usability by enabling users to find the desired images or videos more effectively.

- Advanced Search and Filters
  Google Search offers advanced search options and filters that allow users to refine their search results. These options, such as date range, file type, and site-specific searches, are easily accessible through the search settings or advanced search page. The availability of these features enhances usability by enabling users to narrow down their searches and find more specific information. Google Search enables users to filter search results by language. This option is particularly beneficial for multilingual users or when searching for content in a specific language. Google Search's search results page includes a "Tools" button that allows users to access additional filters, such as sorting by relevance or date, and filter options for image searches, such as size, color, and usage rights. By using the verbatim search feature, users can instruct Google Search to include all the words in the search query exactly as typed.




software engineering - Google S
google.com/search?q=software+engineering&source=hp&ei=tJGOZOadN_zM2roPyKCDqAM8iflsig=AOEireoAAAAAZI6fxAAS8M9wJR3cSSwv35Ffy5FzXCWL&ved=0ahUKEwjmx7r2...
software engineering
Images
Books
Videos
Salary
Course
Jobs
PDF
Degree
Types of
All filters
Tools
About 1,55,00,00,000 results (0.44 seconds)

What is software engineering? Software engineering is the branch of computer science that deals with the design, development, testing, and maintenance of software applications. Software engineers apply engineering principles and knowledge of programming languages to build software solutions for end users. 2 days ago

Coursera
https://www.coursera.org › ... › Development

What Does a Software Engineer Do? - Coursera

About featured snippets • Feedback

People also ask

Is software engineering salary high?

How many years is software engineering after 12th?

Is PCM necessary for software engineer?

Does software engineering require math?

Software engineering

Software engineering is an engineering-based approach to software development. A software engineer is a person who applies the engineering design process to design, develop, maintain, test, and evaluate computer software. Wikipedia

Highest pay

TERMWORK-3 List all features of a programming language and write programs to show they help to write a robust code.

There are different types of programming languages. Based on their features they are categorized as procedural programming language, object-oriented programming language, Functional programming language, Scripting language and Logic Programming language. The features are language specific but few common features of a programming language are

- Syntax: Each programming language has its own syntax or set of rules that govern how programs are written and structured. Enforcing type safety helps catch type-related errors during compilation rather than at runtime, reducing the chances of unexpected behavior or crashes.
  Example: Basic syntax of C

```
//hello world
#include <stdio.h>
// Defining a main function
void main()
{
        // code
        printf("Hello World!");
}
```

- Modularity: Modularity refers to the concept of dividing a system or a complex entity into smaller, self-contained modules or components. Each module performs a specific function or task and can be developed, tested, and maintained independently of the other modules.
  Example of a module in C

```
//module to add two numbers
void add ( int a, int b)
{
        int sum;
        sum = a + b;
        printf(" Sum of numbers= %d",sum) ;
       return 0;
}
```

- Libraries: Libraries are collections of precompiled code and resources that provide specific functionality or services to developers. They are designed to simplify software development by offering reusable components, functions, classes, or modules that can be incorporated into applications or projects.

Example: Math library in C

```
//program to find square root of a number
#include <stdio.h>
#include <math.h>
int main()
 {
    float number = 25;
    float squareRoot = sqrt(number);
    printf("Square root of %.2f is %.2f\n", number, squareRoot);
    return 0;
}
```

- Control Structures: Languages offer control structures to control the flow of program execution. Common control structures include conditionals (if-else statements, switch statements) and loops (for loops, while loops) that allow branching and repetition.

  Example: Program showing the use of if else

```
//program to determine is a number is even or odd
#include <stdio.h>
int main()
 {
            int num;
             printf("Enter a number: ");
             scanf("%d", &num);
            if (num > 0)
            {
                        printf("The number is positive.\n");
             }
             else if (num < 0)
            {
                        printf("The number is negative.\n");
            }
            else
            {
                        printf("The number is zero.\n");
            }
            return 0;
}
```

TERMWORK-4 Study the ASSERTIONS in C and its importance in writing reliable code. Prepare an appropriate business scenario and implement the same.

Assertions are statements used to test assumptions made by programmers. They are an important tool for ensuring the correctness of your code in C. Assertions are statements that test for certain conditions that should always be true. If an assertion fails, the program will terminate immediately, allowing you to quickly identify and fix the problem.

Uses of Assertion:

1. Debugging
2. Input Validation
3. Program correctness
4. Testing and quality assurance
5. Error handling

Business scenario: Banking Transaction Validation

Suppose we are developing software for a banking application that handles customer transactions. The system allows users to deposit and withdraw money from their accounts. Our task is to implement the transaction functionality and ensure the validity of the transactions.

To write reliable code for the banking application, assertions can be utilized to validate various conditions and ensure the integrity of the transaction data.
The assertions check conditions such as:
1.The account pointer is not NULL.
2.The amount deposited or withdrawn is greater than zero.
3.The resulting balance after the transaction does not exceed the maximum allowed balance.

By using these assertions, the program can catch potential issues early on, such as invalid inputs or exceeding the account's balance limits, ensuring the reliability of the banking application. When running the program, the successful transactions will display appropriate messages with the updated balance. However, the invalid scenarios will trigger assertion failures, terminating the program and providing error messages indicating the specific conditions that were violated.

TERMWORK-5 Study the POSIX standard system calls and write a program to demonstrate it.

The Portable Operating System Interface (POSIX) is a family of standards specified by the IEEE Computer Society for maintaining compatibility between operating systems. POSIX defines both the system and user-level application programming interfaces (APIs), along with command line shells and utility interfaces, for software compatibility (portability) with variants of Unix and other operating systems. POSIX is also a trademark of the IEEE. POSIX is intended to be used by both application and system developers. The POSIX standard defines a set of system calls that provide an interface for interacting with the operating system. These system calls allow programmers to perform various tasks such as file I/O, process management, inter-process communication, and more.

POSIX standard system calls

- fork: Create a New Process

  fork() creates a new process by duplicating the current (parent) process. The system call creates a new child process with a different process ID (PID) but shares the code, stack, and heap with the parent. The child gets a return value of 0, while the parent receives the PID of the child.

- getpid : Get Process ID

  getpid() returns the process ID of the calling process. Knowing the process ID is useful for identifying processes and for debugging.

- getppid: Get Parent Process ID

  getppid() returns the process ID of the parent of the calling process. It's useful for determining the parent of a process.

- execvp: Execute a Program

  The execvp() system call replaces the current process image with a new process image. It is used to run another program in place of the current program (within the same process). This means that after execvp() is called, the current program stops running, and the new program starts executing.

- wait: Wait for Child Process

  The wait() system call makes the parent process wait until all of its child processes have terminated. It allows the parent to retrieve the exit status of the child. It ensures the parent process knows when the child finishes executing, which can be important for synchronization.

- open: The open() system call is used to open a file or create a new file in the file system. It provides a way to access files for reading, writing, or both, depending on the specified flags.

- close: The close() system call is used to close a file descriptor after you have finished using it. It releases the resources associated with the file descriptor and makes it available for reuse by the system.

Program to demonstrate fork(), execvp and waitid system calls

```
#include <stdio.h>

#include <unistd.h>

#include <sys/wait.h>

#include <fcntl.h>

#include <stdlib.h>

int main() {

    pid_t pid;

    int status;

    // Fork to create a child process

    pid = fork();

    if (pid == -1) {

        perror("fork failed");

        exit(1);

    }


    if (pid == 0) {

        // Child process

        printf("Child process (PID: %d, Parent PID: %d)\n", getpid(), getppid());

        // Using execvp to run the 'ls' command

            char *args[] = {"ls", "-l", NULL};

             execvp(args[0], args);

            // If execvp fails

            perror("execvp failed");

            exit(1);

            } else {

              // Parent process

                  printf("Parent process (PID: %d) waiting for child (PID: %d)\n", getpid(), pid);

             // Wait for the child process to finish

                  wait(&status);
```

```
if (WIFEXITED(status)) {
 printf("Child process terminated with status: %d\n", WEXITSTATUS(status));
 }
// Demonstrating file handling with POSIX calls
int fd = open("example.txt", O_CREAT | O_WRONLY | O_TRUNC, 0644);
if (fd == -1) {
            perror("Failed to open file");
            exit(1);
        }
// Writing to the file
const char *text = "Hello, POSIX world!\n";
write(fd, text, sizeof(char) * 20);
// Close the file
close(fd);
// Reading from the file
fd = open("example.txt", O_RDONLY);
if (fd == -1) {
            perror("Failed to open file");
            exit(1);
            }

char buffer[21];
read(fd, buffer, sizeof(char) * 20);
buffer[20] = '\0';  // Null-terminate the string
printf("File content: %s", buffer);
// Close the file again
```

```
            close(fd);
                }
    return 0;
}
```

```
OUTPUT:
Parent process (PID: 1234) waiting for child (PID: 1235)
Child process (PID: 1235, Parent PID: 1234)
... output of `ls -l` command ...
Child process terminated with status: 0
File content: Hello, POSIX world!
```