**PROGRAMMING ASSIGNMENT**

1. Create a C program to store and manage student details.

Registration number, 1st name, 2nd name, last name and units (intro to programming, networking, accounting, OS and Computer Applications)

1. Define the following terms as used in programming:
2. Compiler
3. Source code
4. Object code
5. Linkers
6. Using an example i.e. a program to add 2 numbers. Explain the compilation process of a C compiler
7. Explain the difference between a compiler and an interpreter. (at least 6 comparisons)
8. List all the main categories of operations available in C programming

**2.DEFINATION OF TERMS**

1. COMPILER

A software tool that translates high-level source code written by a programmer into machine code that can be executed by a computer.

Plays a crucial role in converting human-readable code into a format that a computer can understand and execute.

1. SOURCE CODE

Text based instructions written by a programmer to create computer instructions.

Written using syntax and rules of C programming and serves as starting point for the development process. Source code is later translated by a compiler into machine code , allowing the computer to execute instructions as intended by the programmer.

1. OBJECT CODE

Output generated by a compiler after translating the source code.

A low level representation of the program and consists of machine cod that is specific to the target platform. It is not meant to be easily understood by programmers but is designed for execution by the computer. It is always in form of binary format and is the result of the compilation process.

1. LINKERS

Tools that combine multiple object files generated by compiler into a single executable program.

They resolve references between different files, ensuring that functions and variables declared in one file can be properly used in another. They play a crucial role in final stages of turning source code into a runnable program.

**3.COMPILATION PROCESS OF A C COMPILER TO ADD 2 NUMBERS**

1. Preprocessing

Handles directives like #include and #define.

Expands macros and includes header files.

1. Compilation

Translates preprocessed code into an assembly language.

1. Assembly

Converts assembly code into machine/binary code.

1. Linking

Combines machine code with external libraries to produce the final executable.

Example:

#include <stdio.h>

int main() {

int num1, num2, sum;

printf("Enter two numbers: ");

scanf("%d %d", &num1, &num2);

sum = num1 + num2;

printf("Sum: %d\n", sum);

return 0;

}

**4.DIFFERENCES BETWEEN COMPILERS & INTERPRETORS**

|  |  |  |
| --- | --- | --- |
|  | **COMPILER** | **INTERPRETORS** |
| EXECUTION | Produces an executable file, and the entire program is translated before execution. | Translates and executes the source code line by line. |
| SPEED | Produces faster code as the entire program is optimized before execution. | May have a slower execution as it translates code on-the-fly. |
| PLATFORM DEPENDACY | Generates platform-specific executables, requiring recompilation for different platforms. | Platform-independent, as the interpreter itself is platform-specific. |
| ERROR DETECTION | Detects errors for the entire program before execution. | Halts at the first encountered error, making it easier to debug. |
| MEMORY USAGE | Usually results in a standalone executable file, potentially using more memory. | Typically requires less memory but needs the interpreter to be present. |
| PORTABILITY | Less portable, as different platforms may require recompilation. | More portable, as long as the interpreter is available for the target platform. |

**5.CATEGORIES OF OPERATORS AVAILABLE IN C PROGRAMMING**

1. **Arithmetic Operations**: Addition, subtraction, multiplication, division, modulus.
2. **Relational Operations:** Equality, inequality, greater than, less than, greater than or equal to, less than or equal to.
3. **Logical Operations:** AND, OR, NOT.
4. **Bitwise Operations:** Bitwise AND, OR, XOR, left shift, right shift.
5. **Assignment Operations:** Simple assignment, compound assignment (e.g., `+=`, `-=`, `=`, `/=`).
6. **Increment and Decrement Operations**: Increment (`++`), decrement (`--`).
7. **Conditional (Ternary) Operation**: ‘condition? expression\_if\_true : expression\_if\_false`.
8. **Control Flow Operations:** ‘if`, `else`, `switch`, `while`, `do-while`, `for`.
9. **Function Call and Return Operations:** Defining functions, calling functions, returning values from functions.