|  |
| --- |
| **Functions:** |
| 1. Describe the purpose and structure of a function prototype in C programming. Why is it necessary to declare function prototypes? |
| A: A function prototype is a declaration of a function that specifies its name, return type, and parameters (if any), but does not provide the function's implementation. It essentially informs the compiler about the existence and signature of the function before its actual definition.  Purpose of Function Prototypes:  1.Function prototypes allow functions to be used before they are defined in the code. This enables early compilation of source files because the compiler knows the names, return types, and parameter types of functions that are used in the program.  2.Function prototypes help ensure type consistency between function calls and definitions. If a function is called with incorrect arguments or return types, the compiler can issue warnings or errors during compilation.  3.Function prototypes facilitate modular programming by allowing functions to be declared in header files and then used in multiple source files. This promotes code reusability and helps in organizing code into manageable units.  The structure of a function:  return\_type function\_name(parameter1\_type parameter1, parameter2\_type parameter2, ...)  Necessisity to declare a function:  1.Without function prototypes, the compiler would encounter function calls before it sees the function definitions. This would result in compilation errors or warnings because the compiler wouldn't know the return types and parameter types of the functions being called.  2.Function prototypes help catch type mismatches between function calls and definitions. If a function is called with incorrect arguments or return types, the compiler can issue warnings or errors during compilation.  3.Function prototypes allow functions to be declared in header files and then used in multiple source files. This promotes modularity, code reuse, and better organization of code.2 |
| 2. Explain the difference between call by value and call by reference in C programming. Provide examples to illustrate both concepts. |
| A: Call by Value:  In call by value, a copy of the actual argument's value is passed to the function parameter.  Any modifications made to the parameter within the function do not affect the original argument.  Primitive data types, such as int, float, char, etc., are passed by value.  Call by value is used when the function does not need to modify the original argument or when modifications should not affect the original argument.  Example for call by value:  #include <stdio.h>  void increment(int x);  int main()  {  int num = 5;  printf("Before increment: %d\n", num);  increment(num);  printf("After increment: %d\n", num);  return 0;  }  void increment(int x) {  x++;  }  Call by Reference:  In call by reference, the address of the actual argument is passed to the function parameter.  The function parameter becomes an alias for the original argument, and modifications made to the parameter within the function directly affect the original argument.  Pointers or references to variables are used for call by reference.  Call by reference is used when the function needs to modify the original argument.  Example for call by reference:  #include <stdio.h>  void increment(int \*x);  int main()  {  int num = 5;  printf("Before increment: %d\n", num);  increment(&num);  printf("After increment: %d\n", num);  return 0;  }  void increment(int \*x)  {  (\*x)++;  } |
| 3. Discuss the concept of recursion in C programming. Provide an example of a recursive function and explain how it works. |
| A: Recursion is a programming technique in which a function calls itself directly or indirectly to solve a problem. It's a powerful concept used in many algorithms and data structures. Recursion allows a problem to be solved by breaking it down into smaller, simpler subproblems.  This is the condition under which the recursion stops. It defines the smallest problem that can be solved directly without further recursion. Without a base case, the recursion would continue indefinitely, leading to a stack overflow.  This is the part of the function where it calls itself with modified arguments to solve a smaller instance of the original problem. The recursive case eventually leads to the base case.  The recursion must eventually reach the base case. Otherwise, it will result in infinite recursion and cause a stack overflow.  #include <stdio.h>  long long int factorial(int n)  {  if (n == 0)  {  return 1;  }  else  {  return n \* factorial(n - 1);  }  }  int main()  {  int n = 5;  unsigned long long result = factorial(n);  printf("Factorial of %d is %lld\n", n, result);  return 0;  } |
| 4. What is the significance of the return statement in C programming? How are values returned from functions? |
| A: The return statement is significant because it allows a function to send a value back to the caller. It's used to terminate the execution of a function and return control to the calling function or the operating system. The return statement can also be used without a value in a void function to simply exit the function.  The syntax of the return statement is:  return expression;  Values are returned from functions by assigning the value to be returned to the return statement. When the function is called, the expression following the return keyword is evaluated, and the resulting value is sent back to the caller. |
| 5. Describe the role of function parameters and arguments in C programming. How are function arguments passed to parameters? |
| A: Function parameters and arguments play a crucial role in passing data to and from functions.Parameters are variables declared in the function definition. They act as placeholders for values that will be passed into the function when it is called. Parameters specify the type and order of values that the function expects to receive.  Function Arguments: Arguments are the actual values passed to a function when it is called. These values are substituted for the parameters in the function's definition. Arguments must match the type and order of the parameters in the function's prototype or definition.  Passing Arguments to Parameters: In C programming, function arguments are passed to parameters either by value or by reference.  Pass by Value: When arguments are passed by value, a copy of the value is made and passed to the function. Changes made to the parameter inside the function do not affect the original argument.  Pass by Reference: When arguments are passed by reference, the address (memory location) of the argument is passed to the function. This allows the function to directly modify the original data. |