Victoria University of Bangladesh

Final Assessment-Fall 2023

Name : MD Sujan ali

ID : 1119460091

Program : BBA

Batch : 46th

COURSE CODE : CSE-211

COURSE TITLE : Structure Programming Language

Submitted to : Md shahin khan shanto

Course title: Structured Programming Language
Course code: CSE - 211

Ans: to the Q.N (1)-(a)

(a) Function:

A function is a mathematical concept that describes a relationship between input corresponds to exactly One output. In Programming, a function is a reusable block of code that performs a specific task, helping organize and modularize code.

P.t.o

Give the Syntax of a function

In programming, a function is a reusable block of code that performs a specific task. In C, the syntax for function declaration is

```
return_type function_name(parameter1_type parameter1_name, parameter2_type parameter2_name, ...) {
    // function body
    // Statements
    return return_value; // optional return statement
}
```

* 'return_type': the data type of the value that the function returns.

* 'function_name': the name of the function.

* 'parameter1_type', 'parameter2_type', etc.: the data types of the function parameters.

* 'parameter1_name', 'parameter2_name', etc.: the names of the function parameters.

* 'return_value': the value that the function returns (if the return type is not 'void').

p.t.o

For Example:

```
int add(int a, int b) {
    return a+b;
}
```

in this Example, the function 'add' takes two integers as parameters ('a' and 'b'), performs addition, and returns the result as an integer.

(b)-(1)

C-program to calculate average for the given three numbers using function:

write a c program to calculate the average of 3 numbers using function. the program takes three numbers from the user as input, calculates their average and prints the result on the output window.

P.T.O

Sample Example:

Enter three numbers: 10 20 30
Average of 10, 20 and 30 is: 20.000000
Enter three numbers: 5 2 7
Average of 5, 2 and 7 is: 4.666667

To calculate the average of three numbers using a user-defined function, we have to create a function that takes the three numbers as argument calculate their average and returns the result.

p.t.o

7

As the average of three numbers can be an integer or a float value therefore, we will keep the return type of the function a float which will take care of both integer and decimal point numbers. A float value is nothing but a decimal point value.

Implementation in the following program

```
// c program to find the average
// of three numbers using a
// function
```

p.t.o

8

```
# include <stdio.h>
// function declaration
float average(int, int, int);

int main() {
    int num1, num2, num3;
    float avg;
    Printf("Enter three numbers:");
    Scanf("%d %d %d", &num1, &num2,
          & num3);
    // Get average using average()
    function
    avg = average(num1, num2, num3);
    Printf("Average of %d, %d and
           %d is: %f", num1, num2, num3,
           avg);
    return 0;
```

```
// Function to calculate average
   of three numers
float average(int x, int y, int z){
        float avg;
        avg = (x+y+z)/3.0;
        return avg;
}
```

Output

Enter three numbers: 20 30 40

Average of 20, 30 and 40 is: 30.000000

Enter three numbers: 10 11 11

Average of 10, 11 and 11 is: 10.666667

p.t.o

Program Explanation

* The Program begins with the declaration of a function called average(), which takes three integer parameters (x, y, and z) and returns a floating-point value (avg).
* This function is declared before the main() function to inform the compiler about its existence.
* Variables num1, num2, and num3 are used to store user input numbers and a floating

p.t.o

-point variable avg to store the calculated average.

* to calculate the average of the three input numbers, the average() function is called inside the main() function with the values of num1, num2 and num3 as arguments.

* The result is stored in the avg variable.

* Finally, the Program uses printf() to display the calculated average along with the input values.

c)-(1)

The advantages of using function

function in programming offer several advantages:

① Modularity: functions allow breaking down a program into smaller, manageable pieces, making the code more organized and easier to understand.

② Reusability: Once defined, functions can be reused in different parts of a program promoting code reuse and reducing redundancy.

③ Abstraction: Functions abstract away the implementation details, providing a high-level interface. Users only need to know what a function does, not how it achieves it.

④ Readability: Breaking code into functions enhances readability by giving meaningful names to blocks of code, improving the overall clarity of the program.

P.t.o

(5) Debugging: Functions simplify debugging since issue can be isolated to specific function making it easier to identify and fix problems.

(6) Scoping: Function provide scope for variables, limiting their accessibility to specific parts of the code, reducing naming conflicts and improving code reliability

(7) testing: Function make it easier to test individual components of a program.

(8) Encapsulation: Function encapsulate logic, preventing unintended interference with other parts

Ans: to the q.no:- (2) - (a)

(2)-(a)

Define Array:

Arrays are "lists" of related values. Every value in the array is usually of the exact same type and only differentiated by the position in the array. For example, all the quiz scores for a test could be stored in an array with the single variable name: quiz_scores.

Another way is to define an Array

Arrays: p.t.o

An Array is a data structure that stores a collection of elements, each identified by an index or a key. It allows for efficient access and manipulation of the elements based on their position in the sequence.

Example:

```
number = [1, 2, 3, 4, 5]
```

Here "number" is an array containing integers, and each element can be accessed by its index (e.g., 'numbers[2]' would be 3).

(b) - (2)

Here's a simple C Program to find the average marks obtained by a class of 30 students in 5 structured programming tests:

```
#include <stdio.h>
Struct student {
    char name[50]
    float marks[5]

};
int main() {
    struct student students[30];
    int numStudents = 30;
    int numtests = 5;
```

P.t.o

```
// Input marks for each student
for (int i=0; i<numStudents; ++i) {
    printf("Enter name of student %d:
        ", i+1);
    scanf("%s", students[i].name);
    printf("Enter marks for student
        %d in 5 tests:\n", i+1);
    for (int j< numtests; ++j) {
        printf("test %d: ", j+1);
        scanf("%f", &students[i].
            marks[j]);
    }
}
// calculate average marks for
   each student
for (int i=0; i<numStudents; ++i){
```

P.t.o

```
    float totalMarks = 0;
    for (int j = 0; j < numTests; ++j) {
        totalMarks += Students[i].
        marks[j];

    }
    float averageMarks = totalMarks /
        numTests;
    Printf("Average marks for %s: %.
        2f\n", Students[i].name,
        averageMarks);

}
return 0;

}
```

This program a 'Students' structure to store the name and marks of each students. It then takes input for 30 students and calculate and prints their average marks for the 5 tests.

(c)-(2)

Arrays offer advantages such as efficient storage, quick access to elements through indexing, and simplicity in managing collections of data. They provide a structured way to way organize and manipulate data, making algorithms more straightforward improving overall program performance.

Arrays have advantages like:

(1) Efficient Random Access:

elements in an array can ~~be~~

P.t.o

be accessed directly through their index, allowing for fast and constant-time retrieval.

② Memory Efficiency: Arrays allocate a contiguous block of memory, minimizing overhead ensuring efficient use of memory.

③ Simplicity: Arrays provide a straightforward and easy-to-understand way of organizing and managing collections of data.

④ Fixed Size: In some cases a fixed size structure is desirable.

⑤ Cache Locality: contiguous memory storage enhances cache locality, improving performance by reducing cache

Ans: to the Q.N (3)-(a)

(a)-(3)

A C-Program to calculate GCD bon two Positive integer:

In in this section, we will discuss the GCD of two number in C. The Highest common Multiple & the Greatest common Divisor is the greatest number that exactly divides both numbers

Example:-

The G.C.D of 24 and 40 is 8.

The G.C.D of and 12 is 6

p.t.o

various Methods to calculate the GCD

* Using prime factorization,
* Euclid's Algorithm,
* Lehmer's GCD algorithm

Method 1

For a user input n1 & n2.

* Use a for loop linearly traversing such that (i<=num1 || i<=num2)
* Find max i value such that both n1 and n2 are divisible by i

(num1 % i == 0 && num2 % i == 0)

P.t.o

c code (Method 1)

```
#include<stdio.h>
int main()
{ int n1=18, n2=45, gcD=1;
  for(int i=1; i<=n1 || i<=n2; i++) {
      if(n1%i==0 && n2%i==0)
          gcd=i;
  }
  printf("the GCD of %d and %d is:
          %d", n1, n2, gcd);
  return 0;
}
// Time complexity : O(N)
// Space complexity : O(1)
```

ouput

the GCD of 18 and 45 is: 9

p.t.o

Method 2
Euclidean Algorithm is used for this method
* Euclidean Algorithm: GCD of two numbers remains constant even if we subtract the smaller number from the higher number.
is an example of a manual calculation that you can understand before moving ahead with the code.

Example,

Let m = 104, n = 24

m > n

m = m - n = 104 - 24 = 80

Now m = 80, n = 24

m = m - n = 56 - 24 = 32

now m = 32, n = 24

P.t.o

m = m - n = 32 - 24 = 8

now = 8, n = 24

Since, m < n

Now we will need to change subtraction order.

n = n - m = 24 - 8 = 16

now m = 8, n = 16

n = n - m = 16 - 8 = 8

now m = 8, n = 8

m = n. So, stop. GCD = 8

code (Method 2)

```
#include<stdio.h>
//Using Euclidean Algorithm (Repeated subtraction)
int main()
{
  int n1 = 109, n2 = 24, gcd = 1;
  while (n1 != n2)
  {
```

P.t.o

```
if (n1 > n2)
    n1 -= n2;
else n2 -= n1;
}
printf("the GCD: %d", n1, n2, gcd);
return 0;
}
```

output

the GCD: 8

* Method 3

Euclidean Algorithm is again used for this method, this method has two changes.

* Uses recursive approach
* Also, handles if one of the numbers is 0

P.t.o

code (Method 2)

```
#include<stdio.h>
// Using Recursive Euclidean Algorithm
  (Repeated subtraction) int calcred
   (int n1, int n2)
{
 // taken care of the case n1 is 0
 // we have given explanation above
 if (n1==0)
     return n2;
// taken care of the case n2 is 0
// we have given explanation above
  if (n2==0)
     return n1;
  // base case
    if (n1==n2)
      return n1;
```

P.t.o

```
// n1 is greater
    if (n1 > n2)
        return calcGCD(n1-n2, n2);
    // n2 is greater
    return calcGCD(n1, n2-n1);
}
int main()
{
    int n1=45, n2=18;
    int gcd = calcGCD(n1, n2);
    printf("the GCD of %d and %d
            is: %d", n1, n2, gcd);
    return 0;
}
```

output

The GCD of 45 and 18 is: 9

(b)-(3)

A c-Program to calculate sum of the series using goto statements:

Here,

A simple c program using goto statements to calculate the sum of the series $225+227+229+\cdots+N$:

```
#include<stdio.h>
int main(){
    int N, num=225, sum=0;
    Printf("Enter the value of N:");
    scanf("%d", &N);
```

P.t.o

```
// Using goto statement to iterate
through the Series
start:
sum+=num;
num+=2;
IF(num<=N)
    goto start;
PrintF("sum of the series: %d/n",
            sum);
,return 0;
```

3 This Program takes the value of N as input and uses a goto statement to iterate through the series, adding the odd number until it reaches N. Finally, it prints the sum of the series

Ans:- to the Q.N (5) - (a).

(a) - (5)

Define Recursion

Recursion is a Programming or math ematical concept where a function calls itself either directly or indirectly to solve a Problem by breaking it down into smaller instances of the same Problem.

more easy to say Recursion means "Defining a Problem in terms of itself." This can be a very powerful tool in writing

P.t.o

algorithms. Recursion comes directly from Mathematics, where there are many examples or expressions written in terms of themselves. For example, the Fibonacci Sequence is defined as: F(i) = F(i-1) + F(i-2)

Give an appropriate Syntax:

Syntax in English sets forth a specific order for grammatical elements like subjects, verb, direct and indirect objects etc. ~~Recursion~~ Recursion is a programming concept where a function calls itself during its execution

p.t.o

Syntax,

```
def recursive_function(parameter):

#base case
if base_condition:
    return base_value

#recursive case
else:
    return recursive_function(modi-
    fied_parameter)
```

Advantage of Recursion:

Recursion allows for the implementation of algorithms in a more concise and expressive manner, particularly when solving problems with

p.t.o

inherent recursive structures. it can lead to code that is easier to understand and maintain.

more 5 Advantage of Recursion

Simplicity: Recursive solutions often express the problem in a natural and simple way.

Readability: Recursive code can be more readable than iterative code can be more for certain problem

Divide and conquer: Recursive algorithms can naturally implement the divide-and-conquer strategy.

Modularity: Recursive functions can promote modularity by breaking down

(b) - (5)

Here's

A c Program to calculate the sum of the series 1+2+3+---+n using recursion

The sum of the first n natural numbers have been shown here. Both the iterative and recursive approaches have been shown below. the sum can also be computed using the mathematical formula $\frac{n(n+1)}{2}$.

Example:

Suppose = 6

So the Sum of first 6 natural

p.t.o

numbers = 1+2+3+4+5+6 = 21

by using the mathematical formula we obtain the sum of first 6 natural numbers = $\frac{6 \cdot (6+1)}{2} = 21$

The algorithm, pseudocode and time complexity of the program have also been covered in the following sections.

1. Algorithm
2. Pseudocode
3. time complexity
4. program and output:

* Using formula
* Iterative approach
* Recursive approach

p.t.o

C program to calculate the sum of the series 1+2+3+...+n using recursion.

```
#include<stdio.h>

//Function to calculate the sum recursively int calculatesum(int n){

    if(n==0){
        return 0; //Base case sum of first 0 natural numbers is 0

    }else{
        return n+calculatesum(n-1); //Recursive case: sum of first n natural numbers is n+ sum of first (n-1) natural numbers

    }
```

p.t.o

```
}
// Main function to test the program
int main() {
    int n;
    printf("Enter the value of n: ");
    scanf("%d", &n);

    // Displaying the sum
    printf("Sum of the series 1+2+3+...+%d
            is: %d\n", n, calculateSum(n));
    return 0;
}
```

This program defines a recursive function sumSeries that calculate the sum of the series. the 'main' function takes input for the value of 'n' and then calls the sumSeries function to compute the result.