PROGRAMMING FOR PROBLEM SOLVING

UNIT 3 ONE SHOT

WELCOME

INTELLIGENCE LEARNING

UNIT 3 BCS101 / BCS201: PROGRAMMING FOR PROBLEM SOLVING

Ques. What is Looping Control Instuctions ? Describe the type of looping statement in C with necessary syntax?

Ans. Looping in C means repeating a block of code again and again until a condition is met.
Types of Loops in C

1. **for loop**
2. **while loop**
3. **do...while loop**

1. for loop :- It is used to execute a set of statements repeatedly until a particular condition is satisfies.

SYNTAX:-

```
for (start; condition; step) {
   // code to repeat
}
```

Example:-

```
for (int i = 1; i <= 5; i++) {
   printf("Hello\n");
}
```

2. While Loop :- It is entry control loop. Here condition is checked first and if condition is true. Only then the block will be execute ‘n’ times according to condition. We do not put semicolon after while condition statement.

SYNTAX:-

```
while (condition) {
   // code to repeat
}
```

Example:-

```
int i = 1;
while (i <= 5) {
   printf("Hello\n");
   i++;
}
```

3. do...while loop :- It is exit control loop. In do while loop , block is executed first and then condition is checked so block will execute at least 1 times even condition is false. And the condition is true for ‘n’ time the block will execute (n+1) times. We put semicolon after while condition statement.

SYNTAX:-

```
do {
   // code to repeat
} while (condition);
```

Example:-

```
int i = 1;
do {
   printf("Hello\n");
   i++;
} while (i <= 5);
```

Ques. Give the loop statement to print the following sequence of integers: ⇒ -6, -4, -2, 0, 2, 4, 6

Ans.

```
#include <stdio.h>
#include <conio.h>

void main() {
    int i;
    for (i = -6; i <= 6; i++) {
        if (i % 2 == 0) {
            printf("%d ", i);
        }
    }
    getch();
}
```

C program to check whether a number is a prime number or not.

```
#include <stdio.h>
#include <conio.h>

void main() {
    int n, i, f = 0;

    printf("Enter Number: ");
    scanf("%d", &n);

    for (i = 1; i <= n; i++) {
        if (n % i == 0) {
            f++;
        }
    }
```

```
if (f == 2)
      printf("Prime number\n");
   else
      printf("Not Prime\n");

   getch();
}
```

C program to generate the Fibonacci series

```
#include <stdio.h>
#include <conio.h>
void main()
{
   int n, a = 0, b = 1, c, i;
   printf("Enter number of terms: ");
   scanf("%d", &n);
   printf("%d %d ", a, b);  // print first two terms
   for (i = 2; i < n; i++)
   {
      c = a + b;
      printf("%d ", c);
      a = b;
      b = c;
   }
   getch();
}
```

OUTPUT:-

```
Enter number of terms: 5
0 1 1 2 3
```

Program to Calculate Sum of Fibonacci Series

```
#include <stdio.h>
#include <conio.h>
void main()
{
   int a = 1, b = 1, n, c, i, s = 0;
   printf("Enter number of Terms: ");
   scanf("%d", &n);
   for (i = 1; i <= n; i++)
   {
      c = a + b;
      s = s + c;
      a = b;
      b = c;
   }
   printf("Sum of series = %d", s);
   getch();
}
```

Program to Find Sum of Digits

```
#include <stdio.h>
#include <conio.h>
void main()
{
   int m, d, s = 0;
   printf("Enter Number: ");
   scanf("%d", &m);
   while (m > 0)
   {
      d = m % 10;
      s = s + d;
      m = m / 10;
   }
   printf("Sum of digits = %d", s);
   getch();
}
```

OUTPUT:-

Enter Number: 153

Sum of 153 = 9

Program to Check Whether a Number is Armstrong

```
#include <stdio.h>
#include <conio.h>

void main()
{
   int m, n, d, s = 0;
   printf("Enter Number: ");
   scanf("%d", &m);

   n = m;

   while (m > 0)
   {
      d = m % 10;
      s = s + (d * d * d);
      m = m / 10;
   }
```

```
    if (n == s)
        printf("It is an Armstrong Number");
    else
        printf("It is Not an Armstrong Number");

    getch();
}
```

Explanation:

Armstrong number: Sum of **cube of digits** = original number.

Example: $1^3 + 5^3 + 3^3 = 153 \rightarrow$ **Armstrong**

OUPTUT:-

Enter Number: 153

Armstrong

Program to Reverse a Number

```
#include <stdio.h>
#include <conio.h>
void main()
{
   int m, n, d, r = 0;
   printf("Enter Number: ");
   scanf("%d", &m);
   n = m;
   while (m > 0)
   {
      d = m % 10;
      r = r * 10 + d;
      m = m / 10;
   }
   printf("Reverse of %d = %d", n, r);
   getch();
}
```

OUTPUT:-

Enter Number: 121
Reverse of 121 = 121

Program to Check Whether a Number is a Palindrome

```
#include <stdio.h>
#include <conio.h>
void main()
{
   int m, n, d, r = 0;
   printf("Enter Number: ");
   scanf("%d", &m);
   n = m;
   while (m > 0)
   {
      d = m % 10;
      r = r * 10 + d;
      m = m / 10;
   }
```

```
    if (n == r)
        printf("Palindrome");
    else
        printf("Not Palindrome");

    getch();
}
```

OUTPUT:-

Enter Number: 121
Palindrome

Explanation:-

A **palindrome number** is a number that stays the **same** when its digits are **reversed**.
For example:
121 reversed is 121 → same → **palindrome**
1331 reversed is 1331 → same → **palindrome**
123 reversed is 321 → not same → **not a palindrome**

Original number = 121
Reversed number = 121
They are equal → It's a palindrome □

Program to Add the Given Series

Series: 1 + (1+2) + (1+2+3) + ... + (1+2+3+...+n)

```
#include <stdio.h>
#include <conio.h>
void main()
{
   int i, j, n, s = 0;
   printf("Enter value of n: ");
   scanf("%d", &n);
   for (i = 1; i <= n; i++)
   {
      for (j = 1; j <= i; j++)
      {
         s = s + j;
      }
   }
   printf("Sum of series = %d", s);
   getch();
}
```

OUTPUT:-

Enter value of n: 3
Sum of series = 9

Explanation:

Nested loop calculates the sum of the series:
1 + (1+2) + (1+2+3) = 1 + 3 + 5 = 9

Write a C program to print the following pattern:

```
*
* *
* * *
```

```
#include <stdio.h>
#include <conio.h>
void main() {
   int n, i, j;
   clrscr();
   printf("Enter Number of Rows:\n");
   scanf("%d", &n); // take input from user
   for (i = 1; i <= n; i++)
    {
      for (j = 1; j <= i; j++)
       {
         printf("* ");
       }
      printf("\n"); // go to next line after each row
    }
   getch();
}
```

Pattern

A

A B

A B C

A B C D

A B C D E

A B C D E F

```
#include <stdio.h>
#include <conio.h>
void main() {
    int i, j;
    for(i = 65; i <= 70; i++)
    {
        for(j = 65; j <= i; j++)
        {
            printf("%c ", j);
        }
        printf("\n");
    }
    getch();
}
```

BREAK	CONTINUE
1. It is used to come out of the Loop.	1. It is used to come in the beginning of Loop.
2. Break will skip all the coming iteration.	2. While Continue will skip only current iteration.
3. It is used to come out of switch statement.	3. It is not used in switch statement.

BREAK

```
#include <stdio.h>
int main() {
   for(int i = 1; i <= 5; i++) {
      if(i == 3) {
         break;  // stop the loop when i is 3
      }
      printf("%d\n", i);
   }
   return 0;
}
```

OUTPUT:-
1
2

CONTINUE

```
#include <stdio.h>
int main() {
   for(int i = 1; i <= 5; i++) {
      if(i == 3) {
         continue;  // skip this round when i is 3
      }
      printf("%d\n", i);
   }
   return 0;
}
```

OUTPUT:-
1
2
4
5

What are the Types of Functions in C?

1. Library Functions

I. Already written and provided by C language.

II. We just need to **use** them.

2. User-defined Functions

- Created by the **programmer**.
- Used to perform a **specific task** again and again.
- Helps make programs clean and reusable.

Q: What is the meaning of prototype of a function? (2016-17). Explain function declaration and definition of a function with example? (2015-16, 2018-19)

Ans:

A **function prototype** is simply the **declaration** of a function that specifies function name, parameters, and return type.

Declaration of Function:

Syntax:

Return_type function_name (parameter type);

Examples:

int fun(); int sum(int, float); void fun();

Calling of Function:

Syntax:

Variable = function_name (parameter value); // Variable is optional

Examples:

y = fun(20, 3); fun();

Definition of Function:

Syntax:

```
Return_type function_name (parameter declaration)
{ // Body of function
 return (value); // return is optional }
```

Q: Define Actual & Formal Argument / Parameters (2014–15, 2015–16, 2017–18)

ANS.

Actual Argument:

The variables which are used during function call by the calling function are known as **actual parameters** or **arguments**.

Formal Argument:

The variables which are used in function definition to collect the values of actual argument are known as **formal parameters** or **arguments**.

Q. Difference between Call by Value and Call by Reference (or Address)?

Call by Value	Call by Reference
Value of actual argument is passed to the function.	Address of actual argument is passed to function.
Pointer is not used.	Pointer variable is used.
If value of formal argument is changed, there is **no change** in the value of actual argument.	If value of formal argument is changed, there **is a change** in the value of actual argument.

CALL BY VALUE EXAMPLE

```
#include <stdio.h>
#include <conio.h>
void swap(int a, int b);
void main ()
{
int a,b;
printf (“\n Enter Two Number:\n”);
scanf(“%d %d”, &a, &b);
swap (a,b);
getch();
void swap (int a, int b)
{
printf (“Before Swapping \n”);
printf (“a=%d b=%d”, a, b);
a=a+b;
b=a-b;
a=a-b;
printf (“After Swapping \n”)
printf (“a=%d b=%d”, a, b);
}
```

CALL BY REFERENCE EXAMPLE

```
#include <stdio.h>
#include <conio.h>
void swap(int *a, int* b);
void main ()
{
int a,b;
printf (“\n Enter Two Number:\n”);
scanf(“%d %d”, &a, &b);
swap (a,b);
getch();
void swap (int *a, int* b)
{
printf (“Before Swapping \n”);
printf (“a=%d b=%d”, *a, *b);
*a=*a+*b;
*b=*a-*b;
*a=*a-*b;
printf (“After Swapping \n”)
printf (“a=%d b=%d”, *a, *b);
}
```

Q. What are the main principles of recursion? Explain in detail? (2014-15, 2015-16)

Ans: A function is calling itself again and again to solve a particular problem is called recursion.

Principle of Recursion:

- A recursive program must call itself recursively.
- A recursive program must have a base condition.
- A recursive program must change its state & move toward the base condition.

Note:

In recursion, function is calling itself again & again so we need a condition to stop the infinite recursive calls, this condition is known as **Base Condition**.

WAP to find factorial of a given number without using recursion

```
#include <stdio.h>

int main() {
    int num, i, fact = 1;

    printf("Enter a number: ");
    scanf("%d", &num);

    for (i = 1; i <= num; i++) {
        fact = fact * i;
    }

    printf("Factorial = %d\n", fact);

    return 0;
}
```

Output:-

```
Enter a number: 5
Factorial = 120
```

WAP to find factorial of a given number using recursion.

```
#include <stdio.h>
#include <conio.h>
int fact(int m);
void main() {
  int m, f;
  clrscr();
  printf("Enter no:");
  scanf("%d", &m);
  f = fact(m);
  printf("Factorial = %d", f);
  getch();
}
int fact(int m) {
  if (m == 1)
    return 1;
  else
    return (m * fact(m - 1));
}
```

Output :-

Enter no = 5

Factorial = 120

Q: Write a C program to generate Fibonacci series using recursion

```
#include <stdio.h>
#include <conio.h>

int fib(int y);

void main() {
   int n, i, x;
   clrscr();
   printf("Enter no of Terms: ");
   scanf("%d", &n);

   for (i = 1; i <= n; i=i+1) {
     x = fib(i)
       printf("%d\t", x);
   }

   getch();
}
```

```
int fib(int y) {
    if (y == 1)
        return 0;
    else if (y == 2)
        return 1;
    else
        return (fib(y - 1) + fib(y - 2));
}
```

Output:-

```
Enter no of Terms
5
0  1  1  2  3
```

Q: What is storage class? Describe automatic, register, static and external with neat syntax. *(2014-15)*
Q: What do you mean by scope & life time of a variable *(2015-16, 2018-19)*
Q: Differentiate b/w static & register class in C language *(2014-15, 2016-17)*

Ans: Storage class in C decides the part of storage to allocate memory for a variable.
Declaration of Variable:
Syntax:
storage_class datatype variable_name;

Storage Class/Key word	Storage	Default Initial Value	Scope	Life	Declare Example
Automatic	memory	garbage	local	The block where declared	auto int a;
Register	register	garbage	local	The block where declaredwith in the built-in reg.	register int a;
Static	memory	zero	local	Value persists b/w different calls	static int a;
External	memory	zero	global	Throughout the program	extern int a;

THANK YOU FOR WATCHING

DO LIKE SHARE COMMENT AND SUBSCRIBE