24-Dec-2024 Assignment

Q1 Basic Global and Local Variable Usage

Problem Statement: Write a program that declares a global variable and a local variable with the same name. Modify and print both variables to demonstrate their scope and accessibility.

```
#include<stdio.h>

void function(void);

int a=90;

int main()

{

   printf("Value of a= %d\n",a);

   function();

   return 0;

}

void function()

{

   int a=20;

   printf("a=%d",a);

}
```

Q2 Global Variable Across Functions

Problem Statement: Declare a global variable and create multiple functions to modify its value. Each function should perform a different operation (e.g., addition, subtraction) on the global variable and print its updated value.

```
#include<stdio.h>

int a=12;

void f_1(void);

void f_2(void);

void f_3(void);

int main()

{
    f_1();
    f_2();
    f_3();

    return 0;
}
void f_1(void)
{
    a+=3;
    printf("After addition,a=%d\n",a);
}
void f_2(void)
{
    a=a-3;
    printf("After subtraction, a=%d\n",a);
}
void f_3(void)
{
    a*=3;
     printf("After multiplication, a=%d\n",a);
}
```

Q3 **3. Local Variable Initialization**

Problem Statement: Write a program with a function that declares a local variable and initializes it to a specific value. Call the function multiple times and observe how the local variable behaves with each call.

```
#include <stdio.h>
void f_1(void);
int main()
{
    f_1();
    f_1();
    f_1();
    return 0;
}
void f_1(void)
{
    int a = 10;
    a+=5;
    printf("The value of a is %d\n", a);
}
```

Q4 **4. Combining Global and Local Variables**

Problem Statement: Write a program that calculates the sum of a global variable and a local variable inside a function. Print the result and explain the variable scope in comments.

```
#include <stdio.h>

int g = 20;//Global variable

void sum(void);

int main()

{

    sum();

    return 0;

}

void sum(void)

{

    int l = 10; // Local variable

    printf("%d+%d= %d\n", g, l, g+l);
}
```

Q5 **5. Global Variable for Shared State**

Problem Statement: Write a program that uses a global variable as a counter. Multiple functions should increment the counter and print its value. Demonstrate how global variables retain their state across function calls.

```
#include<stdio.h>

int counter=0;

void f_1(void);

void f_2(void);

int main()

{

   f_1();

   f_2();

   return 0;

}

void f_1(void)

{

   counter++;

   printf("Counter=%d\n",counter);

}

void f_2(void)

{

   counter++;

   printf("Counter=%d\n",counter);

}
```

Q6 Shadowing Global Variables

Problem Statement: Write a program where a local variable in a function shadows a global variable with the same name. Use the global scope operator to access the global variable and print both values.

```
#include <stdio.h>

int value = 100;

void shadow(void);

int main()

{

   int value = 50; //Local variable shadows the global variable. Here local one is
having high priority

   printf("Local value = %d\n", value);

   return 0;

}//output: value=50
```

Q7 Read-Only Global Variable

Problem Statement: Declare a global constant variable and write a program that uses it across multiple functions without modifying its value. Demonstrate the immutability of the global constant.

```
#include<stdio.h>

int const var=15;

void f_1(void);

void f_2(void);

int main()

{

   f_1();

   f_2();

   return 0;

}
```

```
void f_1(void)

{

   int sum=var+5;

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

}

void f_2(void)

{

   int mul=var*5;

   printf("Sum=%d\n",mul);

   //var+=2;

   //printf("Global variable=%d\n",var);

}
```

Q8 **Global Variable for Configuration**

Problem Statement: Use a global variable to store configuration settings (e.g., int configValue = 100). Write multiple functions that use this global configuration variable to perform operations.

```
#include<stdio.h>

int configValue=100;

void config_1(void);

void config_2(void);

void config_3(void);

int main()

{

   config_1();

   config_2();

   config_3();
```

```
    return 0;

}

void config_1()

{

    configValue+=1;

    printf("config_1=%d\n",configValue);

}

void config_2()

{

    configValue+=2;

    printf("config_2=%d\n",configValue);

}

void config_3()

{

    configValue+=3;

    printf("config_3=%d\n",configValue);

}
```

Q9 **Local Variables with Limited Scope**

Problem Statement: Write a program where local variables are declared inside a block (e.g., if or for block). Demonstrate that they are inaccessible outside the block.

```
#include <stdio.h>

int main() {

  for (int i = 0; i < 5; i++) {

        int mul = i * 2;

        printf("%d*2 = %d\n",i, mul);

    }
    //printf("%d*2 = %d\n",i, localVar);

     return 0;

  }
```

Q10 **Combining Local and Global Variables in Loops**

Problem Statement: Write a program that uses a global variable to track the total sum and a local variable to store the sum of elements in an array. Use a loop to calculate the local sum, then add it to the global total.

```
#include<stdio.h>

float total_sum=1;

int main()

{

   int a[4],sum_elements;

   printf("Enter array elements\n");

   for(int i=0;i<4;i++)

   {

      scanf("%d",&a[i]);



   }

   for(int i=0;i<4;i++)

   {

      sum_elements+=a[i];

   }

   printf("Sum of elements=%d\n",sum_elements);

   total_sum+=sum_elements;

   printf("Total=%f",total_sum);
```

```
    return 0;

}
```

Q1. Static Variable in a Loop

Problem Statement: Write a program that uses a static variable inside a loop to keep track of the cumulative sum of numbers from 1 to 10. The loop should run multiple times, and the variable should retain its value between iterations.

```
#include <stdio.h>

int main() {

      static int sum = 0;

    for (int i = 1; i <= 10; i++) {

        sum += i;

        printf("%d: Cumulative sum = %d\n", i, sum);

    }

     return 0;

}
```

Q2 **Static Variable to Count Iterations**

Problem Statement: Use a static variable inside a loop to count the total number of iterations executed across multiple runs of the loop. Print the count after each run.

```
#include <stdio.h>

void loop_1(void);

int main() {

    loop_1();

    loop_1();

    loop_1();
    return 0;

}



void loop_1() {

    static int count = 0;

    for (int i = 1; i <= 10; i++) {

        count++;

    }

    printf("Total iterations: %d\n",count);

}
```

Q3 **Static Variable in Nested Loops**

Problem Statement: Use a static variable in a nested loop structure to count the total number of times the inner loop has executed across multiple runs of the program.

```
#include <stdio.h>

void loop_1(void);

int main() {

    loop_1();

    loop_1();

    loop_1();

    return 0;

}

void loop_1() {

    static int inner_loop_count = 0;

    for (int i = 1; i <= 10; i++) {

        for(int j=1;j<=4;j++){

            inner_loop_count++;
        }
    }
    printf("Total iterations of inner loop: %d\n",inner_loop_count);
}
```

Q4 **Static Variable to Track Loop Exit Condition**

Problem Statement: Write a program where a loop executes until a specific condition is met. Use a static variable to track and display the number of times the loop exited due to the condition being true.

```
#include <stdio.h>

int main() {

      static int sum = 0;

      static int count;

    for (int i = 1; i <= 10; i++) {

        sum += i;

        printf("%d: Cumulative sum = %d\n", i, sum);

        if(sum>=25)

        {

           count++;

           break;

        }

        count++;

    }

    printf("Total count = %d\n",count);

     return 0;

}
```

Q5 **Static Variable to Track Loop Re-entry**

Problem Statement: Write a program where a static variable keeps track of how many times the loop is re-entered after being interrupted (e.g., using a break statement).

```
#include <stdio.h>

int main() {

    static int re_entry_count = 0;

    for (int i = 1; i <= 10; i++) {

        printf("Iteration %d\n", i);

        if (i == 5) {

            printf("Loop interrupted at iteration %d\n", i);

            break;

        }

    }

    re_entry_count++;

    printf("Re-entry count: %d\n", re_entry_count);



    for (int i = 6; i <= 10; i++) {

        printf(“Iteration %d\n", i);

    }

    return 0;

}
```

Q6 **Static Variable for Step Count in Loops**

Problem Statement: Create a program with a loop that increments by a variable step size. Use a static variable to count and retain the total number of steps taken across multiple runs of the loop.

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

```
void loop_1(int step_size);

int main() {

    loop_1(2);

    loop_1(3);

    loop_1(4);

    return 0;

}


void loop_1(int step_size) {

    static int total_steps = 0;

    int current_steps = 0;

    for (int i = 1; i <= 20; i += step_size) {

        current_steps++;

    }

    total_steps += current_steps;


    printf("Steps in current run: %d, Total steps f: %d\n", current_steps, total_steps);

}
```

Problem statement on const Type specifier

Q1 Using const for Read-Only Array

> **Problem Statement**: Declare an array of integers as const and use a loop to print each element of the array. Attempt to modify an element inside the loop and explain the result.

```
#include <stdio.h>

int main()
```

```
{

    int const a[5]={1,2,3,4,5};

    for(int i=0;i<5;i++)

    {

        printf("a[%d]=%d,",i,a[i]);

        //a[0]=8; Enabling this comment will cause error due to assigning value to read only
variable

    }

    return 0;

}
```

Q2 **const Variable as a Loop Limit**

Problem Statement: Declare a const integer variable as the upper limit of a loop. Write a loop that runs from 0 to the value of the const variable and prints the iteration count.

```
#include<stdio.h>

int main()

{

    int const a=5;

    int count=0;

    for(int i=0;i<=a;i++)

    {

    printf("%d,",i);

    count++;

    }

    printf("\nThe count value=%d",count);

    return 0;

}
```

Q3. Nested Loops with const Limits

Problem Statement: Use two const variables to define the limits of nested loops. Demonstrate how the values of the constants affect the total number of iterations.

```
#include<stdio.h>

int main()

{

   int const outer=5;

   int const inner=3;

   int count=0;

   for(int i=0;i<=outer;i++)

   {

      for(int j=0;j<=inner;j++)

      {

         printf("Outer loop:%d,Inner loop:%d\n",i,j);

         count++;


      }


   }

   printf("\nThe count value=%d",count);

   return 0;

}
```

Q4 const for Read-Only Pointer in Loops

Problem Statement: Declare a const pointer to an integer and use it in a loop to traverse an array. Print each value the pointer points to.

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

```
int main() {

    int arr[] = {10, 20, 30, 40, 50};

    int *const ptr = arr; //the array name arr decays into a pointer to the first element of
the array. This is a feature of C: when an array name is used in an expression, it is
treated as a pointer to its first element

    printf("Array values:\n");

    for (int i = 0; i < 5; i++) {

        printf("%d ", *(ptr + i));

    }

    printf("\n");



    // ptr = &arr[1]; // Uncommenting this line will cause an error because ptr is a const
pointer



    return 0;

}
```

Q5. const for Loop-Invariant Variable

Problem Statement: Declare a const variable that holds a mathematical constant (e.g., PI = 3.14). Use this constant in a loop to calculate and print the areas of circles for a range of radii.

```
#include <stdio.h>

int main() {

    const float PI = 3.14; // Declare a const variable for PI

    int r1,r2;

    printf("Enter the starting range of radii\n");

    scanf("%d",&r1);
    printf("Enter the ending range of radii\n");
   scanf("%d",&r2);
   for (int i = r1; i<=r2; i++) {
       float area = PI *i*i;
       printf("%d\t%.2f\n", i, area); // Print the radius and corresponding area
   }
   return 0;
}
```

Q6 const Variable in Conditional Loops

Problem Statement: Use a const variable as a termination condition for a while loop. The loop should terminate when the iteration count reaches the value of the const variable.

```
#include <stdio.h>

int main() {
   const int limit = 10;
   int count = 0;
   while (count < limit) {
       printf("Iteration %d\n", count + 1);
       count++;
   }
   printf("Loop terminated after %d iterations.\n", limit);
   return 0;
}
```

Q7 const and Immutable Loop Step Size

Problem Statement: Declare a const variable as the step size of a for loop. Use this step size to iterate through a range of numbers and print only every nth number.

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

```
int main() {
    const int step_size = 4;
    int start = 0;
    int end = 20;
    for (int i = start; i < end; i += step_size) {
        printf("%d\n", i);
    }
    return 0;
}
```

Q8 . const Variable for Nested Loop Patterns

Problem Statement: Use two const variables to define the number of rows and columns for printing a rectangular pattern using nested loops. The dimensions of the rectangle should be based on the const variables.

```
#include <stdio.h>

int main() {
    const int row = 6;
    const int column=4;


    for (int i = 1; i <=row; i++) {
        printf("\n");
        for(int j=1;j<=column;j++){
            printf("*");
        }
    }
    return 0;
}
```