ASSIGNMENT DAY 5 – 24/12

LOCAL AND GLOBAL VARIABLES USAGE:

```c
#include <stdio.h>
int Number = 3;

void Usage() {
   int Number = 20;
   printf("Local variable value inside function: %d\n", Number);
}
int main() {
   printf("Global variable value in main: %d\n", Number);
   Usage();
   printf("Global variable value after function call: %d\n", Number);
   return 0;
}
```

OUTPUT:

```
Global variable value in main: 3
Local variable value inside function: 20
Global variable value after function call: 3
```

COMBING LOCAL AND GLOBAL VARIABLES:

```c
#include <stdio.h>

int totalSum = 0;
```

```
void calculate_local_sum(int arr[], int size) {
    int localSum = 0;
    for (int i = 0; i < size; i++) {
        localSum += arr[i];
    }
    totalSum += localSum;
}

int main() {
    int arr[] = {1, 2, 3, 4, 5};
    int size = sizeof(arr) / sizeof(arr[0]);

    calculate_local_sum(arr, size);

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

    return 0;
}
```

OUTPUT:

```
Total Sum: 15
```

GLOBAL CONSTANT:

```
#include <stdio.h>

#define GLOBAL_CONSTANT 100
```

```
void display_constant() {
   printf("The value of GLOBAL_CONSTANT is: %d\n", GLOBAL_CONSTANT);
}

void use_constant_in_computation() {
   int result = GLOBAL_CONSTANT * 2;
   printf("Result of computation using GLOBAL_CONSTANT: %d\n", result);
}

int main() {
   display_constant();
   use_constant_in_computation();
   return 0;
}
```

OUTPUT:

```
The value of GLOBAL_CONSTANT is: 100
Result of computation using GLOBAL_CONSTANT: 200
```

GLOBAL SHADOW:

GLOBAL VARIABLES ACROSS FUNCTIONS:

```
#include <stdio.h>

int Num = 0;
```

```
void addNum() {
   Num += 3;
   printf("Num after addition: %d\n", Num);
}

void subtractNum() {
   Num -= 1;
   printf("Num after subtraction: %d\n", Num);
}
void multiplyNum() {
   Num *= 9;
   printf("Num after multiplication: %d\n", Num);
}

int main() {
   printf("Initial Num: %d\n", Num);
   addNum();
   subtractNum();
   multiplyNum();
   return 0;
}
```

OUTPUT:

```
Initial Num: 0
Num after addition: 3
Num after subtraction: 2
Num after multiplication: 18
```

GLOBAL VARIABLES SHARED STATE:

```
#include <stdio.h>

int counter = 0;

void incrementCounter() {
  counter++;
  printf("Counter value after increment: %d\n", counter);
}
void incrementCounterByTwo() {
  counter += 2;
  printf("Counter value after increment by 2: %d\n", counter);
}
int main() {
  printf("Initial counter value: %d\n", counter);
  incrementCounter();
  incrementCounter();
  incrementCounterByTwo();
  incrementCounter();
  return 0;
}
```

OUTPUT:

```
Initial counter value: 0
Counter value after increment: 1
Counter value after increment: 2
Counter value after increment by 2: 4
Counter value after increment: 5
```

LOCAL GLOBAL SUM:

```
#include <stdio.h>

int globalNum = 50;

void Sum(){
   int localNum = 30;
   int sum = globalNum + localNum;
   printf("Sum of global and local variables: %d\n", sum);
}

int main() {
   Sum();
   return 0;
}

/*The Global variable can be accessed anywhere in the program including any functions
and the local varibale can be accessed only within the program, Here while calling the
sum function the global available variable 5o is accessed along with that specific function's local value and
performs addition on both of it*/
```

OUTPUT:

```
Sum of global and local variables: 80
```

LOCAL SCOPE STATEMENT:

```
#include <stdio.h>

int main() {
  if (1) {
    int x = 10;
    printf("Inside if block: x = %d\n", x);
  }

  for (int i = 0; i < 1; i++) {
    int y = 20;
    printf("Inside for loop: y = %d\n", y);
  }

  return 0;
}
```

OUTPUT:

```
Inside if block: x = 10
Inside for loop: y = 20
```

LOCAL VARIABLE INITIALIZATION:

```
#include <stdio.h>

void localCheck() {
  int value = 3;
  printf("Local variable value inside function: %d\n", value);
}
```

```
int main() {
   localCheck();
   localCheck();
   localCheck();

   return 0;
}
```

OUTPUT

```
Local variable value inside function: 3
Local variable value inside function: 3
Local variable value inside function: 3
```

CONST CONDITIONAL LOOPS:

```
#include <stdio.h>

int main() {
   const int limit = 5;
   int count = 0;

   while (count < limit) {
      printf("Iteration count: %d\n", count);
      count++;
   }

   return 0;
}
```

OUTPUT:

```
Iteration count: 0
Iteration count: 1
Iteration count: 2
Iteration count: 3
Iteration count: 4
```

CONST IMMUTABLE LOOP:

```
#include <stdio.h>

int main() {
  const int stepSize = 2;

  for (int i = 0; i < 10; i += stepSize) {
    printf("%d\n", i);
  }

  return 0;
}
```

OUTPUT:

```
0
2
4
6
```

CONST LOOP INVARIANT:

```
#include <stdio.h>

int main() {
   const float PI = 3.14;

   for (int radius = 1; radius <= 5; radius++) {
      float area = PI * radius * radius;
      printf("Radius: %d, Area: %.2f\n", radius, area);
   }

   return 0;
}
```

OUTPUT:

```
Radius: 1, Area: 3.14
Radius: 2, Area: 12.56
Radius: 3, Area: 28.26
Radius: 4, Area: 50.24
Radius: 5, Area: 78.50
```

CONST LOOP LIMIT:

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

```
int main() {
   const int limit = 5;

   for (int i = 0; i < limit; i++) {
      printf("Iteration count: %d\n", i);
   }

   return 0;
}
```

OUTPUT:

```
Iteration count: 0
Iteration count: 1
Iteration count: 2
Iteration count: 3
Iteration count: 4
```

CONST NESTED LOOP:

```
#include <stdio.h>

int main() {
   const int rows = 4;
   const int cols = 5;

   for (int i = 0; i < rows; i++) {
      for (int j = 0; j < cols; j++) {
        printf("* ");
      }
      printf("\n");
    }

    return 0;
}
```

OUTPUT:

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

CONST READ ONLY POINTERS:

```
#include <stdio.h>

int main() {
    int arr[] = {10, 20, 30, 40, 50};
    const int *ptr = arr;

    for (int i = 0; i < 5; i++) {
      printf("%d\n", *ptr);
      ptr++;
    }

    return 0;
```

```c
}
```

OUTPUT:

```
10
20
30
40
50
```

CONST READ ONLY ARRAYS:

```c
#include <stdio.h>

int main() {
    const int arr[] = {1, 2, 3, 4, 5};

    for (int i = 0; i < 5; i++) {
        printf("%d ", arr[i]);
    }

    return 0;
}
```

OUTPUT:

```
1 2 3 4 5
```

STATIC VARIABLES ITERATION:

```c
#include <stdio.h>

void count_iterations() {
  static int count = 0;
  for (int i = 1; i <= 5; i++) {
    count++;
    printf("Iteration %d\n", count);
  }
}

int main() {
  count_iterations();
  count_iterations();
  return 0;
}
```

OUTPUT:

```
Iteration 1
Iteration 2
Iteration 3
Iteration 4
Iteration 5
Iteration 6
Iteration 7
Iteration 8
Iteration 9
Iteration 10
```

STATIC VARIABLES LOOP:

```c
#include <stdio.h>

void track_sum() {
    static int total = 0;
    for (int i = 1; i <= 10; i++) {
        total += i;
        printf("Total after adding %d: %d\n", i, total);
    }
}

int main() {
    track_sum();
    track_sum();
    return 0;
}
```

OUTPUT:

```
Total after adding 1: 1
Total after adding 2: 3
Total after adding 3: 6
Total after adding 4: 10
Total after adding 5: 15
Total after adding 6: 21
Total after adding 7: 28
Total after adding 8: 36
Total after adding 9: 45
Total after adding 10: 55
Total after adding 1: 56
Total after adding 2: 58

Total after adding 3: 61

Total after adding 4: 65

Total after adding 5: 70

Total after adding 6: 76

Total after adding 7: 83

Total after adding 8: 91

Total after adding 9: 100

Total after adding 10: 110
```

STATIC VARIABLES NESTED LOOPS:

```
#include <stdio.h>

void count_inner_loop_executions() {
  static int innerLoopCount = 0;
  for (int i = 1; i <= 3; i++) {
    for (int j = 1; j <= 2; j++) {
      innerLoopCount++;
      printf("Inner loop execution count: %d\n", innerLoopCount);
    }
  }
}

int main() {
  count_inner_loop_executions();
  count_inner_loop_executions();
  return 0;
}
```

OUTPUT:

```
Inner loop execution count: 1
Inner loop execution count: 2
Inner loop execution count: 3
Inner loop execution count: 4
Inner loop execution count: 5
Inner loop execution count: 6
Inner loop execution count: 7
Inner loop execution count: 8
Inner loop execution count: 9
Inner loop execution count: 10
Inner loop execution count: 11
Inner loop execution count: 12
```

STATIC VARIABLES NESTED LOOP:

```
#include <stdio.h>

void count_steps(int stepSize) {
  static int totalSteps = 0;
  for (int i = 0; i < 10; i += stepSize) {
    totalSteps++;
  }
  printf("Total steps taken so far: %d\n", totalSteps);
}
```

```
int main() {
  count_steps(1);
  count_steps(2);
  count_steps(3);
  return 0;
}
```

OUTPUT:

```
Total steps taken so far: 10
Total steps taken so far: 15
Total steps taken so far: 19
```

STATIC VARIABLES TRACK ENTRY:

```
#include <stdio.h>

void track_reentry() {
  static int reentryCount = 0;
  for (int i = 1; i <= 3; i++) {
    if (i == 2) {
      reentryCount++;
      break;
    }
  }
  printf("Reentry count: %d\n", reentryCount);
}

int main() {
  track_reentry();
  track_reentry();
  return 0;
}
```

OUTPUT:

```
Reentry count: 1
Reentry count: 2
```

STATIC VARIABLES TRACK EXIT:

```
#include <stdio.h>

void check_condition() {
    static int exitCount = 0;
    int i = 1;

    while (i <= 10) {
        if (i == 5) {
            exitCount++;
            printf("Loop exited due to condition being true. Exit count: %d\n", exitCount);
            break;
        }
        i++;
    }
}

int main() {
    check_condition();
    check_condition();

    return 0;

}
```

OUTPUT:

```
Loop exited due to condition being true. Exit count: 1

Loop exited due to condition being true. Exit count: 2
```