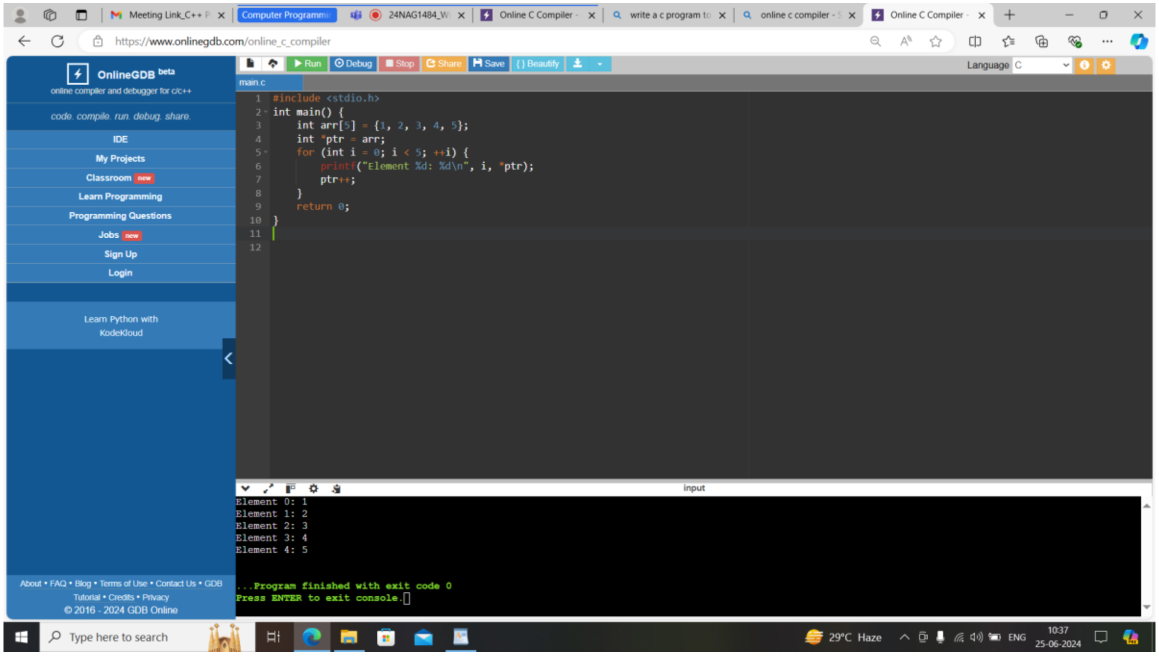
Pointer Arithmetic :-

Write a C program to create an integer array of size 5, initialize it with values from 1 to 5, and then use pointer arithmetic to print each element of the array.

```
#include <stdio.h>

int main() {

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

    int *ptr = arr;

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

        printf("Element %d: %d\n", i, *ptr);

        ptr++;

    }

    return 0;

}
```



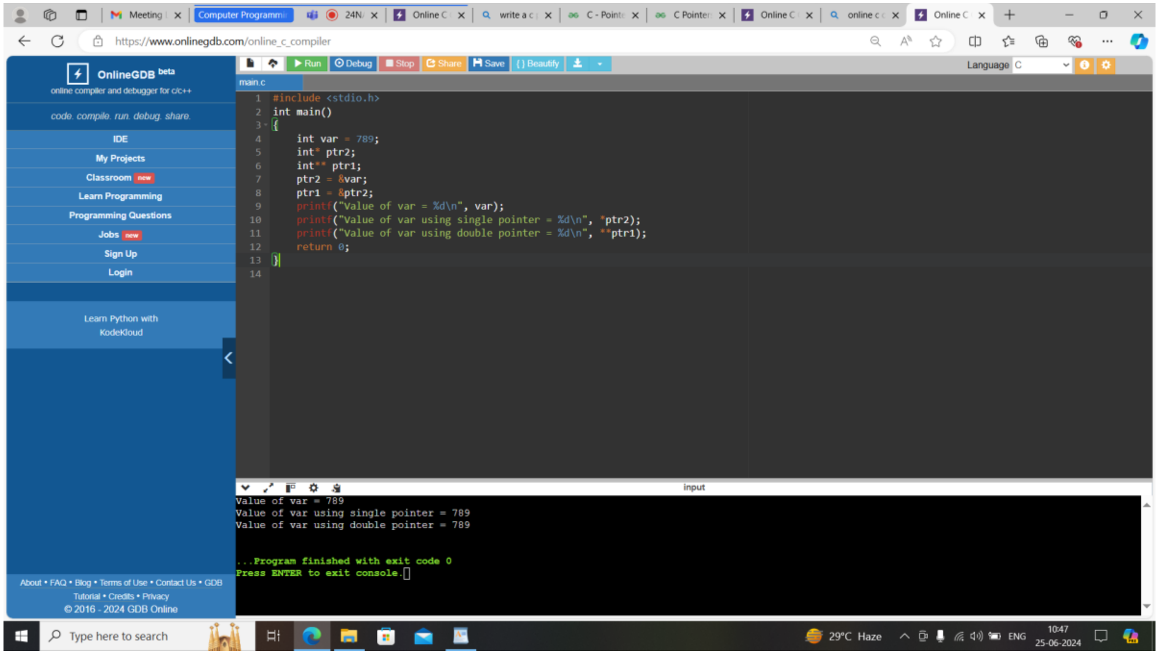
```
#include <stdio.h>
int main() {
    int arr[5] = {1, 2, 3, 4, 5};
    int *ptr = arr;
    for (int i = 0; i < 5; ++i) {
        printf("Element %d: %d\n", i, *ptr);
        ptr++;
    }
    return 0;
}
```

```
Element 0: 1
Element 1: 2
Element 2: 3
Element 3: 4
Element 4: 5


...Program finished with exit code 0
Press ENTER to exit console.
```

Pointer to Pointer :-

Write a C program to create a pointer to a pointer for an integer variable. Initialize the integer variable with a value, and then print its value using both the single pointer and the pointer to pointer.

```
include <stdio.h>

int main()

{

        int var = 789;

        int* ptr2;

        int** ptr1;

        ptr2 = &var;

        ptr1 = &ptr2;

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

        printf("Value of var using single pointer = %d\n", *ptr2);

        printf("Value of var using double pointer = %d\n", **ptr1);

        return 0;

}
```



```c
#include <stdio.h>
int main()
{
    int var = 789;
    int* ptr2;
    int** ptr1;
    ptr2 = &var;
    ptr1 = &ptr2;
    printf("Value of var = %d\n", var);
    printf("Value of var using single pointer = %d\n", *ptr2);
    printf("Value of var using double pointer = %d\n", **ptr1);
    return 0;
}
```

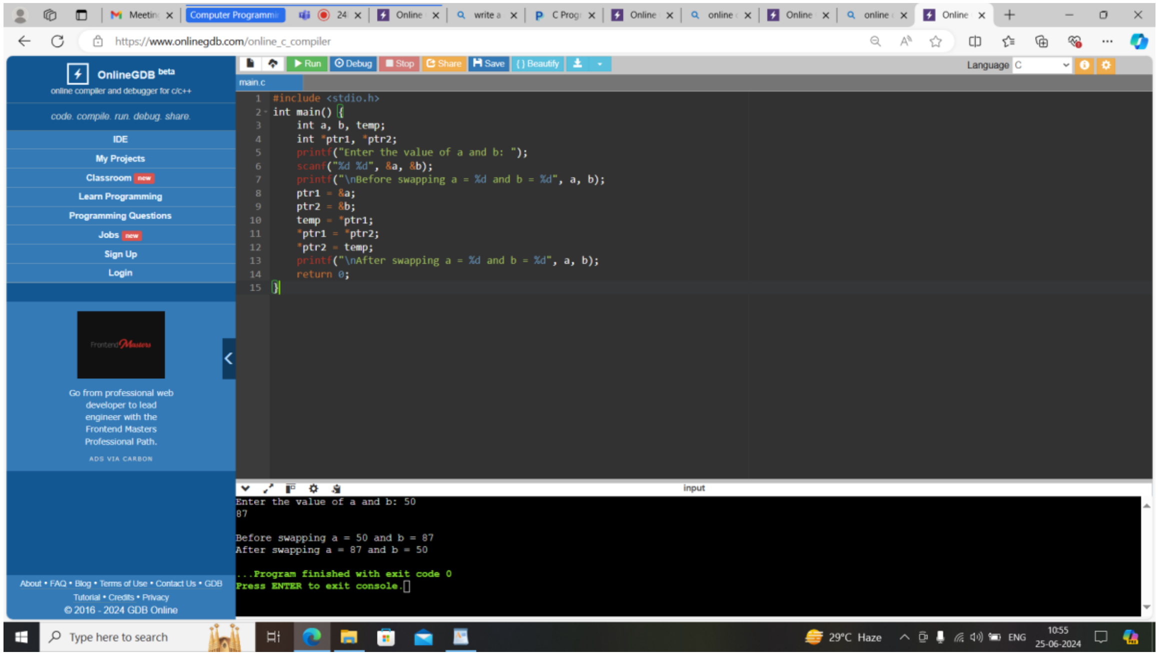
```
Value of var = 789
Value of var using single pointer = 789
Value of var using double pointer = 789


...Program finished with exit code 0
Press ENTER to exit console.
```

Pointer Function Parameters (swap) :-

Write a C function void swap(int *a, int *b) that swaps the values of two integers. Then, write a main function to test this swap function using pointer arguments.

```c
#include <stdio.h>

int main() {

    int a, b, temp;

      int *ptr1, *ptr2;

    printf("Enter the value of a and b: ");

    scanf("%d %d", &a, &b);

    printf("\nBefore swapping a = %d and b = %d", a, b);

    ptr1 = &a;

    ptr2 = &b;

    temp = *ptr1;

    *ptr1 = *ptr2;
```

```
        *ptr2 = temp;

        printf("\nAfter swapping a = %d and b = %d", a, b);

            return 0;

}
```



```
#include <stdio.h>
int main() {
    int a, b, temp;
    int *ptr1, *ptr2;
    printf("Enter the value of a and b: ");
    scanf("%d %d", &a, &b);
    printf("\nBefore swapping a = %d and b = %d", a, b);
    ptr1 = &a;
    ptr2 = &b;
    temp = *ptr1;
    *ptr1 = *ptr2;
    *ptr2 = temp;
    printf("\nAfter swapping a = %d and b = %d", a, b);
    return 0;
}
```

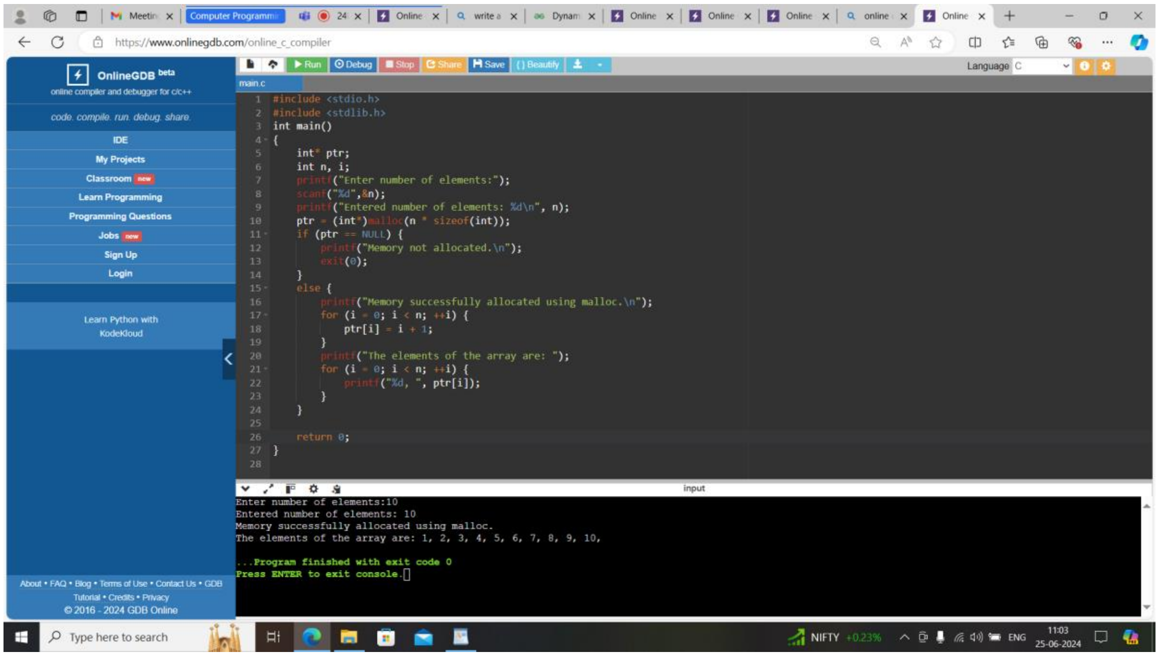
```
Enter the value of a and b: 50
87

Before swapping a = 50 and b = 87
After swapping a = 87 and b = 50

...Program finished with exit code 0
Press ENTER to exit console.
```

Dynamic Memory Allocation :-

Write a C program to dynamically allocate memory for an array of integers of size 10. Initialize the array with values from 1 to 10, then print the values and free the allocated memory.

```
#include <stdio.h>

#include <stdlib.h>

int main()

{

        int* ptr;

        int n, i;

        printf("Enter number of elements:");
```

```
    scanf("%d",&n);
    printf("Entered number of elements: %d\n", n);
    ptr = (int*)malloc(n * sizeof(int));
    if (ptr == NULL) {
        printf("Memory not allocated.\n");
        exit(0);
    }
    else {
        printf("Memory successfully allocated using malloc.\n");
        for (i = 0; i < n; ++i) {
            ptr[i] = i + 1;
        }
        printf("The elements of the array are: ");
        for (i = 0; i < n; ++i) {
            printf("%d, ", ptr[i]);
        }
    }

    return 0;
}
```



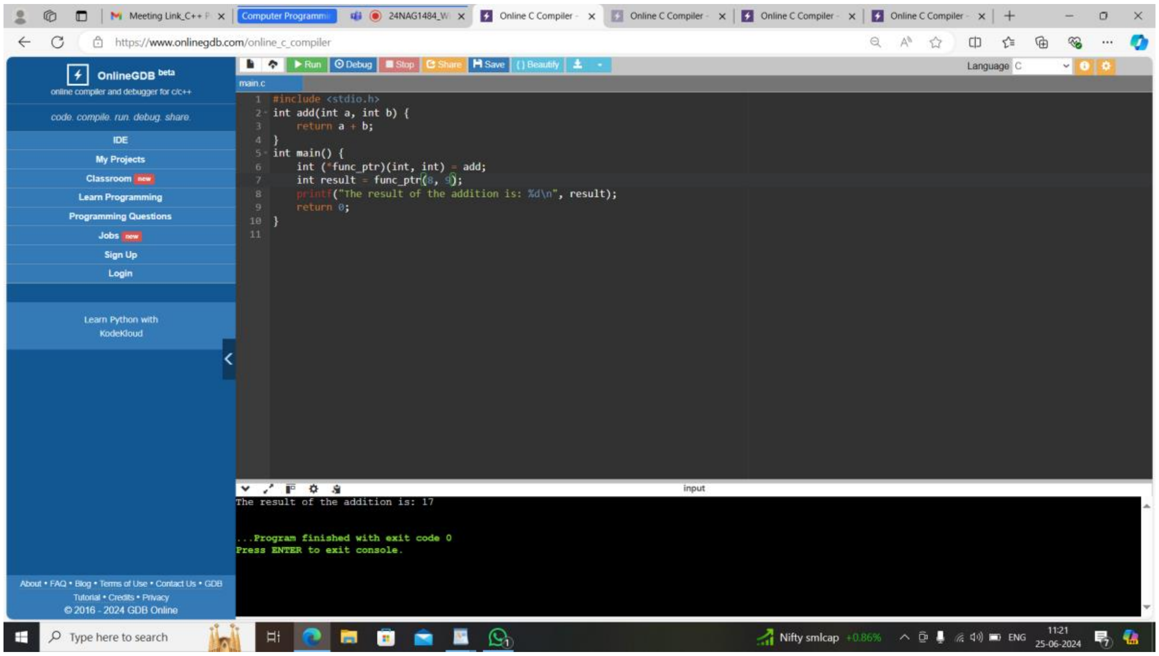
```c
#include <stdio.h>
#include <stdlib.h>
int main()
{
    int* ptr;
    int n, i;
    printf("Enter number of elements:");
    scanf("%d",&n);
    printf("Entered number of elements: %d\n", n);
    ptr = (int*)malloc(n * sizeof(int));
    if (ptr == NULL) {
        printf("Memory not allocated.\n");
        exit(0);
    }
    else {
        printf("Memory successfully allocated using malloc.\n");
        for (i = 0; i < n; ++i) {
            ptr[i] = i + 1;
        }
        printf("The elements of the array are: ");
        for (i = 0; i < n; ++i) {
            printf("%d, ", ptr[i]);
        }
    }

    return 0;
}
```

```
Enter number of elements:10
Entered number of elements: 10
Memory successfully allocated using malloc.
The elements of the array are: 1, 2, 3, 4, 5, 6, 7, 8, 9, 10,

...Program finished with exit code 0
Press ENTER to exit console.
```

Pointer to Function :-

Write a C program to create a function pointer that points to a function int add(int, int). Use the function pointer to call the add function and print the result.

```c
#include <stdio.h>

int add(int a, int b) {

        return a + b;

}

int main() {

    int (*func_ptr)(int, int) = add;

      int result = func_ptr(8, 9);

        printf("The result of the addition is: %d\n", result);

        return 0;

}
```



```c
#include <stdio.h>
int add(int a, int b) {
    return a + b;
}
int main() {
    int (*func_ptr)(int, int) = add;
    int result = func_ptr(8, 9);
    printf("The result of the addition is: %d\n", result);
    return 0;
}
```

```
The result of the addition is: 17


...Program finished with exit code 0
Press ENTER to exit console.
```

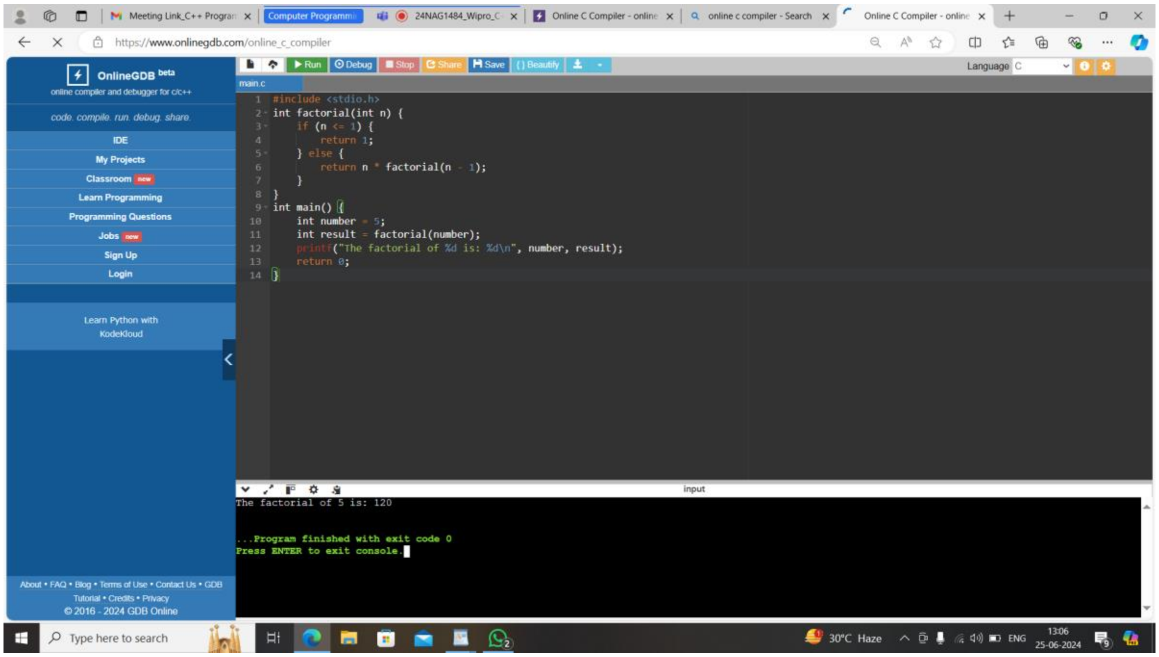
Functions

Recursive Function:-

Write a C function int factorial(int n) that calculates the factorial of a given number using recursion. Test this function in the main program by calculating and printing the factorial of 5.

```c
#include <stdio.h>

int factorial(int n) {

    if (n <= 1) {

        return 1;

    } else {

        return n * factorial(n - 1);

    }

}

int main() {

    int number = 5;
```

```
        int result = factorial(number);

        printf("The factorial of %d is: %d\n", number, result);

        return 0;

    }
```



Meeting Link_C++ Program
Computer Programmi
24NAG1484_Wipro_C
Online C Compiler - online
online c compiler - Search
Online C Compiler - online
https://www.onlinegdb.com/online_c_compiler
OnlineGDB beta
online compiler and debugger for c/c++
code. compile. run. debug. share.
IDE
My Projects
Classroom new
Learn Programming
Programming Questions
Jobs new
Sign Up
Login
Learn Python with KodeKloud
About • FAQ • Blog • Terms of Use • Contact Us • GDB Tutorial • Credits • Privacy
© 2016 - 2024 GDB Online
Run
Debug
Stop
Share
Save
{ } Beautify
Language C
main.c
input
Type here to search
30°C Haze
ENG
13:06
25-06-2024

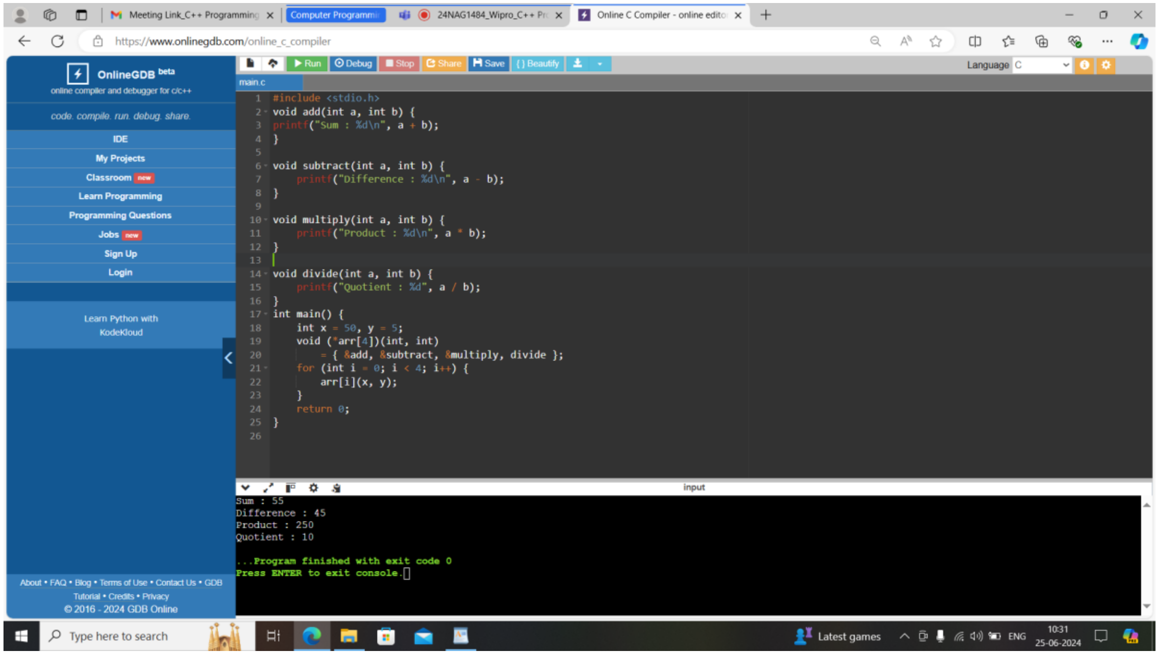
```
#include <stdio.h>
int factorial(int n) {
    if (n <= 1) {
        return 1;
    } else {
        return n * factorial(n - 1);
    }
}
int main() {
    int number = 5;
    int result = factorial(number);
    printf("The factorial of %d is: %d\n", number, result);
    return 0;
}
```

```
The factorial of 5 is: 120


...Program finished with exit code 0
Press ENTER to exit console.
```

Array of Function Pointer :-

Write a C program to create an array of function pointers, where each function takes two integers as arguments and returns an integer. Include functions for addition, subtraction, multiplication, and division. Use the array to perform these operations on two integers and print the results.

```
#include <stdio.h>

void add(int a, int b) {

printf("Sum : %d\n", a + b);

}

void subtract(int a, int b) {

        printf("Difference : %d\n", a - b);
}
void multiply(int a, int b) {
    printf("Product : %d\n", a * b);
}
void divide(int a, int b) {
    printf("Quotient : %d", a / b);
}
int main() {
    int x = 50, y = 5;
    void (*arr[4])(int, int)
        = { &add, &subtract, &multiply, divide };
    for (int i = 0; i < 4; i++) {
        arr[i](x, y);
    }
    return 0;
}
```



```c
#include <stdio.h>
void add(int a, int b) {
printf("Sum : %d\n", a + b);
}

void subtract(int a, int b) {
    printf("Difference : %d\n", a - b);
}

void multiply(int a, int b) {
    printf("Product : %d\n", a * b);
}

void divide(int a, int b) {
    printf("Quotient : %d", a / b);
}
int main() {
    int x = 50, y = 5;
    void (*arr[4])(int, int)
        = { &add, &subtract, &multiply, divide };
    for (int i = 0; i < 4; i++) {
        arr[i](x, y);
    }
    return 0;
}
```

```
Sum : 55
Difference : 45
Product : 250
Quotient : 10

...Program finished with exit code 0
Press ENTER to exit console.
```

Higher-Order Functions:

Write a C function void applyFunction(int arr[], int size, void (*func)(int *)) that takes an array, its size, and a pointer to a function that operates on each element of the array. Write a sample function to double the value of each element and use applyFunction to apply it to an array.

Static Variables in Functions:

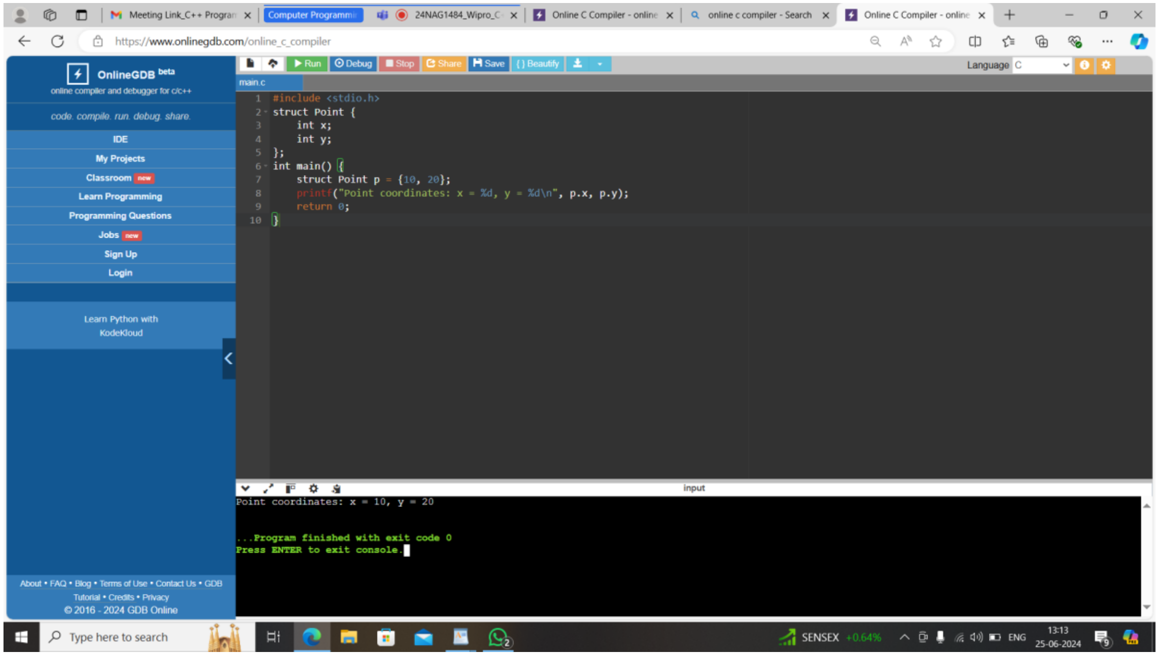
Which a C function that uses a static variable to count how many times the function has been called. Test this function in the main program by calling it multiple times and printing the count.

Structures

Structure Basics:

Define a structure struct Point with two integer members x and y. Write a C program to create a Point variable, initialize it with values, and print the values.

```
#include <stdio.h>

struct Point {

        int x;

        int y;
};

int main() {

        struct Point p = {10, 20};

        printf("Point coordinates: x = %d, y = %d\n", p.x, p.y);

        return 0;

}
```



Meeting Link_C++ Progran
Computer Programmi
24NAG1484_Wipro_C
Online C Compiler - online
online c compiler - Search
Online C Compiler - online
https://www.onlinegdb.com/online_c_compiler
OnlineGDB beta
online compiler and debugger for c/c++
code. compile. run. debug. share.
IDE
My Projects
Classroom new
Learn Programming
Programming Questions
Jobs new
Sign Up
Login
Learn Python with KodeKloud
About • FAQ • Blog • Terms of Use • Contact Us • GDB Tutorial • Credits • Privacy
© 2016 - 2024 GDB Online
Run
Debug
Stop
Share
Save
{ } Beautify
Language C
main.c
input
Type here to search
SENSEX +0.64%
ENG
13:13
25-06-2024

```
#include <stdio.h>
struct Point {
    int x;
    int y;
};
int main() {
    struct Point p = {10, 20};
    printf("Point coordinates: x = %d, y = %d\n", p.x, p.y);
    return 0;
}
```

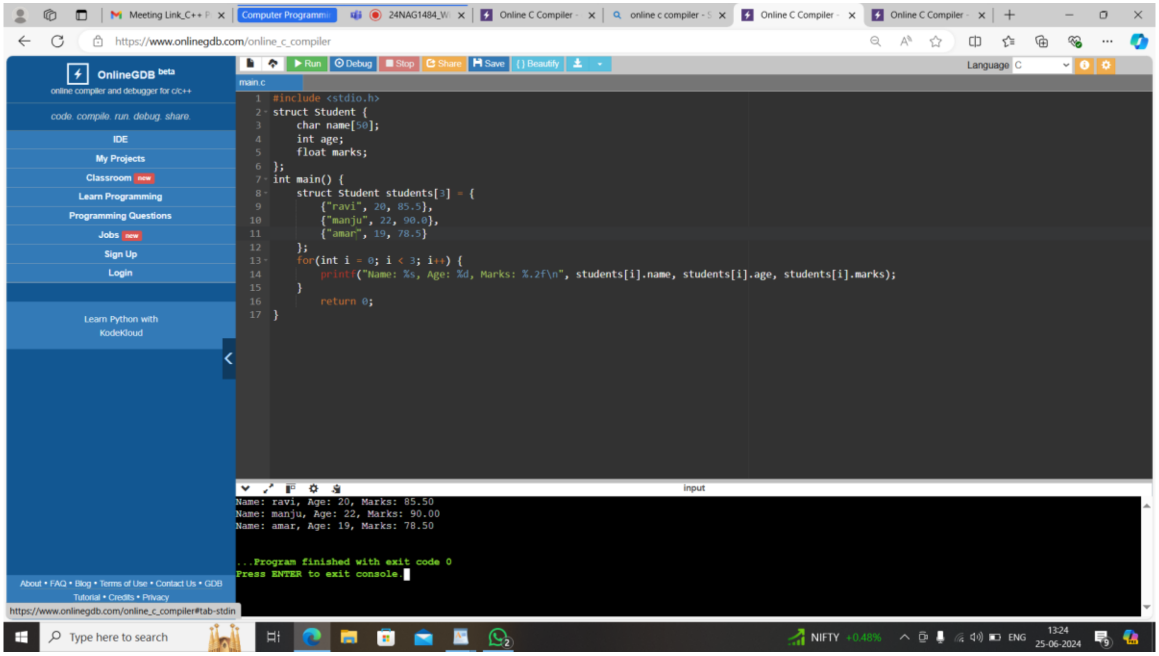
```
Point coordinates: x = 10, y = 20


...Program finished with exit code 0
Press ENTER to exit console.
```

Array of Structures:-

Write a C program to define a structure struct Student with members name, age, and marks. Create an array of 3 students, initialize them with values, and print the details of each student

```
#include <stdio.h>

struct Student {

        char name[50];

        int age;

        float marks;

};
```

```
int main() {
    struct Student students[3] = {
        {"ravi", 20, 85.5},
        {"manju", 22, 90.0},
        {"amar", 19, 78.5}
 };
for(int i = 0; i < 3; i++) {
printf("Name: %s, Age: %d, Marks: %.2f\n", students[i].name, students[i].age, students[i].marks);
    }
    return 0;
}
```



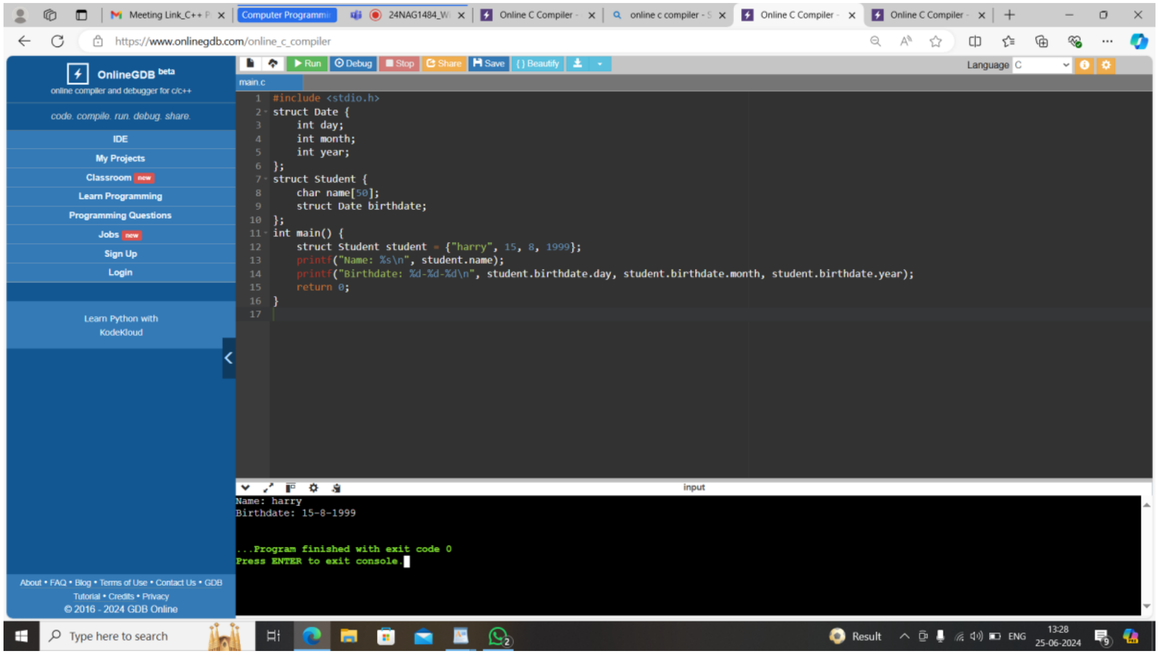
```
#include <stdio.h>
struct Student {
    char name[50];
    int age;
    float marks;
};
int main() {
    struct Student students[3] = {
        {"ravi", 20, 85.5},
        {"manju", 22, 90.0},
        {"amar", 19, 78.5}
    };
    for(int i = 0; i < 3; i++) {
        printf("Name: %s, Age: %d, Marks: %.2f\n", students[i].name, students[i].age, students[i].marks);
    }
        return 0;
}
```

```
Name: ravi, Age: 20, Marks: 85.50
Name: manju, Age: 22, Marks: 90.00
Name: amar, Age: 19, Marks: 78.50


...Program finished with exit code 0
Press ENTER to exit console.
```

Nested Structures:

Write a C program to define a structure struct Date with members day, month, and year, and another structure struct Student with members name and birthdate of type struct Date. Create a Student variable, initialize it with values, and print the student's details including the birthdate.

```c
#include <stdio.h>
struct Date {
    int day;
    int month;
    int year;
};
struct Student {
    char name[50];
    struct Date birthdate;
};
int main() {
    struct Student student = {"harry", 15, 8, 1999};
    printf("Name: %s\n", student.name);
    printf("Birthdate: %d-%d-%d\n", student.birthdate.day, student.birthdate.month, student.birthdate.year);
    return 0;
}
```



```c
#include <stdio.h>
struct Date {
    int day;
    int month;
    int year;
};
struct Student {
    char name[50];
    struct Date birthdate;
};
int main() {
    struct Student student = {"harry", 15, 8, 1999};
    printf("Name: %s\n", student.name);
    printf("Birthdate: %d-%d-%d\n", student.birthdate.day, student.birthdate.month, student.birthdate.year);
    return 0;
}
```

```
Name: harry
Birthdate: 15-8-1999


...Program finished with exit code 0
Press ENTER to exit console.
```