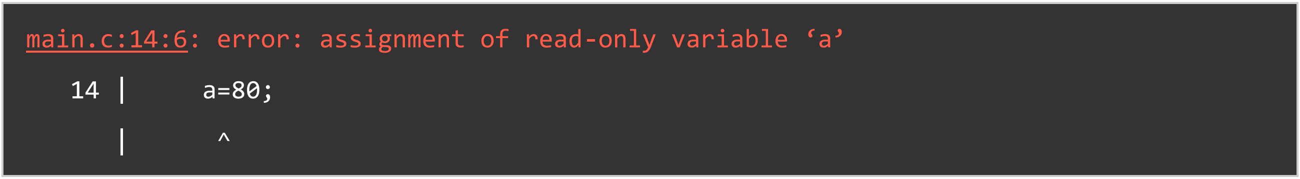
```
#include <stdio.h>
int main()
{
   int a=50;
   printf("oo1 a=%d\n",a);
   a=80;
   printf("oo2 a=%d\n",a);
   return 0;
}
```

```
oo1 a=50
oo2 a=80
```

CONST

```
#include <stdio.h>
int main()
{
   int const a=50;
   printf("oo1 a=%d\n",a);
   a=80;
   printf("oo2 a=%d\n",a);
   return 0;
}
```



```
main.c:14:6: error: assignment of read-only variable ‘a’
   14 |     a=80;
      |      ^
```

CONST AND POINTERS

```
#include <stdio.h>
int main()
{
    Int const a=50;
    printf("oo1 a=%d\n",a);
    int *p;
    p=&a;
    *p=80;
    printf("oo2 a=%d\n",a);
    return 0;
}
```

```
oo1 a=50
oo2 a=80
```

GLOBAL CONST

```
#include <stdio.h>
int const a=50;
int main()
{
    printf("oo1 a=%d\n",a);
    int *p;
    p=&a;
    *p=80;
    printf("oo2 a=%d\n",a);
```

```
   return 0;

}
```

When we decalare a as global const it is stored in rom and it is read only m/y so the pointer cant be access the variable and modify it.If it was declared as local scope it was stored in ram and pointer can access and modify it.

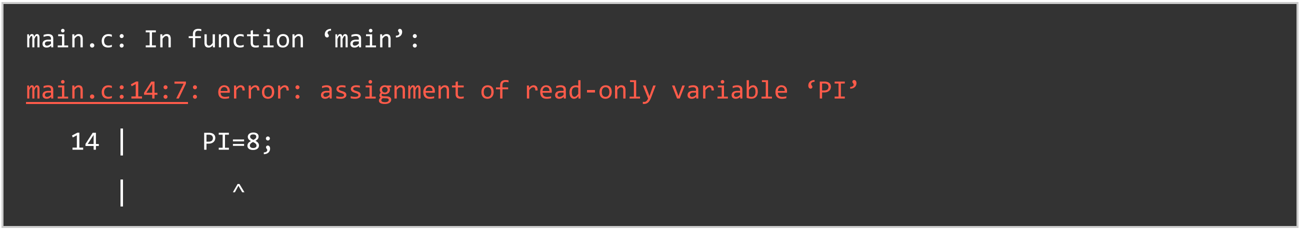
ASSIGNMENTS

Assignment 1: Constant Variable Declaration
Objective: Learn to declare and initialize constant variables.
Write a program that declares a constant integer variable for the value of Pi (3.14) and prints it.
Ensure that any attempt to modify this variable results in a compile-time error.

Program

```
#include <stdio.h>

int main()

{

    float const PI=3.14;

    printf("oo1 a=%f\n",PI);

    PI=8;

    printf("oo2 a=%f\n",PI);

    return 0;

}
```

Output



```
main.c: In function ‘main’:
main.c:14:7: error: assignment of read-only variable ‘PI’
   14 |     PI=8;
      |       ^
```

Assignment 2: Using const with Pointers

Objective: Understand how to use const with pointers to prevent modification of pointed values.

Create a program that uses a pointer to a constant integer. Attempt to modify the value through the pointer and observe the compiler's response.

Program

```
#include <stdio.h>

int main()

{

    int const a=28;

    printf("oo1 a=%d\n",a);

    int *p;

    p=&a;

    *p=18;

    printf("oo2 a=%d\n",a);

    return 0;

}
```

```
oo1 a=28

oo2 a=18
```

Assignment 3: Constant Pointer

Objective: Learn about constant pointers and their usage.

Write a program that declares a constant pointer to an integer and demonstrates that you cannot change the address stored in the pointer.

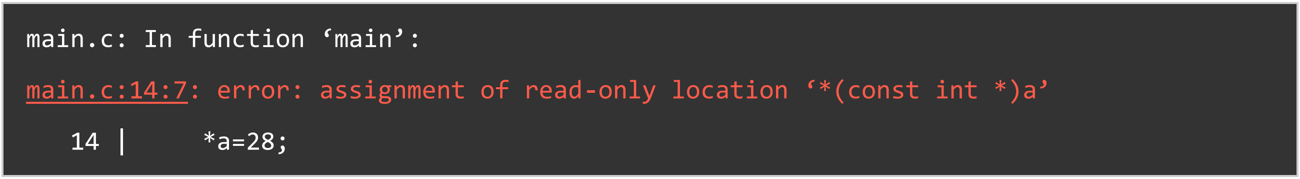
```
 #include <stdio.h>
int main()
{
    int* const a=(int*)0X00028;
    *a=28;
    printf("oo1 a=%d\n",a);
    int *p;
    p=&a;
    *p=18;
    printf("oo2 a=%d\n",a);
    return 0;
}
```

```
#include <stdio.h>

int main() {

    int value = 28;
    int* const a = &value;
    *a = 28;
    printf("oo1: Value = %d, Address of a = %p\n", *a, (void*)a);
     *a = 18;
    printf("oo2: Value = %d, Address of a = %p\n", *a, (void*)a);
    return 0;

}
```

```
oo1: Value = 28, Address of a = 0x7fffd43bd3fc

oo2: Value = 18, Address of a = 0x7fffd43bd3fc
```

Assignment 4: Constant Pointer to Constant Value
Objective: Combine both constant pointers and constant values.
Create a program that declares a constant pointer to a constant integer. Demonstrate that neither the pointer nor the value it points to can be changed.

```
 #include <stdio.h>

#include<limits.h>

int main()

{

    int const* const a=(int*)0X00028;

    *a=28;

    printf("oo1 a=%d\n",a);

    int *p;

    p=&a;

    *p=18;

    printf("oo2 a=%d\n",a);

    return 0;

}
```



```
main.c: In function ‘main’:
main.c:14:7: error: assignment of read-only location ‘*(const int *)a’
   14 |     *a=28;
```

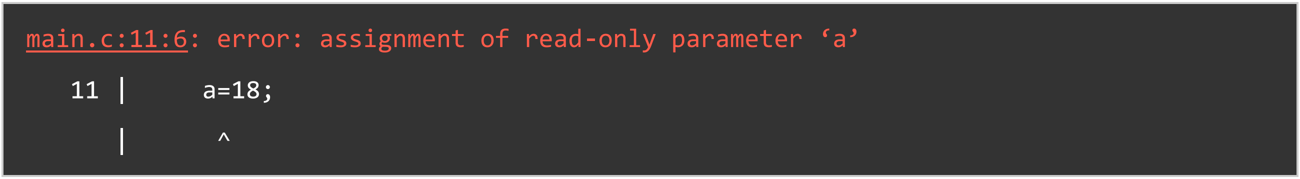
```
#include <stdio.h>


int main()
{


    const int value = 28;
    const int* const a = &value;


    printf("a points to value: %d\n", *a);
    printf("Pointer 'a' is constant, and the value it points to is also constant.\n");


    return 0;
}
```

```
a points to value: 28
Pointer 'a' is constant, and the value it points to is also constant.
```

Assignment 5: Using const in Function Parameters
Objective: Understand how to use const with function parameters.
Write a function that takes a constant integer as an argument and prints its value. Attempting to modify this parameter inside the function should result in an error.

```
 #include<stdio.h>
void func(int const a){
    a=18;
    printf("The value of a=%d",a);
}
```

```
    int main()

{

    func(5);

    return 0;

}
```



```
main.c:11:6: error: assignment of read-only parameter ‘a’
   11 |     a=18;
      |      ^
```

Assignment 6: Array of Constants
Objective: Learn how to declare and use arrays with const.
Create an array of constants representing days of the week. Print each day using a loop, ensuring that no modifications can be made to the array elements.

```
 #include <stdio.h>



int main() {



    const char* daysOfWeek[] = {

        "Sunday", "Monday", "Tuesday", "Wednesday",

        "Thursday", "Friday", "Saturday"

    };



    for (int i = 0; i < 7; ++i) {

        printf("%s\n", daysOfWeek[i]);

    }
    return 0;

}
```

```
Sunday

Monday

Tuesday

Wednesday

Thursday

Friday

Saturday
```

Assignment 7: Constant Expressions
Objective: Understand how constants can be used in expressions.
Write a program that uses constants in calculations, such as calculating the area of a circle using const.

```
 #include <stdio.h>

#define PI 3.14

int main() {



    const int r=5;

    int area=PI*r*r;

    printf("The area is %d",area);

    return 0;

}
```

```
The area is 78
```

Assignment 8: Constant Variables in Loops

Objective: Learn how constants can be used within loops for fixed iterations.
Create a program that uses a constant variable to define the number of iterations in a loop, ensuring it cannot be modified during execution.

```
 #include <stdio.h>

#define ITER 5

int main() {

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

        printf("Iteration %d\n", i + 1);

    }

    return 0;

}
```

Output

```
Iteration 1

Iteration 2

Iteration 3

Iteration 4

Iteration 5
```

Assignment 9: Constant Global Variables

Objective: Explore global constants and their accessibility across functions.
Write a program that declares a global constant variable and accesses it from multiple functions without modifying its value.

```
#include <stdio.h>

const int count = 10;

void printCount() {

 printf("The count is: %d\n",count);
}

int main() {

 printf("In main function, count= is: %d\n",count);

 printCount();

 return 0;

}
```

```
In main function, count= is: 10

The count is: 10
```

ARRAYS

```
#include <stdio.h>

int main() {


    int A[5];

    printf("size of int:%d\n",sizeof(int));

    printf("A=%d\n",sizeof(A));

     printf("A=%d\n",A);

    return 0;

}
```

```
size of int:4

A=20

A=-668509376
```

ARRAY MEMORY LOCATIONS ARE CONTIGIOUS

```
#include <stdio.h>
int main() {

    int A[5];
    printf("size of int:%d\n",sizeof(int));
    printf("A=%d\n",sizeof(A));
    for(int i=0;i<=4;i++){
     printf("A=%p--->\n",(A+i));
    }
    return 0;
}
```

```
size of int:4
A=20
A=0x7ffc09633910--->
A=0x7ffc09633914--->
A=0x7ffc09633918--->
A=0x7ffc0963391c--->
A=0x7ffc09633920--->
```

```
#include <stdio.h>
int main() {

    int A[5];
    printf("Enter the elements in the array A\n");
    for(int i=0;i<5;i++){
     scanf("%d",&A[i]);
     printf("\n");
    }
    for(int j=0;j<5;j++){
     printf("A[%d]=%d\n",j,A[j]);
    }
    return 0;
}
```

```
Enter the elements in the array A
1

2

3

4

5

A[0]=1
A[1]=2
A[2]=3
A[3]=4
A[4]=5
```

Qn.Average of grades

```
#include <stdio.h>
int main() {
    int grades[10];
    int count = 10;

    long sum = 0;

    float average = 0.0f;

    printf("\nEnter the 10 grades:\n");

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

    {

        printf("%2u> ",i + 1);

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

        sum += grades[i];


}

        average = (float)sum/count;

        printf("\nAverage of the ten grades entered is: %.2f\n", average);


return 0;

}
```

```
Enter the 10 grades:

 1> 20

 2> 60

 3> 50

 4> 90

 5> 36

 6> 67

 7> 39

 8> 45

 9> 69

10> 100


Average of the ten grades entered is: 57.60
```

INITIALISING ARRAYS

```
#include <stdio.h>
int main() {
    int A[10]={1,2,3};

    for(int i = 0; i <10; ++i)
    {
        printf("%d\n",A[i]);
    }
return 0;
}
```

```
1
2
3
0
0
0
0
0
0
0
```

```
#include <stdio.h>
int main() {
    int A[10]={[2]=3,[0]=1,[1]=2};

    for(int i = 0; i <10; ++i)
```

```
  {
      printf("%d\n",A[i]);
  }
return 0;
}
```

```
1
2
3
0
0
0
0
0
0
0
```

Example of traditional initialization

```
#include <stdio.h>
#define MONTHS 12
int main(void)
{
int days[MONTHS] = {31,28,31,30,31,30,31,31,30,31,30,31}; int index;

for (index = 0; index < MONTHS; index++)
printf("Month %d has %2d days.\n", index +1, days[index]);
return 0;

}
```

```
Month 1 has 31 days.
Month 2 has 28 days.
Month 3 has 31 days.
Month 4 has 30 days.
Month 5 has 31 days.
Month 6 has 30 days.
Month 7 has 31 days.
Month 8 has 31 days.
Month 9 has 30 days.
Month 10 has 31 days.
Month 11 has 30 days.
Month 12 has 31 days.
```

Example using designated initialization

```
#include <stdio.h>
#define MONTHS 12
int main(void)
{
   int days[MONTHS] = {31,28, [4] = 31,30,31, [1] = 29}; int i;
   for (i = 0; i < MONTHS; i++)
   printf("%d %d\n", i + 1, days[i]);
   return 0;
}
```

```
1 31
```

```
2 29

3 0

4 0

5 31

6 30

7 31

8 0

9 0

10 0

11 0

12 0
```

Qn.Initialising all elements to the same value

```
#include <stdio.h>

int main (void)

{

    int array_values[(10)]= { 0, 1, 4, 9, 16 };

    int i;

    for (i=5; i<10; ++i)

        array_values[i] = i *i;

    for (i = 0; i < 10; ++i)

    printf("array_values[i] = %i\n", i, array_values[i]);

    return 0;


}
```

```
array_values[i] = 0

array_values[i] = 1

array_values[i] = 2
```

```
array_values[i] = 3
array_values[i] = 4
array_values[i] = 5
array_values[i] = 6
array_values[i] = 7
array_values[i] = 8
array_values[i] = 9
```

Task

Task: Initializing Arrays

Requirements

* In this challenge, you are going to create a program that will find all the prime numbers from 3-100

*there will be no input to the program

* The output will be each prime number separated by a space on a single line

* You will need to create an array that will store each prime number as it is generated

* You can hard-code the first two prime numbers (2 and 3) in the primes array

* You should utilize loops to only find prime numbers up to 100 and a loop to print out the primes array

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

```
#include <stdbool.h>
#include <math.h>


bool is_prime(int n) {
    if (n <= 1) {
        return false;
    }
    if (n == 2) {
        return true;
    }
    if (n % 2 == 0) {
        return false;
    }

    for (int i = 3; i <= sqrt(n); i += 2) {
        if (n % i == 0) {
            return false;
        }
    }
    return true;
}

int main() {

    int primes[100];
    int prime_count = 0;
    primes[prime_count++] = 2;
    primes[prime_count++] = 3;
    for (int num = 4; num <= 100; num++) {
        if (is_prime(num)) {
            primes[prime_count++] = num;
        }
    }
        for (int i = 0; i < prime_count; i++) {
        printf("%d", primes[i]);
        if (i < prime_count - 1) {
            printf(" ");
        }
    }

    printf("\n");
    return 0;
}
```

```
   2 3 5 7 11 13 17 19 23 29 31 37 41 43 47 53 59 61 67 71 73 79 83 89 97
```

Qn. Create a program that reverses the elements of an array. Prompt the user to enter values and print both the original and reversed arrays.

```
#include <stdio.h>
int main() {
int n;
printf("Enter the size of the array: ");
scanf("%d", &n);
int arr[n];
printf("Enter the elements in the array:\n");
for (int i = 0; i < n; i++) {
printf("Element %d: ", i + 1);
scanf("%d", &arr[i]);
}
// Reverse the array
int arr2[n];
for (int i = 0; i < n; i++) {
arr2[i] = arr[n - 1 - i];
}
// Print reversed array
printf("The reversed array elements are:\n");
for (int i = 0; i < n; i++) {
printf("%d ", arr2[i]);
}
return 0;
}
```

Output

```
Enter the size of the array: 10
Enter the elements in the array:
Element 1: 3
Element 2: 4
Element 3: 5
Element 4: 6
Element 5: 7
Element 6: 8
Element 7: 9
Element 8: 2
Element 9: 1
Element 10: 21
The reversed array elements are:
21 1 2 9 8 7 6 5 4 3
```

Qn. Write a program that to find the maximum element in an array of integers. The program should prompt the user for input and display the maximum value.

```c
#include <stdio.h>
int main() {
int n;
printf("Enter the size of the array: ");
scanf("%d", &n);
int arr[n];
printf("Enter the elements of the array:\n");
for (int i = 0; i < n; i++) {
printf("Element %d: ", i + 1);
scanf("%d", &arr[i]);
}
int max = arr[0];
for (int i = 1; i < n; i++) {
if (arr[i] > max) {
max = arr[i];
}
}
printf("The maximum element in the array is: %d\n", max);
return 0;
}
```

```
Enter the size of the array: 5
Enter the elements of the array:
Element 1: 3
Element 2: 5
Element 3: 8
Element 4: 3
Element 5: 1
```

```
The maximum element in the array is: 8
```

Qn. Write a program that counts and displays how many times a specific integer appears in an array entered by the user.

```
#include <stdio.h>
int main() {
int n, searchNum, count = 0;
printf("Enter the size of the array: ");
scanf("%d", &n);
int arr[n];
printf("Enter the elements of the array:\n");
for (int i = 0; i < n; i++) {
printf("Element %d: ", i + 1);
scanf("%d", &arr[i]);
}
printf("Enter the number to search for: ");
scanf("%d", &searchNum);
for (int i = 0; i < n; i++) {
if (arr[i] == searchNum) {
count++;
}
}
printf("The number %d appears %d time(s) in the array.\n", searchNum, count);
return 0;
}
```

```
Enter the size of the array: 6
Enter the elements of the array:
Element 1: 3
Element 2: 4
Element 3: 6
Element 4: 3
Element 5: 3
Element 6: 80
Enter the number to search for: 3
The number 3 appears 3 time(s) in the array.
```

Qn. Requirements

* In this challenge, you are to create a C program that uses a two-dimensional array in a weather program.

array is %0

* This program will find the total rainfall for each year, the average yearly rainfall, and the average rainfall for each month

* Input will be a 2D array with hard-coded values for rainfall amounts for the past 5 years

* The array should have 5 rows and 12 columns

●rainfall amounts can be floating point numbers

Answer:

```
#include<stdio.h>
#define YEARS 5
#define MONTHS 12
int main() {
float rainfall[YEARS][MONTHS];
float yearlyTotals[YEARS] = {0};
float totalRainfall = 0;
printf("Enter the rainfall data for each month (in inches):\n");
for (int year = 0; year < YEARS; year++) {
printf("Year 201%d:\n", year);
for (int month = 0; month < MONTHS; month++) {
```

```
printf(" Month %d: ", month + 1);
scanf("%f", &rainfall[year][month]);
yearlyTotals[year] += rainfall[year][month];
}
totalRainfall += yearlyTotals[year];
}
printf("\nYEAR RAINFALL (inches)\n");
for (int year = 0; year < YEARS; year++) {
printf("201%d %.1f\n", year, yearlyTotals[year]);
}
float yearlyAverage = totalRainfall / YEARS;
printf("\nThe yearly average is %.1f inches.\n", yearlyAverage);
printf("\nMONTHLY AVERAGES:\n");
const char *months[MONTHS] = {"Jan", "Feb", "Mar", "Apr", "May", "Jun", "Jul", "Aug", "Sep",
"Oct", "Nov", "Dec"};
for (int month = 0; month < MONTHS; month++) {
float monthlyTotal = 0;
for (int year = 0; year < YEARS; year++) {
monthlyTotal += rainfall[year][month];
}
printf("%s %.1f\n", months[month], monthlyTotal / YEARS);
}
return 0;
}
```