Type Qualifier **const: make it only read only**

```
#include <stdio.h>
int main()
{  int  const a=40;
   printf("%d\n",a);
   a=50;
   printf("%d\n",a);

   return 0;
}
```

```
main.c: In function ‘main’:
main.c:14:6: error: assignment of read-only variable ‘a’
  14 |    a=50;
     |    ^
```

```
#include <stdio.h>
int main()
{  int  const a=40;
   printf("%d\n",a);
   int *p=&a;
   *p=50;
   printf("%d\n",a);

   return 0;
}
```

```
main.c: In function ‘main’:
main.c:14:12: warning: initialization discards ‘const’
qualifier from pointer target type [-Wdiscarded-qualifiers]
  14 |    int *p=&a;
     |        ^
40
50
```

Note: here we can edit even if we us const using pointers
Bcoz int const is storing value in ram;

```
#include <stdio.h>
int  const a=40;
int main()
{
   printf("%d\n",a);
   int *p=&a;
   *p=50;
   printf("%d\n",a);

   return 0;
}
```

Note: here we cant edit using pointers
Bcoz int const is storing value in rom or flash memory;

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.

```
#include <stdio.h>

int main()
{  float const pi=3.14;
   printf("%f\n",pi);
   pi=2.5;
   printf("%f\n",pi);

   return 0;
}
```

```
main.c: In function ‘main’:
main.c:14:7: error: assignment of read-only variable
‘pi’
   14 |     pi=2.5;
      |       ^
```

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.

```
#include <stdio.h>
int main()
{  int  const a=40;
   printf("%d\n",a);
   int *p=&a;
   *p=50;
   printf("%d\n",a);

   return 0;
}
```

```
main.c: In function ‘main’:
main.c:5:12: warning: initialization discards ‘const’ qualifier from
pointer target type [-Wdiscarded-qualifiers]
   5 |    int *p=&a;
     |        ^
40
50
```

```
#include <stdio.h>
int  const a=40;
int main()
{
   printf("%d\n",a);
   int *p=&a;
   *p=50;
   printf("%d\n",a);

   return 0;
}
```

```
main.c: In function ‘main’:
main.c:6:12: warning: initialization discards ‘const’ qualifier from
pointer target type [-Wdiscarded-qualifiers]
   6 |    int *p=&a;
     |        ^
40
```

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 a,b;

   int *const p=&a;
   printf("%p\n",p);
    p=&b;
    printf("%p\n",p);

   return 0;
}
```

```
main.c: In function ‘main’:
main.c:9:7: error: assignment of read-only variable ‘p’
    9 |       p=&b;
      |
```

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.

```c
#include <stdio.h>

int main() {
   int const a = 10;
   const int *const p = &a;
   printf("Before trying to change 'a' value:\n");
   printf("Value of a: %d\n", a);
   printf("Value pointed to by p: %d\n", *p);
    a = 30;
   return 0;
}
```

```
main.c: In function ‘main’:
main.c:15:8: error: assignment of read-only variable ‘a’
   15 |       a = 30;
      |
```

****

```c
#include <stdio.h>
int main() {
   int const a = 10;
   int const b=30;
   const int *const p = &a;
   printf("Before trying to change address:\n");
   printf("Value of a: %ls\n", &a);
   printf("Value pointed to by p: %p\n", p);
    p = &b;
   return 0;
}
```

```
main.c: In function ‘main’:
main.c:16:8: error: assignment of read-only variable ‘p’
   16 |       p = &b;
      |
```

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.

```c
 #include <stdio.h>
void printConstant(const int num) {
   printf("The value of num is: %d\n", num);
   num = 20;
}

int main() {
   int x = 10;
   printConstant(x);
   return 0;
}
```

```
main.c: In function ‘printConstant’:
main.c:4:9: error: assignment of read-only parameter ‘num’
    4 |     num = 20;
      |
```

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.

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 3.14159
int main() {
   const float radius = 5.0;
   const float area = PI * radius * radius;
   printf("The area of the circle with radius %.2f is: %.2f\n", radius, area);
   return 0;
}
```

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>
int main() {
   const int num = 5;

   for (int i = 1; i <= num; i++) {
      printf("Iteration %d\n", i);
   }
   // num = 10;
   return 0;
}
```

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;
}
```

Array

```
#include <stdio.h>
#define limit 10
int main() {
   int grades[limit];
   float sum=0,avg=0;
   printf("enter 10 grades\n");
   for(int i=0;i<limit;i++){
      printf("enter grade %d\n",i+1);
```

```
    scanf("%d",&grades[i]);
    sum=sum+grades[i];

  }
  printf("sum is %f\n",sum);
  printf("average is %f",sum/limit);
}
```

1.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>
void main(){
  int arr[5],rev[5];
  int n;
  printf("enter the elements\n");
  for(int i=0;i<5;i++){
    printf("enter %d number:",i);
    scanf("%d",&arr[i]);
    rev[5-i-1]=arr[i];
  }
  printf("original array is:\n");
    for(int i=0;i<5;i++){
    printf("%d",arr[i]);
  }
  printf("\n");
  printf("reversed array is\n");
  for(int i=0;i<5;i++){
    printf("%d",rev[i]);
  }
}
```

---

2. 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.

```
#include <stdio.h>
int main()
{
  int a[5],temp;
  for(int i=0;i<5;i++){
    printf("enter %d th element\n",i);
    scanf("%d",&a[i]);
  }
  for(int i=0;i<5;i++){
    for(int l=0;l<5-1-i;l++){
      if(a[l]>a[l+1]){
        temp=a[l];
        a[l]=a[l+1];
        a[l+1]=temp;
      }
    }
  }
  printf("largest number in array is %d",a[4]);
  return 0;
}
```

---

3. 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 a[5],temp,count1=1,count2=1,count3=1,count4=1,count5=1;
   for(int i=0;i<5;i++){
      printf("enter %d th element\n",i);
      scanf("%d",&a[i]);
   }
   int i=0;
   for(int i=0;i<5;i++){
      if(a[0]==a[i+1]){
         count1++;
      }

   }
    printf("%d have occured %d times\n",a[0],count1);
   for(int i=0;i<5;i++){
      if(a[1]==a[i]){
         if(i==1){
            continue;
         }
         count2++;
      }

   }
     printf("%d have occured %d times\n",a[1],count2);
    for(int i=0;i<5;i++){
      if(a[2]==a[i]){
         if(i==2){
            continue;
         }
         count3++;
      }

   }
    printf("%d have occured %d times\n",a[2],count3);
    for(int i=0;i<5;i++){
      if(a[3]==a[i]){
         if(i==3){
            continue;
         }
         count4++;
      }

   }
    printf("%d have occured %d times\n",a[3],count4);
    for(int i=0;i<5;i++){
      if(a[4]==a[i]){
         if(i==4){
            continue;
         }
```

```
        count5++;
      }

   }
    printf("%d have occured %d times\n",a[4],count5);
   return 0;
}
```

****

```
#include <stdio.h>

int main() {
   int a[5], count[5] = {0};
   int printed[5] = {0}; // Array to track if an element is already printed

   // Input array elements
   for(int i = 0; i < 5; i++) {
      printf("Enter %d th element: ", i);
      scanf("%d", &a[i]);
   }

   // Count occurrences of each element
   for(int i = 0; i < 5; i++) {
      if (printed[i] == 1) {
         continue; // Skip if this element has already been counted
      }
      for(int j = 0; j < 5; j++) {
         if(a[i] == a[j]) {
            count[i]++;
            printed[j] = 1; // Mark element as counted
         }
      }
   }

   // Print unique occurrences
   for(int i = 0; i < 5; i++) {
      if (count[i] > 0) { // Only print if the element was counted
         printf("%d has occurred %d times\n", a[i], count[i]);
      }
   }

   return 0;
}
```

****

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

Requirements

•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 12 columns

The array should have 5 rows and 12 columns rainfall amounts can be floating point numbers

YEAR RAINFALL (inches)

2010 32.4
2011 37.9
2012 49.8
2013 44.0
2014 32.9

The yearly average is 39.4 inches.

MONTHLY AVERAGES:

Jan Feb Mar Apr May Jun Jul Aug Sep Oct Nov Dec 7.3 7.3 4.9 3.0 2.3 0.6 1.2 0.3 0.5 1.7 3.6 6.7

```
**********************************************************************************************
#include<stdio.h>
void main(){
  float rain_data[5][12]={
    {3.4, 4.2, 5.1, 2.5, 3.9, 4.4, 5.6, 3.1, 2.8, 3.7, 4.1, 3.5},
    {3.1, 3.5, 4.8, 3.2, 4.0, 3.8, 5.2, 4.1, 3.3, 3.9, 3.8, 4.2},
    {2.9, 4.1, 5.0, 2.8, 3.7, 4.5, 5.3, 4.0, 2.7, 3.6, 4.0, 3.9},
    {3.3, 4.0, 5.4, 3.3, 3.8, 4.3, 5.1, 3.9, 3.2, 4.0, 4.2, 3.6},
    {3.5, 4.3, 5.2, 2.9, 4.1, 4.2, 5.0, 3.5, 3.0, 3.8, 4.4, 3.7}
  };
  float total_rain=0,year_avg_rain[5],month_sum_rain[12],year_total_rain[5]={0,0,0,0,0},sum=0;
  char months[12][4]={"Jan","Feb","Mar","Apr","May","Jun","Jul","Aug","Sep","Oct","Nov","Dec"};
  for(int i=0;i<5;i++){
    for(int j=0;j<12;j++){
      year_total_rain[i]=year_total_rain[i]+rain_data[i][j];
      month_sum_rain[j]=rain_data[i][j];
    }

  }
  for(int i=0;i<5;i++){
    year_avg_rain[i]=year_total_rain[i]/12;
  }


  printf("total rainfall for each year\n");
  for(int i=0;i<5;i++){
    printf("202%d\t:%f\n",i,year_total_rain[i]);
  }
  printf("avg rainfall for each year\n");
  for(int i=0;i<5;i++){
    printf("202%d\t:%f\n",i,year_avg_rain[i]);
  }
  printf("avg monthly rainfall for 5 yrs\n");
  for(int i=0;i<12;i++){

      printf("%s\t:%f\n",months[i],month_sum_rain[i]/5);

  }


}
```