* ***Purpose*** **: Classwork.**

* ***Date*** **: 24/12/2025**

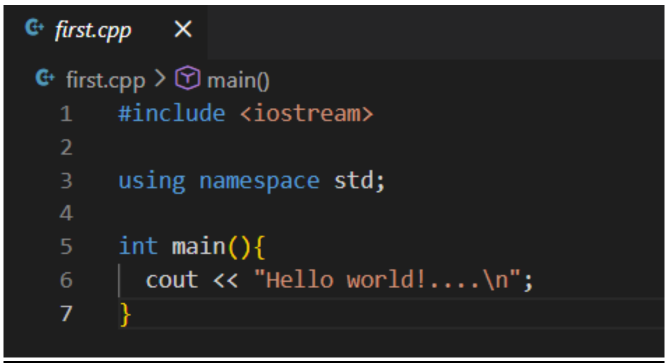
* ***Author*** **: Vikas Srivastava**

* ***ID*** **: 55984**

* ***Batch ID*** **: 25SUB4505**

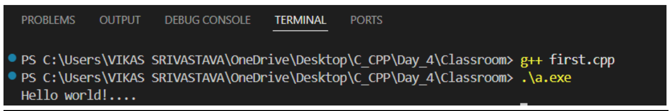
----

1. ***Code :*** Write a program in C++ to display the message "Hello world!" on the screen using the cout output statement.



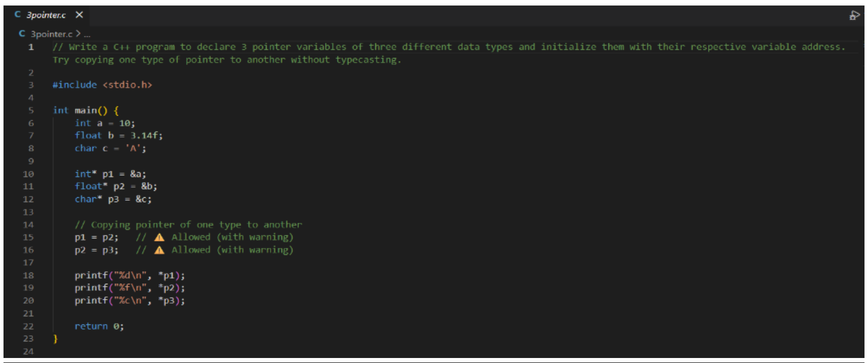
```cpp
#include <iostream>

using namespace std;

int main(){
    cout << "Hello world!....\n";
}
```

Output :



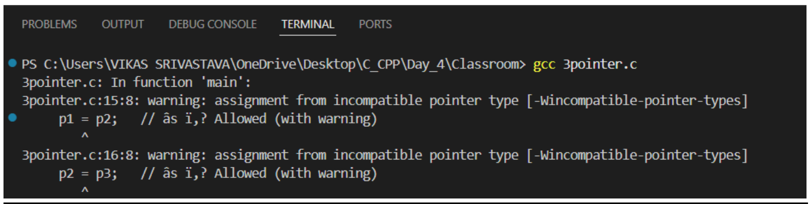
```
PS C:\Users\VIKAS SRIVASTAVA\OneDrive\Desktop\C_CPP\Day_4\Classroom> g++ first.cpp
PS C:\Users\VIKAS SRIVASTAVA\OneDrive\Desktop\C_CPP\Day_4\Classroom> .\a.exe
Hello world!....
```

2. ***Code :*** Write a program to demonstrate pointer type mismatch, where pointers of different data types are assigned to each other, leading to compiler warnings and undefined behavior when dereferencing such pointers.



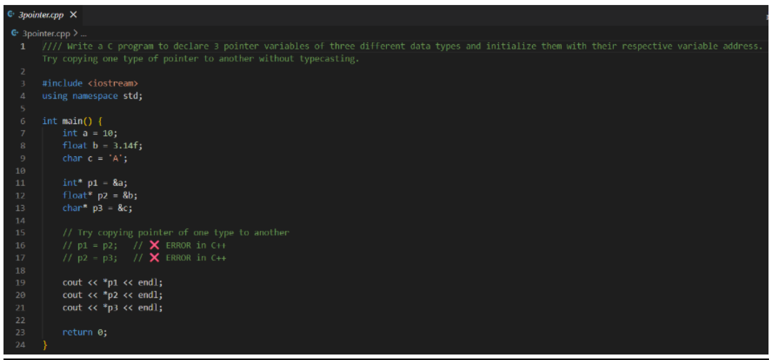
```c
// Write a C++ program to declare 3 pointer variables of three different data types and initialize them with their respective variable address.
Try copying one type of pointer to another without typecasting.

#include <stdio.h>

int main() {
    int a = 10;
    float b = 3.14f;
    char c = 'A';

    int* p1 = &a;
    float* p2 = &b;
    char* p3 = &c;

    // Copying pointer of one type to another
    p1 = p2;   // ⚠ Allowed (with warning)
    p2 = p3;   // ⚠ Allowed (with warning)

    printf("%d\n", *p1);
    printf("%f\n", *p2);
    printf("%c\n", *p3);

    return 0;
}
```

Output :



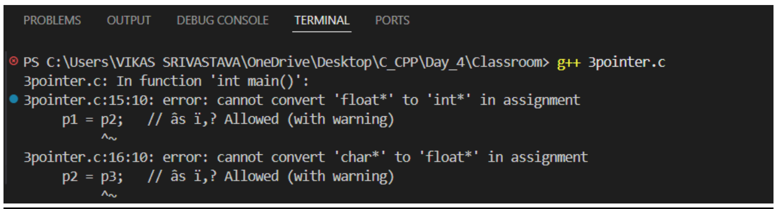
```
PS C:\Users\VIKAS SRIVASTAVA\OneDrive\Desktop\C_CPP\Day_4\Classroom> gcc 3pointer.c
3pointer.c: In function 'main':
3pointer.c:15:8: warning: assignment from incompatible pointer type [-Wincompatible-pointer-types]
     p1 = p2;   // âs ï,? Allowed (with warning)
        ^
3pointer.c:16:8: warning: assignment from incompatible pointer type [-Wincompatible-pointer-types]
     p2 = p3;   // âs ï,? Allowed (with warning)
        ^
```

3. ***Code :*** Write a program to demonstrate strict pointer type safety in C++, where assigning pointers of different data types is not allowed, preventing unsafe memory access and ensuring type correctness at compile time.



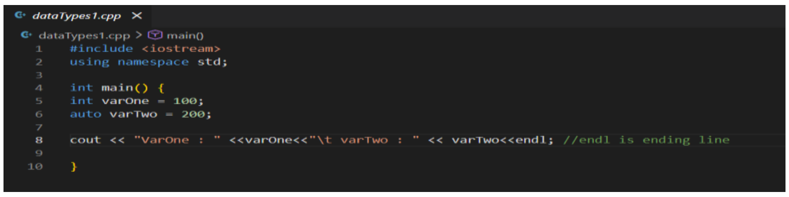
```cpp
//// Write a C program to declare 3 pointer variables of three different data types and initialize them with their respective variable address.
Try copying one type of pointer to another without typecasting.

#include <iostream>
using namespace std;

int main() {
    int a = 10;
    float b = 3.14f;
    char c = 'A';

    int* p1 = &a;
    float* p2 = &b;
    char* p3 = &c;

    // Try copying pointer of one type to another
    // p1 = p2;   // ❌ ERROR in C++
    // p2 = p3;   // ❌ ERROR in C++

    cout << *p1 << endl;
    cout << *p2 << endl;
    cout << *p3 << endl;

    return 0;
}
```

Output :



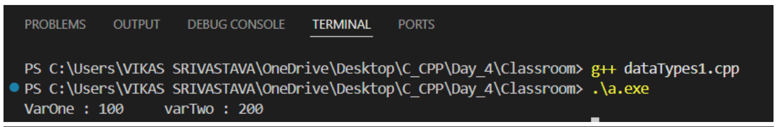
```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS

PS C:\Users\VIKAS SRIVASTAVA\OneDrive\Desktop\C_CPP\Day_4\Classroom> g++ 3pointer.c
3pointer.c: In function 'int main()':
3pointer.c:15:10: error: cannot convert 'float*' to 'int*' in assignment
     p1 = p2;    // âs ï,? Allowed (with warning)
          ^~
3pointer.c:16:10: error: cannot convert 'char*' to 'float*' in assignment
     p2 = p3;    // âs ï,? Allowed (with warning)
          ^~
```

4. ***Code :*** Write a program to demonstrate the use of the auto keyword in C++, where the compiler automatically deduces the data type of a variable at compile time based on the assigned value.



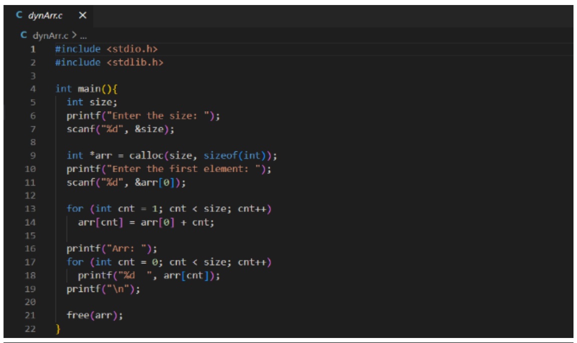
```cpp
#include <iostream>
using namespace std;

int main() {
int varOne = 100;
auto varTwo = 200;

cout << "VarOne : " <<varOne<<"\t varTwo : " << varTwo<<endl; //endl is ending line

}
```

Output :



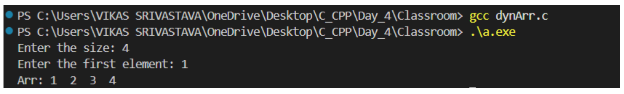
```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS

PS C:\Users\VIKAS SRIVASTAVA\OneDrive\Desktop\C_CPP\Day_4\Classroom> g++ dataTypes1.cpp
PS C:\Users\VIKAS SRIVASTAVA\OneDrive\Desktop\C_CPP\Day_4\Classroom> .\a.exe
VarOne : 100      varTwo : 200
```

5. ***Code :*** Write a program to dynamically allocate memory for an integer array using calloc(), initialize the array with consecutive values starting from a user-given number, display the array elements, and finally free the allocated memory.



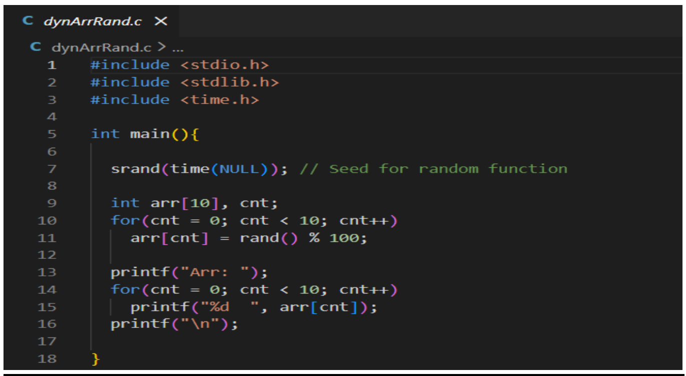
```c
#include <stdio.h>
#include <stdlib.h>

int main(){
  int size;
  printf("Enter the size: ");
  scanf("%d", &size);

  int *arr = calloc(size, sizeof(int));
  printf("Enter the first element: ");
  scanf("%d", &arr[0]);

  for (int cnt = 1; cnt < size; cnt++)
    arr[cnt] = arr[0] + cnt;

  printf("Arr: ");
  for (int cnt = 0; cnt < size; cnt++)
    printf("%d  ", arr[cnt]);
  printf("\n");

  free(arr);
}
```

Output :



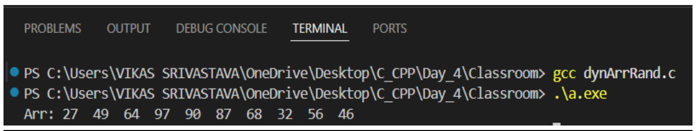
```
PS C:\Users\VIKAS SRIVASTAVA\OneDrive\Desktop\C_CPP\Day_4\Classroom> gcc dynArr.c
PS C:\Users\VIKAS SRIVASTAVA\OneDrive\Desktop\C_CPP\Day_4\Classroom> .\a.exe
Enter the size: 4
Enter the first element: 1
Arr: 1  2  3  4
```

6. ***Code :*** Write a program to generate random numbers using the rand() function, store them in an array, and display the array elements after initializing the random seed using srand(time(NULL)).



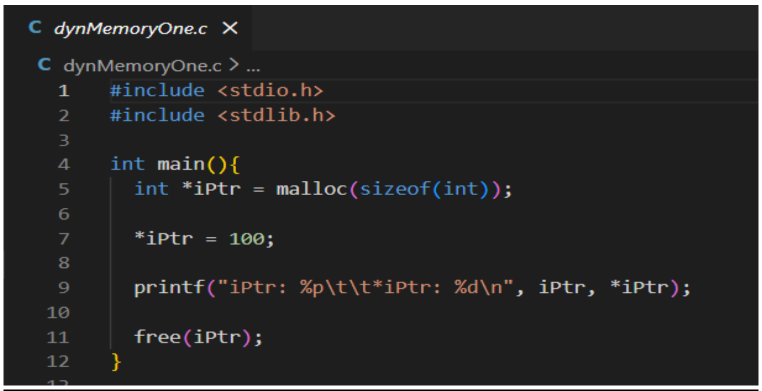
```c
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

int main(){

  srand(time(NULL)); // Seed for random function

  int arr[10], cnt;
  for(cnt = 0; cnt < 10; cnt++)
    arr[cnt] = rand() % 100;

  printf("Arr: ");
  for(cnt = 0; cnt < 10; cnt++)
    printf("%d  ", arr[cnt]);
  printf("\n");

}
```

Output :



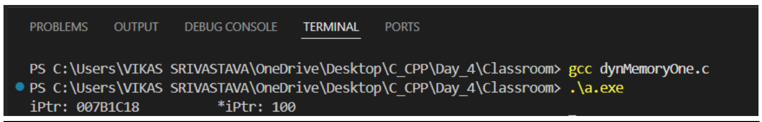
```
PS C:\Users\VIKAS SRIVASTAVA\OneDrive\Desktop\C_CPP\Day_4\Classroom> gcc dynArrRand.c
PS C:\Users\VIKAS SRIVASTAVA\OneDrive\Desktop\C_CPP\Day_4\Classroom> .\a.exe
Arr: 27  49  64  97  90  87  68  32  56  46
```

7. ***Code :*** Write a program to demonstrate dynamic memory allocation using malloc(), store a value in the allocated memory through a pointer, display the memory address and stored value, and finally free the allocated memory.



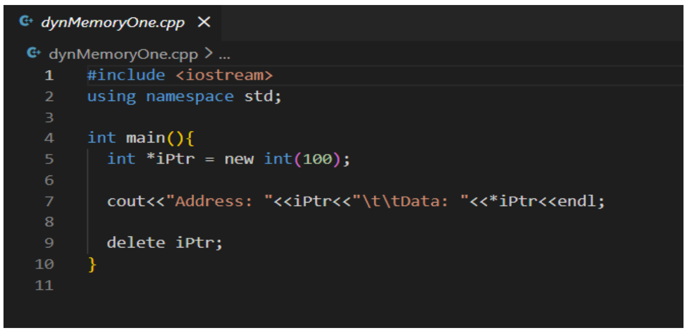
```c
#include <stdio.h>
#include <stdlib.h>

int main(){
  int *iPtr = malloc(sizeof(int));

  *iPtr = 100;

  printf("iPtr: %p\t\t*iPtr: %d\n", iPtr, *iPtr);

  free(iPtr);
}
```

Output :



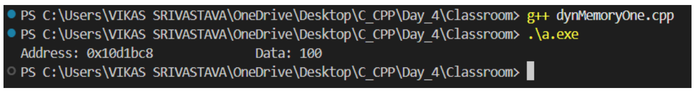
```
PS C:\Users\VIKAS SRIVASTAVA\OneDrive\Desktop\C_CPP\Day_4\Classroom> gcc dynMemoryOne.c
PS C:\Users\VIKAS SRIVASTAVA\OneDrive\Desktop\C_CPP\Day_4\Classroom> .\a.exe
iPtr: 007B1C18          *iPtr: 100
```

8. ***Code*** **:** Write a program to demonstrate dynamic memory allocation in C++ using the new operator to allocate memory for an integer, access the stored value using a pointer, and properly release the memory using the delete operator.



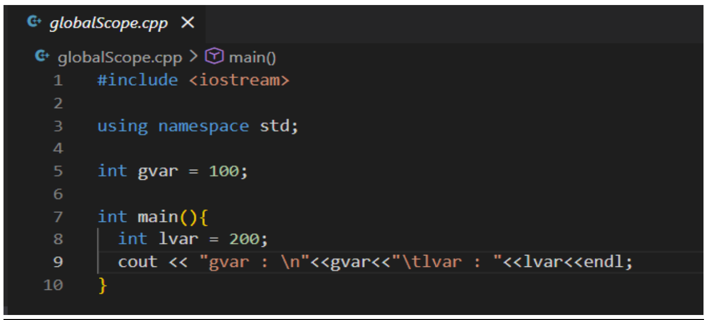
```cpp
#include <iostream>
using namespace std;

int main(){
    int *iPtr = new int(100);

    cout<<"Address: "<<iPtr<<"\t\tData: "<<*iPtr<<endl;

    delete iPtr;
}

```

Output :



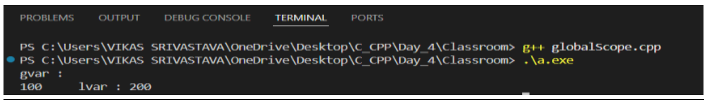
```
PS C:\Users\VIKAS SRIVASTAVA\OneDrive\Desktop\C_CPP\Day_4\Classroom> g++ dynMemoryOne.cpp
PS C:\Users\VIKAS SRIVASTAVA\OneDrive\Desktop\C_CPP\Day_4\Classroom> .\a.exe
Address: 0x10d1bc8              Data: 100
PS C:\Users\VIKAS SRIVASTAVA\OneDrive\Desktop\C_CPP\Day_4\Classroom>
```

9. ***Code :*** Write a program to demonstrate the difference between a global variable and a local variable in C++, showing their scope and accessibility within the program.



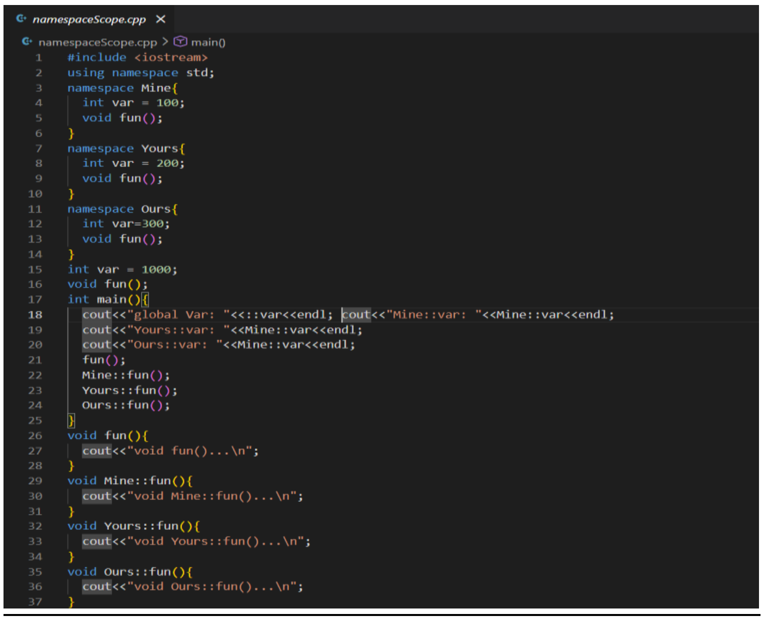
```cpp
#include <iostream>

using namespace std;

int gvar = 100;

int main(){
    int lvar = 200;
    cout << "gvar : \n"<<gvar<<"\tlvar : "<<lvar<<endl;
}
```

Output :



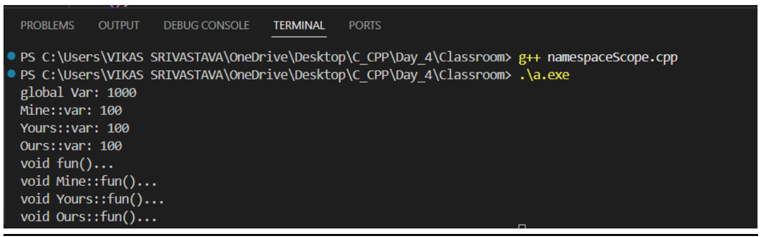
```
PS C:\Users\VIKAS SRIVASTAVA\OneDrive\Desktop\C_CPP\Day_4\Classroom> g++ globalScope.cpp
PS C:\Users\VIKAS SRIVASTAVA\OneDrive\Desktop\C_CPP\Day_4\Classroom> .\a.exe
gvar :
100     lvar : 200
```

10. Code : Write a program to demonstrate the use of multiple namespaces in C++, showing how variables and functions with the same names can coexist and be accessed using the scope resolution operator (::), including global scope and namespace-specific scope.



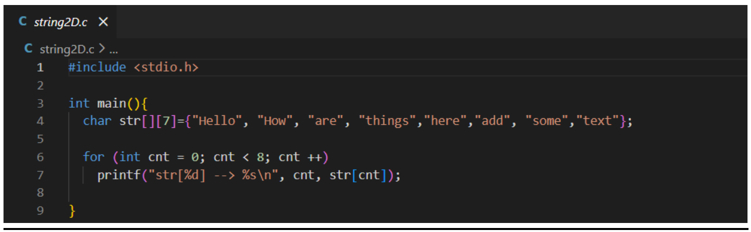
```cpp
#include <iostream>
using namespace std;
namespace Mine{
    int var = 100;
    void fun();
}
namespace Yours{
    int var = 200;
    void fun();
}
namespace Ours{
    int var=300;
    void fun();
}
int var = 1000;
void fun();
int main(){
    cout<<"global Var: "<<::var<<endl; cout<<"Mine::var: "<<Mine::var<<endl;
    cout<<"Yours::var: "<<Mine::var<<endl;
    cout<<"Ours::var: "<<Mine::var<<endl;
    fun();
    Mine::fun();
    Yours::fun();
    Ours::fun();
}
void fun(){
    cout<<"void fun()...\n";
}
void Mine::fun(){
    cout<<"void Mine::fun()...\n";
}
void Yours::fun(){
    cout<<"void Yours::fun()...\n";
}
void Ours::fun(){
    cout<<"void Ours::fun()...\n";
}
```

Output :



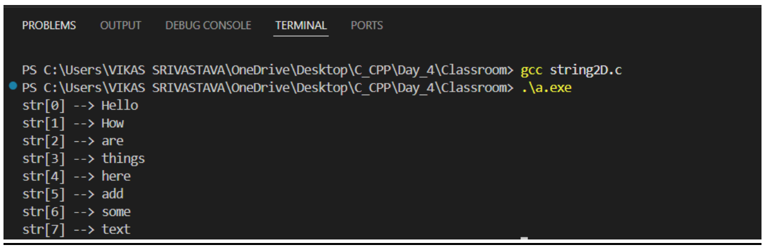
```
PS C:\Users\VIKAS SRIVASTAVA\OneDrive\Desktop\C_CPP\Day_4\Classroom> g++ namespaceScope.cpp
PS C:\Users\VIKAS SRIVASTAVA\OneDrive\Desktop\C_CPP\Day_4\Classroom> .\a.exe
global Var: 1000
Mine::var: 100
Yours::var: 100
Ours::var: 100
void fun()...
void Mine::fun()...
void Yours::fun()...
void Ours::fun()...
```

11. ***Code :*** Write a program to demonstrate how multiple strings can be stored in a two-dimensional character array and printed individually using a loop.



```c
#include <stdio.h>

int main(){
    char str[][7]={"Hello", "How", "are", "things","here","add", "some","text"};

    for (int cnt = 0; cnt < 8; cnt ++)
        printf("str[%d] --> %s\n", cnt, str[cnt]);

}
```

Output :



```
PS C:\Users\VIKAS SRIVASTAVA\OneDrive\Desktop\C_CPP\Day_4\Classroom> gcc string2D.c
PS C:\Users\VIKAS SRIVASTAVA\OneDrive\Desktop\C_CPP\Day_4\Classroom> .\a.exe
str[0] --> Hello
str[1] --> How
str[2] --> are
str[3] --> things
str[4] --> here
str[5] --> add
str[6] --> some
str[7] --> text
```