Task 15: Using the auto Keyword in C++

Objective:
To understand and practice the use of the auto keyword in C++. This task demonstrates how auto simplifies variable declarations by allowing the compiler to automatically deduce the data type from the initializer.

1. Concept Overview

The auto keyword was introduced in C++11. It instructs the compiler to automatically infer the type of a variable based on the value assigned to it. This improves readability, reduces verbosity, and minimizes type-related errors.

2. Program Code

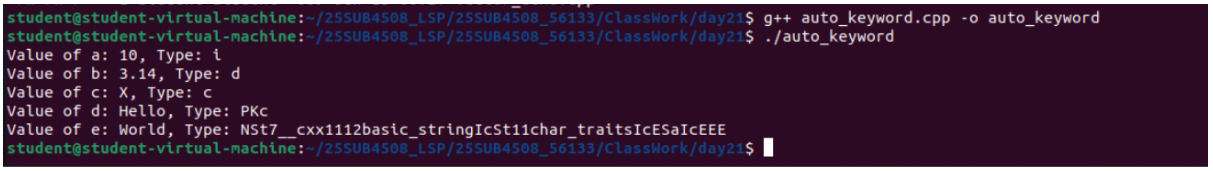
```
#include <iostream>
 #include <typeinfo>
 #include <string>
 using namespace std;

 int main() {
        auto a = 10;            // int
        auto b = 3.14;          // double
        auto c = 'X';           // char
        auto d = "Hello";       // const char*
        auto e = string("World"); // std::string

        cout << "Value of a: " << a << ", Type: " << typeid(a).name() << endl;
        cout << "Value of b: " << b << ", Type: " << typeid(b).name() << endl;
        cout << "Value of c: " << c << ", Type: " << typeid(c).name() << endl;
        cout << "Value of d: " << d << ", Type: " << typeid(d).name() << endl;
        cout << "Value of e: " << e << ", Type: " << typeid(e).name() << endl;

        return 0;
}
```

3. Compilation Instructions

Compile the program using a C++ compiler:

```
 g++ auto_keyword.cpp -o auto_keyword
```

4. Sample Output



5. Observations & Explanation

1. The compiler automatically deduces the data type of each variable.
2. auto a is inferred as int, auto b as double, and auto c as char.
3. String literals are inferred as const char*.
4. std::string objects are correctly inferred as string.
5. typeid().name() shows compiler-specific type representations.

6. Advantages of auto Keyword

- Reduces code verbosity
- Improves readability
- Avoids type mismatches
- Useful with templates and iterators

7. Conclusion

This task demonstrates how the auto keyword simplifies variable declarations and improves code clarity. auto is a powerful feature of modern C++ and is widely used in STL, templates, and large-scale applications.