Switch case in c++:

The switch statement in c++ is a control flow statement that allows you to execute different blocks of code based on the value of an expression. It is an alternative to using a series of if-else statements when you need to compare a variable against multiple values.

Syntax of switch Statement in C++

```
switch (expression) {
   case value_1:
      // statements_1;
      break;
   case value_2:
      // statements_2;
      break;
   .....
   .....
   default:
      // default_statements;
      break;
}
```

Program:

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

int main(){
   char grade;
   cout<<"Enter your grade : ";
   cin>>grade;

   // if(grade=='A'){
   //     cout<<"your marks will be in range of 90 to 100"<<endl;
   // }
   // else if(grade == 'B'){
   //     cout<<"your marks will be in range of 80 to 90"<<endl;
   // }
   // else if(grade == 'C'){
   //     cout<<"your marks will be in range of 70 to 80"<<endl;
   // }
   // else if(grade == 'D'){
   //     cout<<"your marks will be in range of 60 to 70"<<endl;
   // }
   // else{
   //     cout<<"your marks will be in range of 0 to 60"<<endl;
   // }

   switch(grade){
     case 'A': cout<<"your marks will be in range of 90 to 100"<<endl;
               break;
     case 'B': cout<<"your marks will be in range of 80 to 90"<<endl;
               break;
     case 'C': cout<<"your marks will be in range of 70 to 80"<<endl;
               break;
     case 'D': cout<<"your marks will be in range of 60 to 70"<<endl;
               break;
    default: cout<<"your marks will be in range of 0 to 60"<<endl;

   }
   return 0;

}
```

Output:

```
PS E:\C++ PROGRAMMES> g++ switchcase.cpp
PS E:\C++ PROGRAMMES> ./a.exe
```

```
Enter your grade : B

your marks will be in range of 80 to 90

PS E:\C++ PROGRAMMES> ./a.exe

Enter your grade : M

your marks will be in range of 0 to 60
```

Conditions pertaining to 'switch':

Expression Type: the expression should be int or char or enum. It does not accept the float and string as the expression.

Unique case value: The value of each case should be unique. We cannot have single values to multiple cases.

No range checking (in case): The case value cannot be an expression. The value should be constant value.

Fall through behaviour: The break keyword is used in the switch case to break out of the switch when encountered. It is used at the end of every case block so that when the matching case is executed, the program control comes out of the loop.

The break statement is optional. If omitted, all the cases after the matching case will also be executed.

Execution Order: The order of execution is top to bottom case one by one if the value is does not match. It is sequentially executed.

For More Reference: https://www.geeksforgeeks.org/switch-statement-in-cpp/

Home Work:

1. **What we can not use in the case value:**

   1. Non-Integral Data Types:

   Floating-point numbers (float, double): Due to potential precision issues and the way they are represented in memory, comparing them exactly can be problematic.

   Strings: C++ does not have built-in string comparison for switch cases. However, you can use a workaround like converting strings to integers using a hash function, but it's generally not recommended for readability and maintainability.

2. Non-Constant Expressions:

Variables: Their values can change during execution, making it impossible to determine the matching case at compile time.
Function calls: Their return values are not known until runtime.
Arithmetic expressions: They involve variables or calculations that aren't constant.

3. User-Defined Types (Except Enums):

Structures, classes, unions: C++ doesn't allow comparing them directly for equality in a switch statement.

4. Duplicate Case Values:

Each case value must be unique within a switch statement.

Allowed Case Values:

Integer types (int, char, short, long, etc.): They can be directly used as case values.
Enumerated types (enums): Their values are essentially named integers, making them suitable for switch cases.

Key Points:

The compiler needs to know the case values at compile time to generate efficient code.
Enums provide a way to create named constants for readable switch cases.
If you need to handle non-integral types or more complex conditions, consider using if-else statements instead of switch-case.