GSLC – PLC1

Name : Vincensius Adriel
NIM : 2101642301
Class number : 44
Class : LH-01

1. What are the advantages and disadvantages of decimal data types?
2. How does a decimal value waste memory space?
3. In what way is static type checking better than dynamic type checking?
4. What is coercion in programming languages?
5. Explain how coercion rules can weaken the beneficial effect of strong typing?

Answer :

1. What are the advantages and disadvantages of decimal data types?

Decimal data types store a fixed number of decimal digits, with the decimal point at a fixed position in the value. Decimal types have the advantage of being able to precisely store decimal values, at least those within a restricted range, which cannot be done with floating-point.

The disadvantages of decimal types are that the range of values is restricted because no exponents are allowed

For example, A 16-bit integer would only allow 65536 different numbers. If you use one bit as indication of positive or negative, you would only be able to have 32768 positive and negative numbers. (Actually, you would end up with a positive zero and negative zero too, which are both equal, so that's fixed by subtracting one from all negative values. The range then becomes all from -32768 to +32767. (Because you also have to count the 0.), and their representation in memory is mildly wasteful.

2. How does a decimal value waste memory space?

Decimal types are stored very much like character strings, using binary codes for the decimal digits. These representations are called binary coded decimal (BCD). In some cases, they are stored one digit per byte, but in others, they are packed two digits per byte. Either way, they take more storage than binary representations. It takes at least four bits to code a decimal digit.
Therefore, to store a six-digit coded decimal number requires 24 bits of memory.
However, it takes only 20 bits to store the same number in binary.

Decimal values store numeric information as digits encoded using the four bit binary equivalents: 0 (0000) to 9 (1001). That means a single byte can hold values between 0 and 99. But simply using the same byte to hold a binary value will yield values between 0 and 255 (or –128 and +127).

3. In what way is static type checking better than dynamic type checking?

The big benefit of static type checking is that it allows many type errors to be caught early in the development cycle. Static typing usually results in compiled code that executes more quickly because when the compiler knows the exact data types that are in use, it can produce optimized machine code (i.e. faster and/or using less memory). Static type checkers evaluate only the type information that can be determined at compile time, but are able to verify that the checked conditions hold for all possible executions of the program, which eliminates the need to repeat type checks every time the program is executed.

A static type-checker will quickly detect type errors in rarely used code paths. Without static type checking, even code coverage tests with 100% coverage may be unable to find such type errors. However, a detriment to this is that static type-checkers make it nearly impossible to manually raise a type error in your code because even if that code block hardly gets called – the type-checker would almost always find a situation to raise that type error and thus would prevent you from executing your program (because a type error was raised).

4. What is coercion in programming languages?

Many programming languages support the conversion of a value into another of a different data type. This kind of type conversions can be implicitly or explicitly made. Implicit conversion, which is also called coercion, is automatically done. Explicit conversion, which is also called casting, is performed by code instructions. This code treats a variable of one data type as if it belongs to a different data type. The languages that support implicit conversion define the rules that will be automatically applied when primitive compatible values are involved.

5. Explain how coercion rules can weaken the beneficial effect of strong typing?

Coercion is implicit type conversions. So if we implicitly change the types of somevariables/values then we are subverting the typing rules of the language.

Compilers for strong typing languages usually can detect more typing related errors during the compile time. Compared to that, coercion rules
make the language less compile time reliable. In addition, it is considered easier for people to understand codes written in strong typing languages since coercion rules usually adds some logic implicitly behind the code itself.