Lecture 1 Introduce

#CS106L Design Philosophy of C++:

• Allow the programmer full control, responsibility, and choice if they want it.

• Express ideas and intent directly in code.

• Enforce safety at compile time whenever possible.

• Do not waste time or space.

• Compartmentalize messy constructs

Q1:formatted data?

In C++, and generally in computer science, formatted data refers to data that is organized and structured in a specific way. Formatted data is typically easier to read and understand than raw data, and it can be more easily processed by programs and algorithms.

There are many different ways to format data, depending on the type of data and the context in which it is used. For example,

- Text data can be formatted using a specific character encoding, such as UTF-8 or ASCII, and it can be structured using line breaks and other delimiters.
- Numeric data can be formatted using specific number formats, such as floating-point or fixed-point, and it can be arranged into columns and rows to form tables or spreadsheets.
- Dates and times can be formatted using specific date and time formats, such as ISO-8601 or RFC-2822, and it can be represented in different time zones or calendars.
- Formatted data can be used for Human-readable data like CSV(Comma Separated Values), JSON and XML which uses a specific format to represent structured data in a way that can be easily read and understood by both humans and machines.

In C++, the `stringstream` class and the `printf` family functions are examples of how data can be formatted. The `stringstream` class allows data to be formatted into a string, while the `printf` family functions allows data to be formatted into a string that can be output to the console or written to a file.

In general, formatted data is crucial for any kind of data analysis, data storage, and data transmission, it makes it easy to represent data in a standardized way and process it in an efficient and consistent way.

Q2:pipeline?

A pipeline in computer science is a sequence of processing stages in which the output of one stage becomes the input of the next stage. Each stage in a pipeline performs a specific task and the pipeline as a whole performs a more complex task by combining the individual tasks of each stage.

Pipelines are often used in parallel computing, where each stage is executed concurrently on different data, and in streaming data processing, where data flows through the pipeline in real-time. They are useful for breaking down a large and complex problem into smaller, more manageable tasks and for improving the performance of a system by allowing multiple tasks to be executed simultaneously.

An example of pipeline in computer science could be image processing pipeline where image data goes through stages of reading image ,resizing, cropping, apply filter, feature extraction ,compression and finally storing it. Another example could be compiling process, where source code is passed through multiple stages like tokenizing, parsing, code generation, optimization, and finally, assembly or machine code.

Q3:state bit?

In C++, a stream's state bits are flags that indicate the status of a stream. There are several different state bits that can be set by various operations on a stream, such as reading from or writing to a stream.

The main state bits used in C++ are:

- `goodbit`: This is the normal state bit and it is set when there are no errors or other exceptional conditions.
- `eofbit`: This bit is set when a stream reaches the end-of-file (EOF) condition, typically when reading from a file or other input stream.

- `failbit`: This bit is set when an operation on the stream fails, such as reading an invalid data type or writing to a read-only stream.
- `badbit`: This bit is set when a serious error occurs, such as when the stream is corrupted or the device it is associated with becomes unavailable.

Each of these state bits can be checked independently by calling the corresponding method of the stream, such as `eof()`, `fail()` or `bad()`. These methods return a boolean value indicating whether the state bit is set or not.

You can also check the general state of a stream by calling the `good()` method, which will return `false` if any of the bits `eofbit`, `failbit`, or `badbit` are set.

```
if (cin.good()) {

    // no error

} else {

    if (cin.eof()) {

        // reached EOF

    } else if (cin.fail()) {

        // input error

    } else {

        // some other error

    }

}
```

It should be noted that the stream state bits are sticky and once they are set, they will remain set until the stream is cleared using the `clear()` method.

Also, some operations may set more than one state bit for example, the `>>` operator will set the `eofbit` if it reaches the end of the stream, and it will set the `failbit` if it encounters an error.