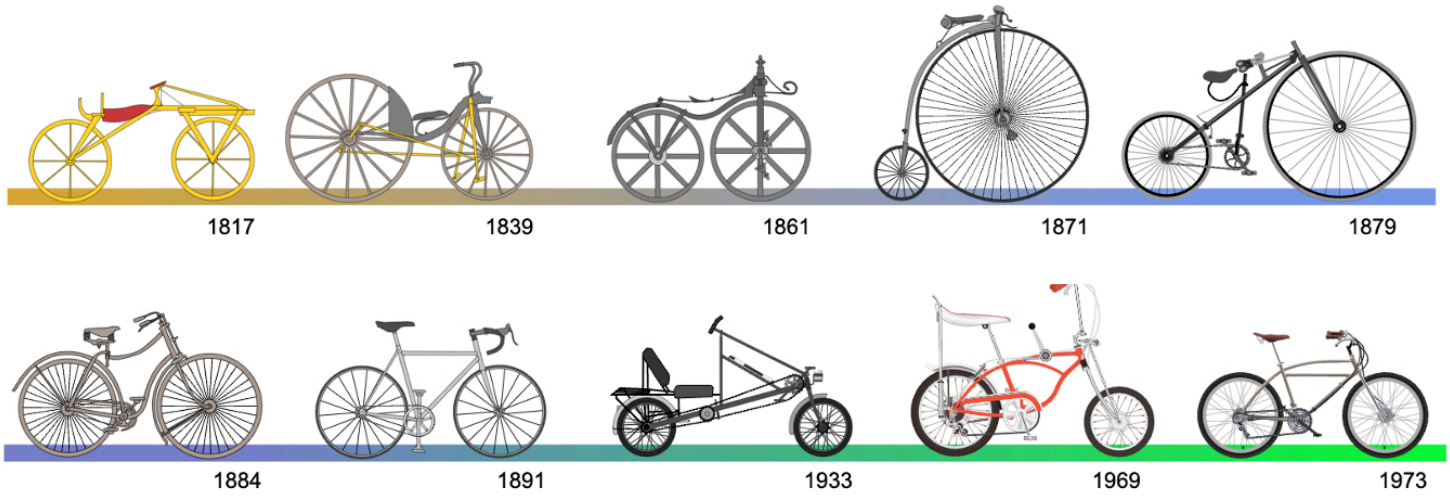

Homework 1 -- Evolution of C++

CS111 & EIE111 -- C++ Programming 2024 Spring
Designer of this project: Zhiyao Liang
zyliang@must.edu.mo
March. 06, 2024

The above picture, found on the Internet [1], shows the bicycle design evolving based on reasonable ideas. Some practical or reasonable ideas should also drive the migration from C to C++. This project is designed to explore the ideas of C++'s evolution.

I. Overview

C++ is designed to be more convenient than C, especially for programming scenarios involving abstraction. Here, the word "abstraction" relates to other jargon, such as Abstract Data Type (ADT), interface, encapsulation, data hiding, etc.

This project is based on possible customer requests to use music player devices. Such a music player should satisfy the following conditions:

- The device stores songs, while each song's information includes
  - title: name of the sone
  - authors: who wrote the song
  - actors: who performed the song
  - year: when was it published
  - media: the music content.
- Each song has a different id in the device to distinguish it from other songs.

- A song can be added to the device.
- A song can be deleted from the device.
- A song whose title contains certain words (as a substring) can be found.
- A song with a specific ID number can be found
- All the songs in the device can be played together individually.
- The memory(storage) for the device can be cloned or replaced by some backup clone.
- The storage of the device can be emptied.
- The number of songs on the devices can be known.
- A selected song can be copied (cloned)
- A selected song can be played

A music player's interface exposes the above functions to a customer. However, quite some details of the device should be hidden from a customer because customers commonly do not care about technical details like the digital format of the media of a song or the memory structure of the device.

In this project, we will write three different versions of programs using C and C++ to experience the advantages of C++ over C.

II Preparation

II.1 Prepare the coding software tools

Be sure that some recommended compilers for C and C++ are installed on your computer and can be used at the command line. For more on the recommended compilers, see Appendix A. 2.

- Be sure that a tool for using makefile is available. See Appendix A.3 for how to install and use such a tool.

II.2 Study the provided code.

A file `code.zip` is provided. After unzipping it, its folder contains the following content:

- The `Compile_and_run` folder contains the `makefile` and running records (screen records of running executable files) for Windows or Mac.
  The `Utility` folder contains the code for generally helpful tools, not just for the Music Player program. It includes two groups of files.
  - `util.h` and `util.c` define some general tools, including the definition of a struct `Bytes` describing a sequence of bytes. `test_util.c` is the testing file.

    - `util2.h` and `util2.cpp` implement a `Bytes` class for a similar purpose. `test_util2.cpp` is the testing file.
- The folder `SongPlayer_v1` contains a C program specifying the interface using a common C style.
- The folder `SongPlayer_v2` contains a C program that specifies the interface using a class-like style.
- The folder `SongPlayer_v3` contains a C++ program that specify the interface using the C++ way.

III Tasks

- Download `code.zip` and unzip it into some folder containing the provided program files.
- There are 74 missing code parts, clearly marked as the 74 tasks. Do the tasks of providing the missing code. These tasks should be done following the task numbers, from small to large. More specifically, the tasks should be done in five sequential stages. Each stage should do the tasks in some different files, compile the files to generate the corresponding executable files, and do the debugging and testing. The following table lists each stage's program files and executable file names.

stage number	code files	executable file (.exe or .out)
1	util.c	test_util
2	song_player_v1.c	test_v1
3	song_player_v2.h, song_player_v2.c	test_v2
4	util2.h, util2.cpp	test_util2
5	song_player_v3.h, song_player_v3.cpp, test_song_player_v3.cpp	test_v3

- Write the report file `pjt1_report.docx` .
- Fill the Excel file `pjt1_self_grading.xlsx` .
- Write the answers for the questions in the file `pjt1_QA.docx`

IV. Submission

- At most, three students can form a group to submit the homework together. Group members can share code and discuss the assignment sufficiently. But sharing between groups is not allowed. Each group should do the work independently.

    - Only one member of the group should submit the homework files. Ensure the group members' names and class info (EIE/CS D1/D2/D3) are mentioned in the report file.
    - It is perfectly ok to do the homework alone, i.e., a one-person group.
- Upload your files at the webpage address of this homework on Moodle, including:
    - A .zip file made by compressing the whole coding folder. I.e., do all the programming in the folder unzipped from `code.zip` and zip this folder as a .zip file.
    - `pjt1_report.docx` .
    - `pjt1_self_grading.xlsx` .
    - `pjt1_QA.docx`
- Deadline: 11 pm, Saturday, April 6, 2024

Appendix

A.1: Knowledge coverage in this assignment

This project covers practicing a wide range of knowledge items of C and C++. Some knowledge items that may not be familiar to a person who has learned C include:

1. Different ways of describing an interface (for program clients)

- as a group of public functions declared in a `.h` file (C style)
- as a struct which contains function pointers (C style)
- as a class (C++ style).

3. Using C++ library container classes like `string` and `vector` .
4. The special class members

- constructors (default constructor, copy constructor ...)
- the destructor

5. Operator overloading: `<< [] += =`
6. Using namespace.
7. Call C code in a C++ program.
8. Exception handling
9. Range-based for loop
10. Design issues of classes, like deep copying.

A.2: Some recommended compilers

- On Windows:

    - `gcc` for C programs and `g++` for C++ programs. MinGW provides these compilers.
    - Or, `cl` (provided by Visual Studio Community) for C and C++.
- On Mac OS X and Linux
    - `gcc` for C programs and `g++` for C++ programs.

A.3: How to use make and makefile

The `make` program is usually available on Mac OS or Linux. A similar tool recommended for Windows is `mingw32-make`, provided after MinGW is installed. See [6] for more information on installing such a tool on Windows.

A text file named `makefile` (case insensitive) records the needed rules for compiling a program. A rule usually has the form:

```
goal: supporting file names
  a command to generate the target
```

After `make` (or mingw32-make) is installed, we do the following to execute a compiling rule to generate a target
- Step 1: at the command line, change the current folder to the one where the file named "makefile" is located.
- Step 2: use the command:

```
make goal
```

The power of `make` is recursive. When executing a rule to reach or generate a goal, all the dependent files described in the rule need to be available; when one of the supporting files is missing, other rules for generating it will be executed...

For example, for this project, the following commands are possible:

- `make all` : generate all the needed executable files depending on binary (`.o` or `.obj`) files.
- `make util.o` : generate the file `util.o`
- `make test_util.exe` : generate the file test_util.exe, and all the depending on binary files.

References

[1] "File: Bicycle evolution-fr.svg"
https://upload.wikimedia.org/wikipedia/commons/d/d6/Bicycle_evolution-fr.svg

[2] "C++ Primer Plus (Developer's Library)", 6th Edition, Stephen Prata, 2011, Addison-Wesley Professional.

[3] "C Primer Plus, 6th Edition", 6th Edition, Stephen Prata, 2014, Addison-Wesley Professional.

[4] "cppreference.com reference"
https://cppreference.com

[5] "cplusplus.com reference"
https://cplusplus.com/reference/

[6] "How to Install and Use “Make” in Windows"
https://www.technewstoday.com/install-and-use-make-in-windows/