

Part A

Notes:

1. First we create a simple struct and class based on three variables.
2. After that we write values to bin file using io operations.

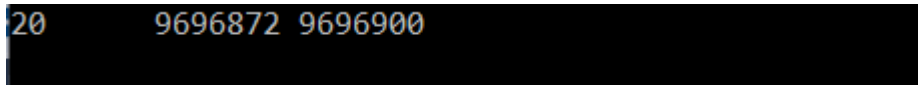
```
void Save(ofstream &of)
{
    of.write((char*)&age, sizeof(age));
    of.write((char*)&first, sizeof(first));
    of.write((char*)&last, sizeof(last));
}
void WriteBinaryFile(string strFile)
{
    ofstream fs;
    fs.open(strFile, ios::out | ios::binary | ios::app);
    if (!fs.is_open())
    {
        cout << "cannot open file" << strFile << endl;
    }
    else
    {
        this->Save(fs);
    }
    fs.close();
}
```

3. After opening a file we need to close it as in a broader context, that is thinking more widely than just C++ and fstream, when an output file is closed, any data remaining in the output buffer is written to the output file. If a file isn't closed properly, then some data may be missing from the file, or the file may be corrupted in some other way. But that's just background information.
4. After compiling and running the program you can see binary file in the project folder.

5. After that we read the data from the binary file and print it on console to check.

```
void ReadBinaryFile(string strFile)
{
    Person p;
    ifstream binaryFile;
    int size = 0;
    binaryFile.open(strFile, ios::in | ios::binary);
    binaryFile.seekg(0, ios::end);
    size = (int)binaryFile.tellg();
    binaryFile.seekg(0, ios::beg);
    while (binaryFile.tellg() < size)
    {
        binaryFile.read((char*)p.age, sizeof(p.age));
        binaryFile.read((char*)p.first, sizeof(p.first));
        binaryFile.read((char*)p.last, sizeof(p.last));
        cout << p.age << "\t" << p.first << "\t" << p.last << endl;
    }
    binaryFile.close();
}
```

Read Binary

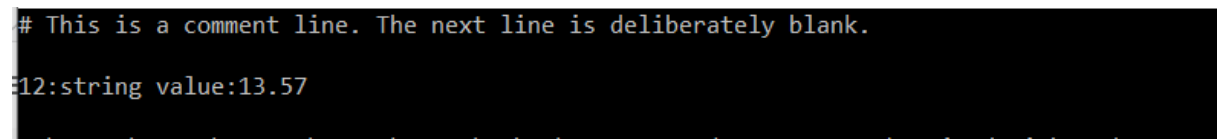


```
20      9696872 9696900
```

Output

Part B

1. For this part we mainly read in a simple text file and start with just simply showing the output lines in the text files.



```
# This is a comment line. The next line is deliberately blank.
12:string value:13.57
```

2. Next we modify the code so that it can now read the file and analyze and avoid comment lines and blank lines and additionally print split line to screen.

```

int main()
{
    ifstream myFileStream("./test2.txt");
    if (!myFileStream.is_open()) {
        cout << "File unable to open" << endl;
        return 0;
    }
    string line;
    while (getline(myFileStream, line)) {
        string str = "";
        if (line.size() < 1)
            continue;
        if (line.at(0)=='#')
        {
            continue;
        }
        for (auto x : line)
        {
            if (x == ':')
            {
                cout << str << endl;
                str = "";
                continue;
            }
            str = str + x;
            if (x == line.at(line.size() - 1))
            {
                cout << str << endl;
            }
        }
    }
}

```

```

12
string value
13.57

```

Part C

1. First we download the nlohmann Json from github and place it in our project directory to make it easier to include in our project.
2. Then we read in the json file and print the values to console.

```
#include "pch.h"
#include <iostream>
#include "../json-develop/single_include/nlohmann/json.hpp"
#include <fstream>

using json = nlohmann::json;
using namespace std;

int main()
{
    std::ifstream ppl("people.json");
    json people = json::parse(ppl);
    cout << people << endl; //This will print the entire json object.
}
```