

Data Structures Chapter 2

1. Recurrence Relations
2. Discrete Math
- 3. Structure**
 - Structure & Array
 - **Structure & Class**
 - **Problem Set – Clock**

Struct and Class

- Believe it or not, only one difference between **struct** and **class** is that by default **struct** members are **public** and **class** members are **private** in C++.
- But as per programming consideration,
 - Use the **struct** keyword for data-only structures.
 - Use the **class** keyword for objects that have both data and functions..

Reference: <https://stackoverflow.com/questions/1127396/struct-constructor-in-c/1127406>

PSet Clock: Demo

- clock3
- clock4
- clock

```
PS C:\GitHub\nowicx\psets\pset04struct-clock> ./clock3
14:38:57
14:38:58
14:38:59
14:39:00
14:39:01
14:39:02
PS C:\GitHub\nowicx\psets\pset04struct-clock> ./clock4
11:59:44
```

```
PS C:\GitHub\nowicx\psets\pset04struct-clock> ./clockx.exe
11:58:57
11:58:58
11:58:59
11:59:00
11:59:01
11:59:02
12:00:13
```

PSet Clock: Step 1

- What is wrong in the code?

ver.1

```
#include <iostream>
#include <iomanip>
struct Clock{
    int hr, min, sec;
};

void tick(Clock *ptr);
void show(Clock *ptr);

int main (void) {
    Clock *clock = {14, 38, 56};

    for(int i = 0; i < 6; ++i) {
        tick(clock);
        show(clock);
    }
    return 0;
}
```

NOTE: Correct the code above **not to use** pointers in main().

ver.1

```
// increment the time by one second.
void tick(Clock *ptr) {
    ptr->sec++;

    // your code here

}

// show the current time in military form.
void show(Clock *ptr) {
    std::cout.fill('0');
    std::cout << std::setw(2) << ptr->hr << ":"
               << std::setw(2) << ptr->min << ":"
               << std::setw(2) << ptr->sec << std::endl;
}
```

PSet Clock: Step 1

- What is wrong in the code?

ver.1

```
#include <iostream>
#include <iomanip>
struct Clock{
    int hr, min, sec;
};

void tick(Clock *ptr);
void show(Clock *ptr);

int main (void) {
    Clock clock = {14, 38, 56};

    for(int i = 0; i < 6; ++i) {
        tick(&clock);
        show(&clock);
    }
    return 0;
}
```

ver.1

```
// increment the time by one second.
void tick(Clock *ptr) {
    ptr->sec++;

    // your code here

}

// show the current time in military form.
void show(Clock *ptr) {
    std::cout.fill('0');
    std::cout << std::setw(2) << ptr->hr << ":"
               << std::setw(2) << ptr->min << ":"
               << std::setw(2) << ptr->sec << std::endl;
}
```

PSet Clock: Step 2

- Rewrite the code using a pointer ***ptr** and new instead of **Clock clock**;

ver.1

```
#include <iostream>
#include <iomanip>
struct Clock{
    int hr, min, sec;
};

void tick(Clock *ptr);
void show(Clock *ptr);

int main (void) {
    Clock clock = {14, 38, 56};

    for(int i = 0; i < 6; ++i) {
        tick(&clock);
        show(&clock);
    }
    return 0;
}
```

ver.2

```
#include <iostream>
#include <iomanip>
struct Clock{
    int hr, min, sec;
};

void tick(Clock *ptr);
void show(Clock *ptr);

int main (void) {

    for(int i = 0; i < 6; ++i) {
        tick(&ptr);
        show(&ptr);
    }
    return 0;
}
```

PSet Clock: Step 2

- Rewrite the code using a pointer ***ptr** and new instead of **Clock clock**;

ver.1

```
#include <iostream>
#include <iomanip>
struct Clock{
    int hr, min, sec;
};

void tick(Clock *ptr);
void show(Clock *ptr);

int main (void) {
    Clock clock = {14, 38, 56};

    for(int i = 0; i < 6; ++i) {
        tick(&clock);
        show(&clock);
    }
    return 0;
}
```

ver.2

```
#include <iostream>
#include <iomanip>
struct Clock{
    int hr, min, sec;
};

void tick(Clock *ptr);
void show(Clock *ptr);

int main (void) {
    Clock *ptr = new Clock {14, 38, 56};

    for(int i = 0; i < 6; ++i) {
        tick(ptr);
        show(ptr);
    }
    return 0;
}
```

PSet Clock: Step 3

- Rewrite ver.2 using **pClock** alias of a pointer to a **struct**.

ver.2

```
#include <iostream>
#include <iomanip>
struct Clock{
    int hr, min, sec;
};

void tick(Clock *ptr);
void show(Clock *ptr);

int main (void) {
    Clock *ptr = new Clock {14, 38, 56};

    for(int i = 0; i < 6; ++i) {
        tick(ptr);
        show(ptr);
    }
    return 0;
}
```


PSet Clock: Step 3

- Rewrite ver.2 using **pClock** alias of a pointer to a **struct**.

ver.3

```
#include <iostream>
#include <iomanip>
struct Clock{
    int hr, min, sec;
};
using pClock = Clock*;
void tick(pClock ptr);
void show(pClock ptr);

int main (void) {
    pClock ptr = new Clock {14, 38, 56};

    for(int i = 0; i < 6; ++i) {
        tick(ptr);
        show(ptr);
    }
    delete ptr;
}
```

ver.3

```
void tick(pClock ptr) { // by one second.
    ptr->sec++;

    // your code here

}

void show(pClock ptr) {
    std::cout.fill('0');
    std::cout << std::setw(2) << ptr->hr << ":"
               << std::setw(2) << ptr->min << ":"
               << std::setw(2) << ptr->sec << std::endl;
}
```

PSet Clock: Step 4

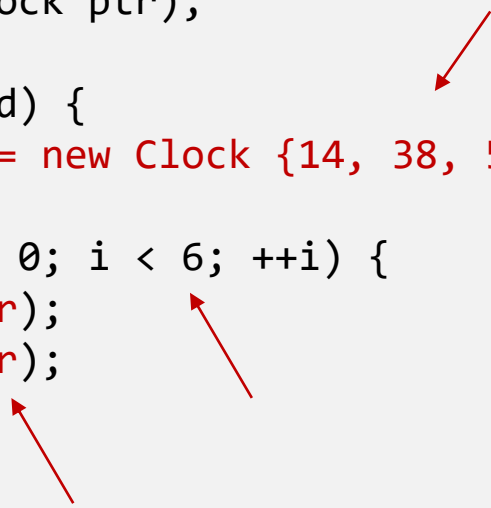
- Remove magic numbers. Do you have any idea?

ver.3

```
#include <iostream>
#include <iomanip>
struct Clock{
    int hr, min, sec;
};
using pClock = Clock*;
void tick(pClock ptr);
void show(pClock ptr);

int main (void) {
    pClock ptr = new Clock {14, 38, 56};

    for(int i = 0; i < 6; ++i) {
        tick(ptr);
        show(ptr);
    }
    delete ptr;
}
```



PSet Clock: Step 4

- Remove magic numbers. Do you have any idea?

ver.4

```
#include <iostream>
#include <iomanip>
struct Clock{
    int hr, min, sec;
};
using pClock = Clock*;
void tick(pClock ptr);
void show(pClock ptr);
void runs(pClock ptr);

int main (void) {
    pClock ptr = new Clock {14, 38, 56};

    runs(ptr);

    delete ptr;
}
```

ver.4

```
void runs(pClock clk) {
    while(true) {
        sleep(1);

        // your code here
    }
}
```

Hint: Use '**\r**' instead of '**\n**' to prevent it from printing a new line.

Replace **show()** with **runs()** such that it ticks and redisplay the time **at the same line** continuously.

On a mac terminal, '**\r**' may **not** work, then use '**\n**'.

PSet Clock: Step 5

- Create **clock.h**, **clock.cpp**, and **clockDriver.cpp** such that they can separate the implementation from interface. Make your files work with **clockDriver.cpp** as provided.

clockDriver.cpp

```
/*
 * C++ for C Coders & Data Structures
 * Lecture note by idebtor@gmail.com
 * This code explains:
 *   - struct and its initialization, using alias
 *   - pointer to struct, new/delete, optional argument
 *   - SIIS(Separation of Interface and Implementation)
 *   - NMN(No Magic Number)
 * 2019/02/15 - created by idebtor@gmail.com
 */
#include "clock.h"
int main (void) {
    pClock clk = new Clock {11, 58, 56};
    for(int i = 0; i < 6; ++i) {
        tick(clk);
        show(clk);
    }
    runs(clk, '\r');
    delete clk;
}
```

On a mac terminal, '\r' may **not** work, then use '\n'.

PSet Clock: Step 5

- Create **clock.h**, **clock.cpp**, and **clockDriver.cpp** such that they can separate the implementation from interface. Make your files work with **clockDriver.cpp** as provided.
 - Do **not** use "using namespace std;" in these files at all.
- Keep the function prototypes in **clock.h** as shown below:

```
void tick(pClock clk);  
void show(pClock clk, char end = '\n');  
void runs(pClock clk, char end = '\n');
```

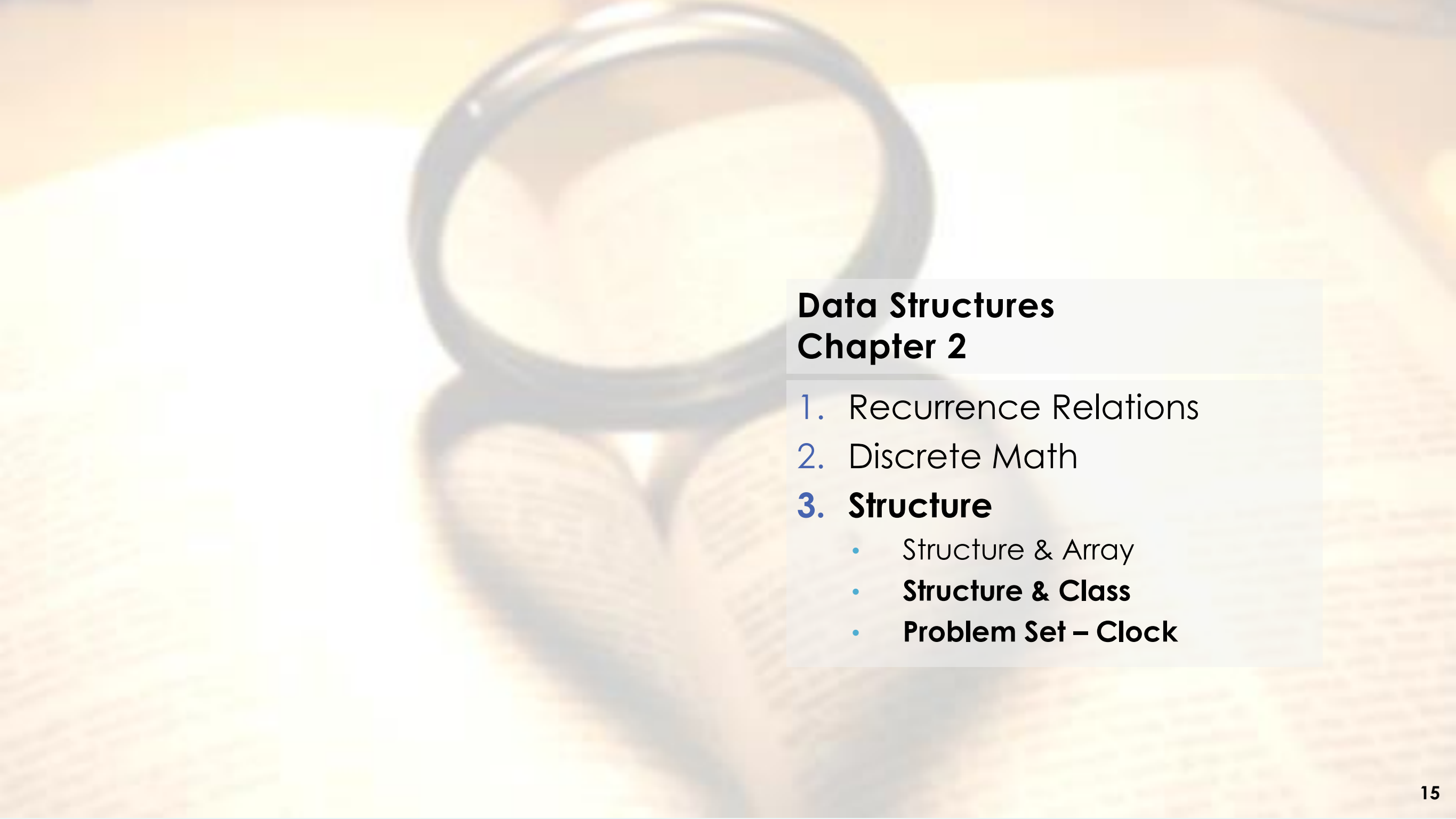
- Use an optional argument. It help you keep DRY principle.
- Sample run:
"-I./" is **unnecessary** since it looks for header files in the current folder by default.

```
PS C:\GitHub\nowicx\psets\pset04struct> g++ clockDriver.cpp clock.cpp -I./ -o clock  
PS C:\GitHub\nowicx\psets\pset04struct> ./clock  
11:58:57  
11:58:58  
11:58:59  
11:59:00  
11:59:01  
11:59:02  
12:08:02
```

← This tick keeps on running.

PSet - Clock:

- Files provided:
 - sample executables
- Files not provided
 - **clockDriver.cpp - do not change, do not submit.**
- Files to submit:
 - **step 3: clock3.cpp**
 - **step 4: clock4.cpp**
 - **step 5: clock.h & clock.cpp**
- Due:
 - 11:55 pm



Data Structures Chapter 2

1. Recurrence Relations
2. Discrete Math
- 3. Structure**
 - Structure & Array
 - **Structure & Class**
 - **Problem Set – Clock**