Lab 20
Type Definition: struct

There are million type of complicated things, which can’t describe just as text, numbers, etc…

Struct: A collection of data items of diverse types

Structure

```
struct Struct_tag {
    type1 member_name1;
    type2 member_name2;
    ...
};
```

other functions …
int main() { … }

Example of struct

Let’s say we want to define a type – Name

What component makes up the whole name?

Given name, Surname, middle name.

struct Name need variables to hold these values.

```c
struct Name{
    string lastname;
    string firstname;
    bool hasMiddle;
}
```
Example of struct

After defining structure -Name, you may create Name variables.

You may use this collection of data as a whole, called structure value. Pick out each data from this collection, called member value.

Ex:
Name name_variable;
name_variable //structure variable
name_variable.member_name //member variable
Example of struct

Each Name variable now has 3 member variables
To assign the value to these variable

-we may assign the structure variable as a whole
  ex:
  Name n1, n2;
  n1 = n2;

-or assign each member variable individually
  ex:
  Name n1;
  n1.lastname = “An”;
  n1.firstname = “Guozhen”;
struct variable as function arguments

void print(Name n)  //this creates a copy of type Name

-pass by value
-whatever change made to this parameter, has nothing to do with the argument one

void print(Name &n)  //n in this function is alias for the struct variable which has passed in as an argument

-pass by reference
-whatever change made in this function, also effects the struct variable which passed in.
May return a struct

Ex:

```c
Name createNew(){
    Name temp;
    temp.lastname= “A”;  
    temp.firstname= “B”; 
    return temp;
}
```
Using struct in another struct

As the struct is a collection of data of diverse type, which including other defined type.

```c
struct Student{
    Name name;
    double GPA;
    string id;
};

Student s1; //Student
s1.GPA; //double
s1.id; //string
s1.name; //Name
s1.name.lastname; //string
s1.name.hasMiddle; //bool
```
Object Oriented

In real world, there are millions of types of things.

With the object oriented design, also known as OO, we can define our own types, to better model the real world problems we want to solve.

Writing and designing software is really an art as much as it is science.
Class

To define an object, there are two parts:

- What components make up the object? collection of data

- What actions does the object perform? function

Class is a custom defined data type which holds both data and functions.

Object is an instantiation of a class. The variable of this custom defined type.
Class

Example: String
- Create a string object
- Manipulate (concatenate, insert, ...)
- Information about this object (length, substring)

We only know there are functions that associate with this object, but we do not care how the data is stored and how the function is implemented to achieve these actions.
Class vs Struct

In user defined type - structure, we used . (dot) to get to it’s member variable. Thus, we have to be aware exactly what made up the structure, what are the member name. One principle of OOP is encapsulation. All the internal information is hiding from the rest of the world.

Ex:

Let’s say we want to define a real world type – Student. What component makes up the student?
Student ID, Last Name, First Name, Gender, GPA, Email address

Class Student need variables to hold these data.
string Student ID,
string Last Name,
string First Name,
string Gender,
double GPA,
string Email address
Example

After defining variables, it also need to act on the variables.

Ex:
Student ID – to setup the student ID, ask for the student ID
One function perform one action.
setStudentID, getStudentID - One mutator, one accessor
For each variable, at least need two member functions to set and
get this variable.

Prototypes:
void setStudentID(string id);
string getStudentID();
Sample class

class Student
{
    private:
    string student_ID, last_name, first_name;
    double gpa;
    Date dob;

    public:
    void setStudentID(string id);
    string getStudentID();
    void setLastName(string lname);
    string getLastName();
    void setGPA(double gpa);
    double getGPA();
    ...
};
Accessibility

For each variables and functions, we can define its accessibility. Access modifier decides who can access the variables or the functions from outside of the class.

**Private** means only members of the class (or friends) can access the variables and modify it.

**Public** means it (member function or variable) can be access directly from outside of the class where this class is visible.

Note: for Student Class as example.
Let’s say we created Student s1 in a main function outside of class Student. We can not modify it’s gpa variable directly by s1.gpa = 2.0 from main function, because this variable is **private**.
I can call it’s member function s1.setGPA( 2.0 ) to set the gpa value, because this member function is **public**.