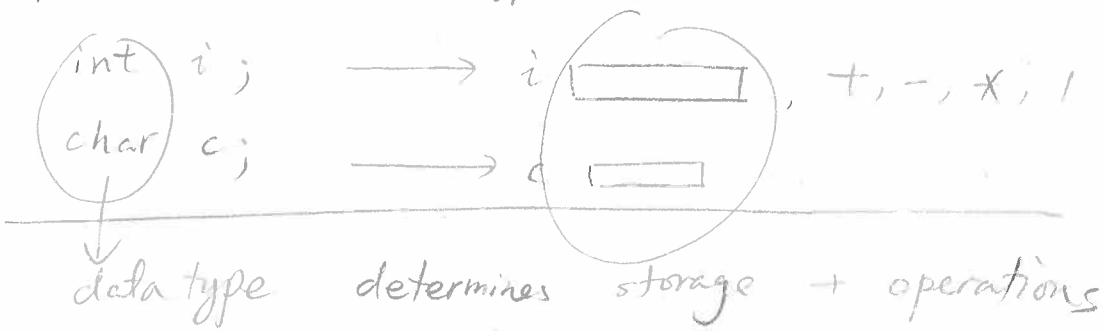
Chapter One

Data structure: • a systematic way of organizing & accessing data

• not only store data, but also have operations that manipulate the data

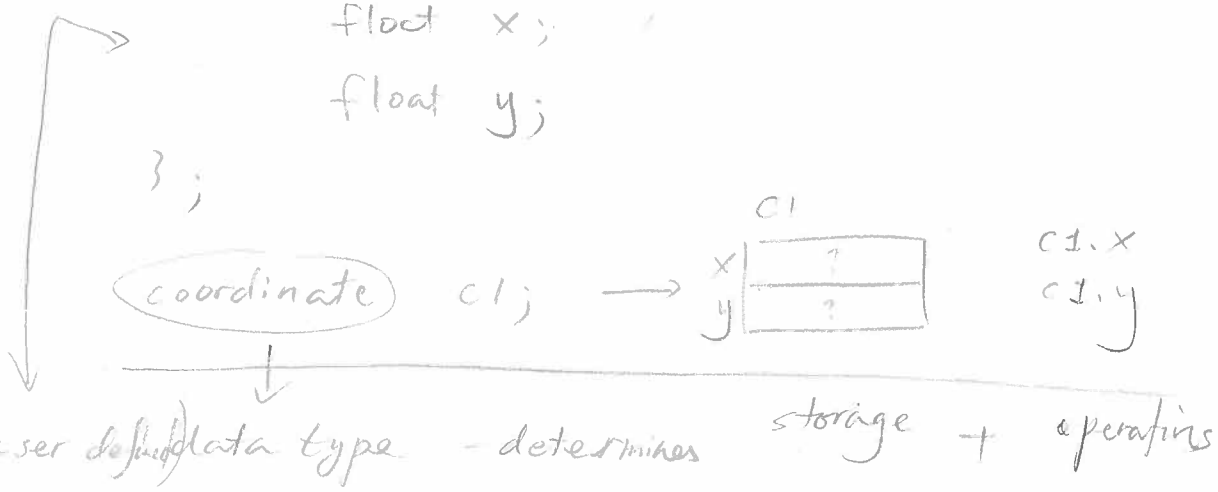
Examples of data structure:

① primitive data type (int, char in C++)



② structures that groups together different but related data types

```
struct coordinate {
    float x;
    float y;
};
coordinate c1;
```



Observe the difference between ① + ②

- ① is part of a language definition
- ② is defined by a user / programmer

When a user wants to define their own data structure, a model called ADT (abstract data type) is usually created first.

ADT: :• a model that gives a simple & clear description of
- the type of data stored
- the operations that support them

• "Abstract" implies that ADT is implementation-independent. It describes "what" instead of "how".

* ADT is like an architect's first sketch of a house. We need blue print & physically build the house.
* For an ADT, we need to create a physical representation of the data and a computer model (program) for the operations

The time24 ADT

Consider a data structure that uses integer values for hours and minutes to represent the time of data in a 24 hour clock.

- hours are in the range 0 to 23
- minutes are in the range 0 to 59

Here is the ADT for time24:

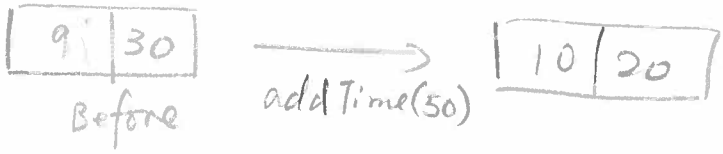
- Data members: hours (0 ~ 23)
  minutes (0 ~ 59)

- operations: (a process that can accept data, carry out calculation & return a value)
  (next page)

Transformer

- addTime(m): update the current time by adding m minutes & adjust the hours + minutes to fall in their specified range

  postcondition: The new time is m minutes later



- duration(t): Time t is input. Measure the length of time from the current time to time t, and return the result as time24 value

  precondition: Time t must not be earlier than the current time

Observer

- readTime(): Input from keyboard the hours, mins (hh:mm)

  postcondition: set hour & minute values to hh & mm adjust them to normal range if needed

- writeTime(): Display on the screen the current time in the form hh:mm

- getHour( ): return the hour value for the current time

- getMinute( ): return the minute value for the current time

Inline function

```
class Rectangle {
    double area( ) const
    { return length * width; }
    void setSide(double len, double wid)
    { width = wid;
      length = len;
    }
};
```

```
r.setSide(8, 10);
cout << r.area();
```

compiler →

```
r.width = 10;
r.length = 8;
cout << r.length * r.width;
```

ADT Notation

• ADT Operation Description

- operation Name:

  Action statement specifies the input argument, the type of operation on the elements of the data structure & the output value

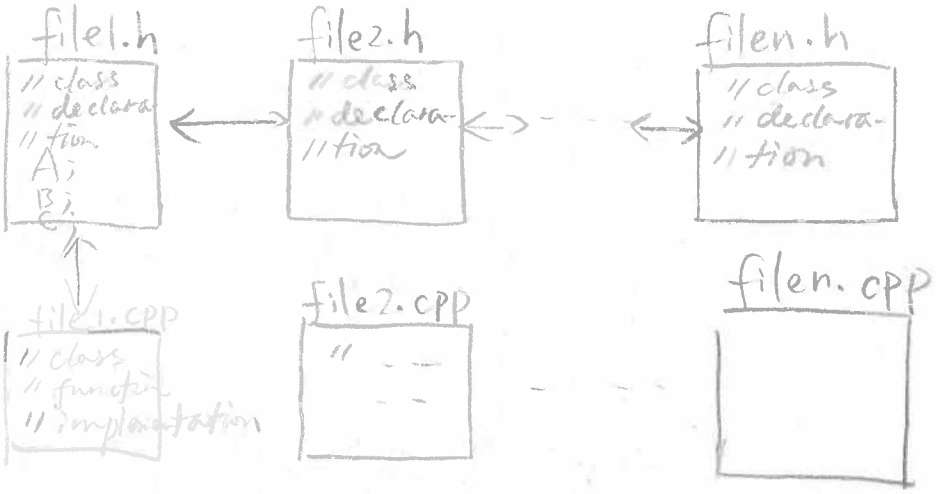
- Preconditions:

  Necessary conditions that must apply to the input arguments & the current state of objed to allow successful execution of the operation

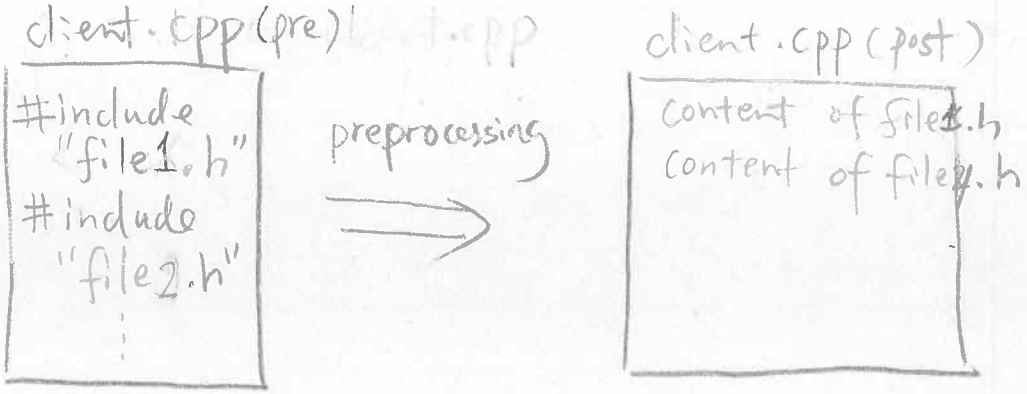
- Post condition:

  Changes to the data structure caused by the operation

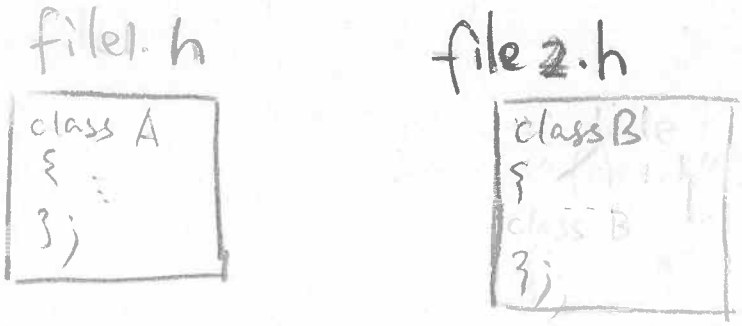
Multiple Files Project

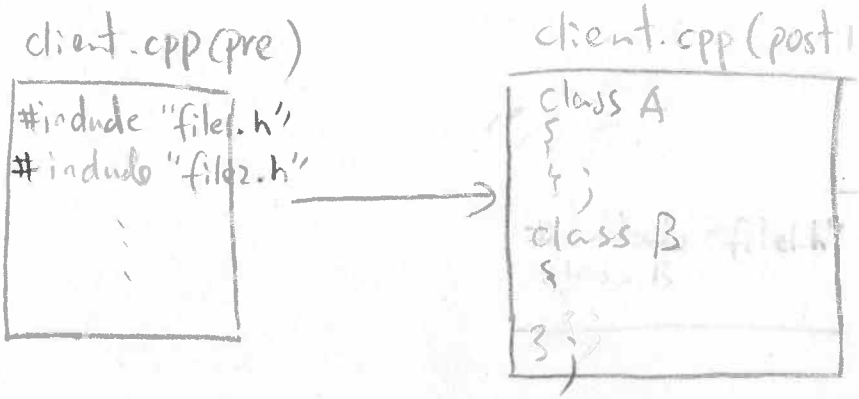


Preprocessing



ex.





- For an ADT, we need to create a physical representation of the data and an implementations for the operations.

- C++ class construct provides a physical realigatim of an ADT.

C++ class construct syntax

Two parts:

- class declaration
  - a blue print of user defined data type
  - "What" in terms of programming language

- class implementatin
  - build the class according to "blue print" in programming language

C++ class syntax:

1) class declaration (className.h)

```
class className {

    private:
        // data members
        // prototypes of member functions

    public:
        // data members
        // prototypes of member functions

};
```

Accessible to all member functions

Accessible to any programs

- "private" can be ignored if it goes before "public". Anything before "public" is "private" by default.
- Everything defined in class can be accessed by all class members.
- Only things defined in "public" can be accessed by programs outside the class.

2) class implementation (member function) (className.cpp)

```
ReturnType  className::memberFunctionName( ... )
{

  ...

}
```

- if a member function uses data members or member functions within the same class, use directly.

3) Use of c++ classes in programs

3.1) include the header file of class

```
#include "time24.h"
```

3.2) define a variable of type "className"

```
time24  t1, t2;
```

- t1 and t2 are objects/instances of class "time24"

3.3) to use member functions (public ones)

```
t1.readTime( );
t2.readTime( );
t1.duration(t2);
```

objectName . memberfunctionName(...)

Constructor

- A special member function to initialize the object.
- has the same name as the class, no return type
- all classes have default constructor given by the language;

```
className( );
// 1) allocate/set aside memory space for
//    the data members of the class
// 2)
className::className( )
{ // do nothing
}
```

- users/programmers can {overwrite default write their own} constructs

1)
```
className( int i );

className::className( int i )
{  // do something useful besides allocate
}  // memory
```

2)
```
className( );

className::className( )
{  // do something here besides
   // allocate memory
}
```

className

BUT "function name", return type — NO

Destructor

- A special member function to release the memory allocated for the object
- Name must be ~className()
- No return type
- Not called explicited
- All classes have default destructor given by the language

```
~className();

className::~className()
{  // release memory for data members

}
```

- users/programmer can over write default (or write their own) destructors

1) ~className();

```
className::~className()
{   // release data member memory

    // release dynamically created memory
}
```

2) ~className(<param list>);

```
className::~className(<param list>)
{   // release memory (static & dynamic)
    // do something else
}
```