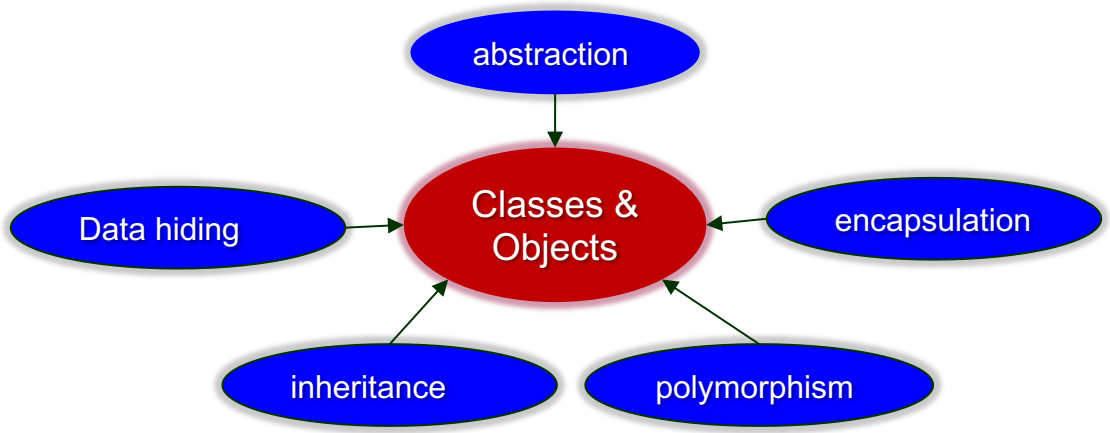
Object Oriented Style



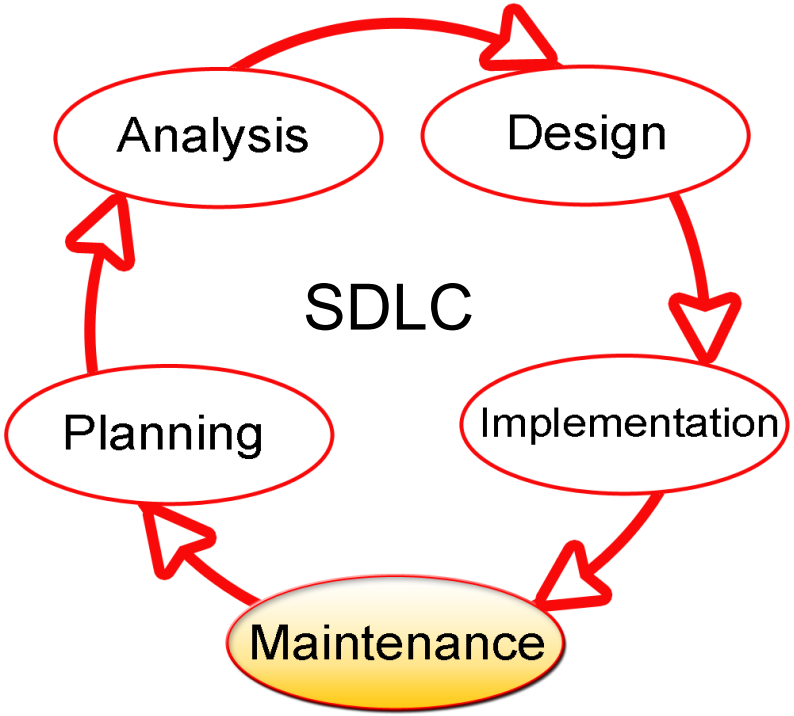
Object Oriented Programming

Focus of assignment 1:

- Abstraction
- Encapsulation
- Data hiding

Software Development Life Cycle

Iteration and Evolution



OO Style for Assignment 1 – levels

1. Procedural code for Tic Tac Toe game
2. OO code for Tic Tac Toe (starting point for assignment 1)
3. Class Board – single class solution
4. Add Class Game - two class solution
5. Add Class Player and more – multiple class solution

```
class Player {
protected:
  int player;
public:
  virtual void getMove(Board*,
          int&, int&) = 0;
  …
};
```

```
class Board {
  int grid[boardSize][boardSize];
public:
  char addMove(int x1,int y1,int x2,int y2);
  void winningStatus();
  bool isValid(int x,int y);
  bool isFull();
  void displayBoard();
  …
  // char play() // single class solution
};
```

```
int main() {
  Player* players[2];
  players[0] = new HumanPlayer(1);
  players[1] = new MindfulPlayer(-1);
  Game game(players[0], players[1]);
  game.play();
  // Board board;
  // board.displayBoard();
}
```

```
class Game {
  Board board;
  //Player* players[2]; //for three+ class solution
public:
  //Game(Player*, Player*); //for three+ class solution
  char play(); //two or multiple class solution
  …
};
```

Lecture 7

Strings, Stream, Static Constant and File I/O

Topics covered by last lecture

- Pointers
- Pointers and arrays
- Pointers and functions
- new operator
- Dynamic memory allocation

Would need a few weeks of practice to get used to

Buy a house

```
House myhouse;
 //you live there until you die
```

Rent a house

```
House *housekey = new House;
   //you live there as long as you pay rent
delete housekey;
```

Topics covered by this lecture

- String
- Stream
- file I/O
- Static variable
- Constant variable
- Copy constructor

Hear the words in the lecture, learn their meaning from the Internet.

Two string types

string1.cpp

- C-string: inherited from C
  - Define a string as an array of char, say

    char s[10];
  - End of string will be marked with null, '\0', automatically
  - Operate the string as an array with dozens of extra functions, a bit complicated and hard to remember though.

```
#include <cstdlib>
#include <cstring>
```

Highly recommended

- Standard class: string
  - Header file: `#include <string>`
  - Define a string as an object of Class string, say

    `string s;`
  - Also dozens of member functions for string operations

I/O with Class string

stringIO.cpp

- Use `cin` and `cout` for input and output

```
string s;
cin >> s;
cout << s;
```

- It stops at a whitespace:

```
Input: "Hello dongmo!"
Output: "Hello"
```

- For complete lines, use `getline()` function:

```
string line;
cout << "Enter a line of input: ";
getline(cin, line);
cout << line << endl;
```

getline(): more options

getline.cpp

- Can stop reading by specifying "delimiter" character:

```
string line;
cout << "Enter input: ";
getline(cin, line, '?');
```

  – Receives input until "?" encountered

- Be careful mixing `cin >> var` and `getline()`

```
int n;
string line;
cin >> n;
getline(cin, line);
```

Note that `cin>>n` stops at "\n", leaving it on the stream for `getline()`!

Typical C-style functions

- atof: Convert string to double
- atoi: convert string to integer
- strcopy: copy string
- strcat: concatenate strings
- strcomp: Compare two strings
- strchr: Locate first occurrence of character in string
- strstr: Locate substring

Some Member Functions of Class string

Display 9.7 Member Functions of the Standard Class `string`

EXAMPLE	REMARKS
Constructors	
`string str;`	Default constructor; creates empty `string` object `str`.
`string str("string");`	Creates a `string` object with data `"string"`.
`string str(aString);`	Creates a `string` object `str` that is a copy of `aString`. `aString` is an object of the class `string`.
Element access	
`str[i]`	Returns read/write reference to character in `str` at index `i`.
`str.at(i)`	Returns read/write reference to character in `str` at index `i`.
`str.substr(position, length)`	Returns the substring of the calling object starting at `position` and having `length` characters.
Assignment/Modifiers	
`str1 = str2;`	Allocates space and initializes it to `str2`'s data, releases memory allocated for `str1`, and sets `str1`'s size to that of `str2`.
`str1 += str2;`	Character data of `str2` is concatenated to the end of `str1`; the size is set appropriately.
`str.empty( )`	Returns `true` if `str` is an empty `string`; returns `false` otherwise.

(continued)

Typical Member Functions of Class string

Display 9.7 Member Functions of the Standard Class `string`

EXAMPLE	REMARKS
`str1 + str2`	Returns a `string` that has `str2`'s data concatenated to the end of `str1`'s data. The size is set appropriately.
`str.insert(pos, str2)`	Inserts `str2` into `str` beginning at position `pos`.
`str.remove(pos, length)`	Removes substring of size `length`, starting at position `pos`.
Comparisons	
`str1 == str2  str1 != str2`	Compare for equality or inequality; returns a Boolean value.
`str1 < str2  str1 > str2`	Four comparisons. All are lexicographical comparisons.
`str1 <= str2  str1 >= str2`	
`str.find(str1)`	Returns index of the first occurrence of `str1` in `str`.
`str.find(str1, pos)`	Returns index of the first occurrence of string `str1` in `str`; the search starts at position `pos`.
`str.find_first_of(str1, pos)`	Returns the index of the first instance in `str` of any character in `str1`, starting the search at position `pos`.
`str.find_first_not_of (str1, pos)`	Returns the index of the first instance in `str` of any character *not* in `str1`, starting search at position `pos`.

Learn more from the Internet.

compare.cpp

C-string and string: object conversions

- Automatic type conversions
  - From c-string to string object:
    ```
    char aCString[] = "My C-string";
    string stringObj = aCstring;//copy constructor
    ```
    » Perfectly legal and appropriate!
  - From string object to c-sting
    ```
    aCString = stringObj; //Illegal!
    ```
    » Cannot automatically convert a string object to a c-string
  - Must use explicit conversion:
    ```
    strcpy(aCString, stringObj.c_str());
    ```

Remember the way of conversion,
get benefit from both

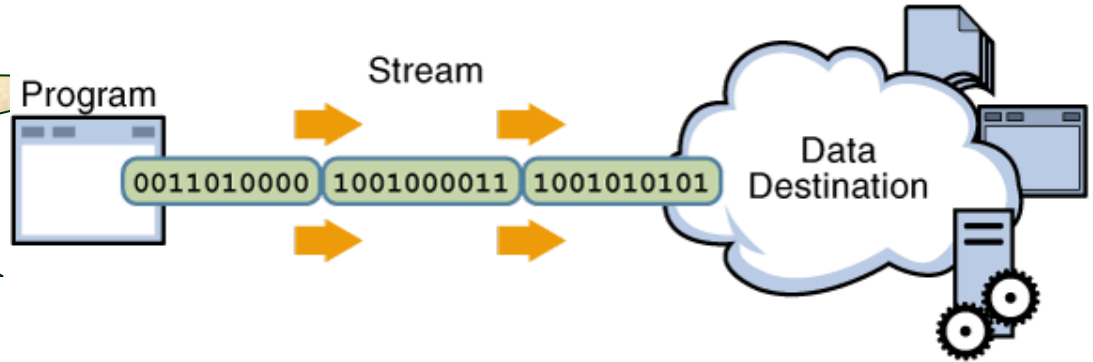
Topics covered by this lecture

- String
- Stream
- file I/O
- Static variable
- Constant variable
- Copy constructor

Streams

- A stream is a flow of characters.
  - If the flow is into your program, the stream is called an input stream (istream).
  - If the flow is out of your program, the stream is called an output stream (ostream).
- Most useful stream classes
  - <iostream>: input and output via standard devices
  - <fstream>: input and output via files
  - <stringstream>: string operation via buffer

Facilitate data transfer from/to external devices



Streams

Automatically match data type

- Use operators: >> and <<:
  - Assume that `inStream` is a stream that comes from some file:

```
int value;
inStream >> value;
```

    » Reads value from the stream, assigned it to *value* as an integer

```
double num;
inStream >> num;
```

    » Reads value from the stream, assigned it to *num* as a double.

  - Assume that `outStream` is a stream that goes to some file

```
outStream << value;
```

    » Writes value to stream

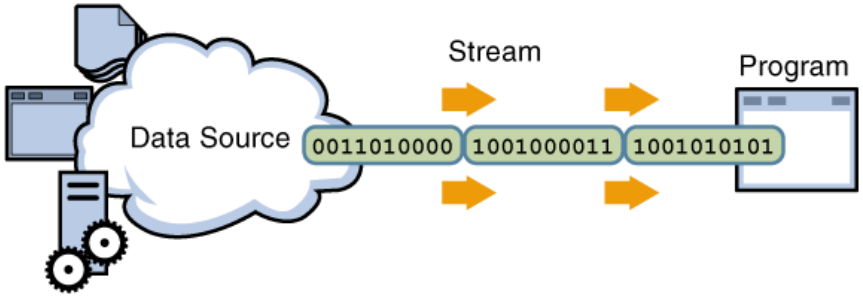
It's called buffer in Java.

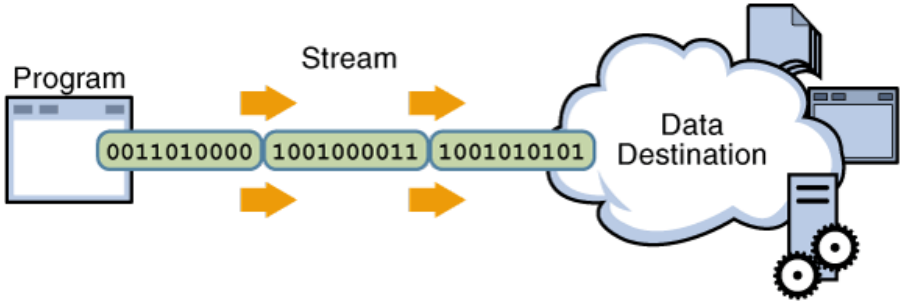
I/O stream character operations

- More I/O stream member functions:
  - get(char)
  - getline(string)
  - put(char)
  - putback(char) // used in istream
  - peek() // reads the next character from the input stream without extracting it
  - ignore()

Topics covered by this lecture

- String
- Stream
- file I/O
- Static variable
- Constant variable
- Copy constructor





File Input: a template

```
#include <iostream>
#include <fstream>
use namespace std;
int main( )
{
  ifstream fin;
  fin.open("data_in.txt");

  string word;

  while (!fin.eof()) {
    fin >> word;
    cout << word << endl;
  }
  fin.close( );

 return 0;
}
```

File I/O stream header file

File input stream object

Open file

Read from the stream word by word

Close file

fileInput.cpp

File output: a template

```
#include <iostream>
#include <fstream>
use namespace std;
int main( ) {
  ofstream fout;

  fout.open("data_out.txt");
  int x = 2;
  int y = 3;

  fout << "x + y = "
       << (x+y);

  fout.close( );

  return 0;
}
```

File I/O stream header file

File input stream object

Open file

Output to the stream

Close file

fileOutput.cpp

Appending to a File

- Standard open operation begins with empty file
  - Even if file exists, contents lost
- Open for appending:

```
ofstream fout;
fout.open(“data_out.txt", ios::app);
```

  - If the file doesn’t exist, create one
  - If the file exists, append the new input to the end of the file

Checking End of File

- Use loop to process file until end
- Two methods to test *end of file*
  - Use member function eof()

```
ifstream fin;
fin.open("data_in.txt");
char next;
while (!fin.eof()){
     fin.get(next);
     cout << next;
}
```

  - Reads each character until file ends
  - eof() member function returns bool

End of File Check with Read

- Use >> operator

```
ifstream fin;
fin.open("data_in.txt");
double next, sum = 0;
while (fin >> next) {
   sum =+ next;
}
cout << "the sum is " << sum << endl;
```

- Read operation returns `bool` value!
  `(fin >> next)`
  - » Expression returns `true` if read successful
  - » Returns `false` if attempt to read beyond end of file
- Copy-and-paste would introduce invisible characters into a file, which might cause bugs.

Tools: File Names as Input

- Stream open operations

```
char fileName[16];
ifstream fin;
cout << "Enter file name: ";
cin >> fileName;
fin.open(fileName);
```

  – Allows the user to provide file name in real time

  – Include a full path to the file in filename unless it is located in the default folder

Formatting Output with Stream Functions

- Format decimal number output:

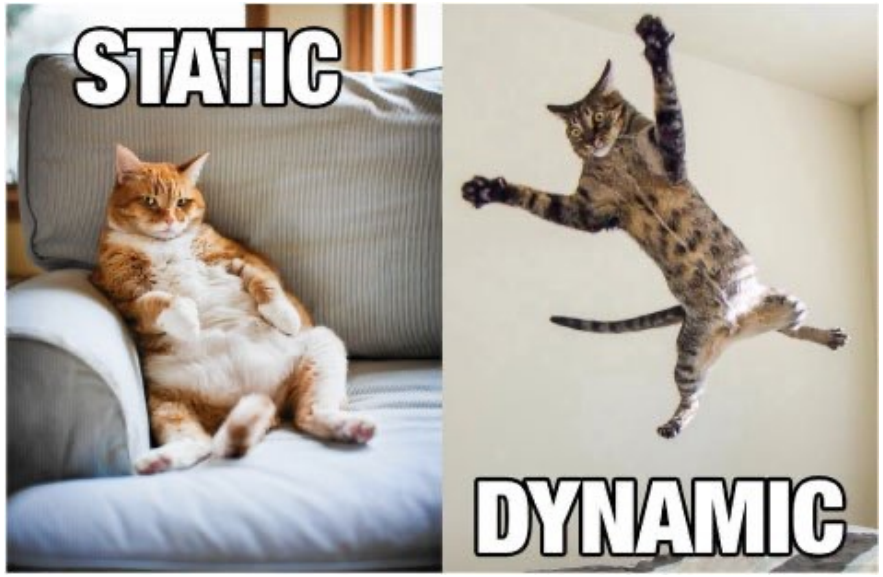
```
fout.setf(ios::fixed);
fout.setf(ios::showpoint);
fout.precision(2);
```

- They are in library <`iomanip`>, called manipulator.
- Member function setf()
  - Allows multitude of output flags to be set
- Member function precision(x)
  - Decimals written with "x" digits after decimal
- Member function width(x)
  - Sets width to "x" for outputted value
  - Only affects "next" value outputted

formating.cpp

Topics covered by this lecture

- String
- Stream
- file I/O
- Static variable
- Constant variable
- Copy constructor



Static variables

Think about the difference between the following two versions of function func():

```
int func() {
    int i = 0;
    i++;
    return i;
}
```

```
int func() {
    static int i = 0;
    i++;
    return i;
}
```

```
int main() {
   cout << func() << endl;
   cout << func() << endl;
   cout << func() << endl;
  return 0;
}
```

staticinFunction.cpp

Static variables

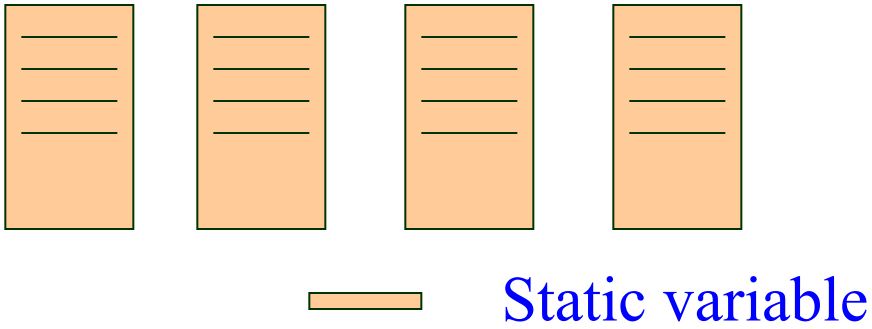
- In C/C++, the keyword **static** has two basic meaning:
  - Allocated once at a fixed address: *static storage*.
  - Only initialize once (it's done when the program is compiled).

- Static variables in a function: *remember values between function calls (initialized when the first time being called and remain the value for further call without re-initialized).*
- Static variables in a class: *belongs to the class but not to an object. All the objects of the class share the same static variables.*

Static data members of a class

Properties of a static data member

- All objects of the class "share" the same storage
- Useful for "tracking" objects of the same class
  - How often a member function is called?
  - How many objects of a class have been created?
  - Identify different object, say object ids.



Static data members of a class

- A static member in a class should be initialized independently to individual objects.
- The way to initialize a static member of a class:
  - int WithStatic::x = 10;

StaticInClass.cpp

Different from other types of variables

Static member functions of a class

A member function in a class can also be static:

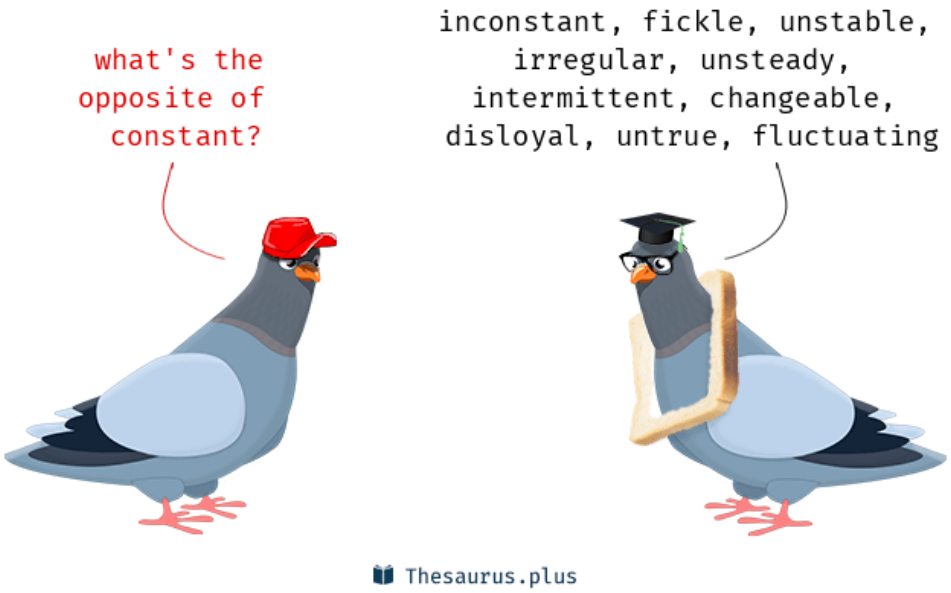
- It is independent to the objects of the class, which requires all variables in a static function needs to be static.
- Normally called by using the class name even though calling from an object is ok.
  - ClassName::StraticFunction();

Note that main function is a static function in any class in Java

SimpleStaticMemerFunction.cpp

Topics covered by this lecture

- String
- Stream
- file I/O
- Static variable
- Constant variable
- Copy constructor



The concept of constants

- The concept of constant (const in C++) is created to allow the programmer to draw a line between what changes and what doesn't. It's a technique that a programmer takes use of compiler to help him check his programs.
- const is used for variables which values never change once they are initialized. It can be also used for functions.
- A constant variable must be initialized while its declaration. You will get an error message if you accidentally try to change the value of a constant.
- Pre-processor vs constants:
  - C style: #define BUFSIZE 100
  - C++ style: const int bufsize = 100;

constant.cpp

Constant objects and member functions

- Constants can be any user defined data type.
  - const Date newYear(1,1);
  - const Date ChristmasOfTheYear(25,12,2017);
- A constant object can only call a constant member function.
- A member function that is specifically declared const is treated as one that will not modify data members in the class or will not call a non-const member function.
  - int getDay() const;
  - void printDate() const;

Constant member functions

- *const* modifier after function declaration indicates that the function does not modify the state.

```
class Date {
    int d, m, y;
  public:
    int getDay() const { return d ; }
    int getNextYear() const ;
    // …
  };
```

Time.h

timeApp.cpp

```
int Date::getNextYear() const { return y++; }        // Error
int Date::getNextYear() { return y; }                //wrong
int Date::getNextYear() const { return y; }          // correct
```

Topics covered by this lecture

- String
- Stream
- file I/O
- Static variable
- Constant variable
- Copy constructor

Copy constructor: a review

A copy constructor of a class is a special constructor for creating a new object as a copy of an existing object. The copy constructor is called whenever an object is initialized from another object of the same class. Typical declaration of a copy constructor:

```
ClassName(const ClassName&);
```

```
Board(const Board& cboard) {
    boardSize = cboard.getBoardSize();

    grid = new int*[boardSize];
    for (int i = 0; i < boardSize; i++)
        grid[i] = new int[boardSize];

    for (int i = 0; i < boardSize; i++)
        for (int j = 0; j < boardSize; j++) {
            grid[i][j] = cboard.grid[i][j];

}
```

Ways of using copy constructor:

```
Board tempBoard = board;

Board tempBoard(board);

Board* tempBoard
    = new Board(board);
```

Homework

- Read relevant content in Chapters 7, 9 & 12.
- Please work on your assignment 1. You have all knowledge now for all the tasks. However, if you still have difficulties to complete practical tasks up to practical 5, please come to PASS or try to get help from your tutor.
- If you have finished your assignment 1, your tutor can mark it in the practical class and you can always improve your code if you do not like your marks.

You are not allowed to fail this assignment!