Basics: From C to C++

Computer Programming for Engineers (DSAF003-42)
Fall, 2021

Practice 6 : ctors and other tools

Instructor:

Youngjoong Ko (nlp.skku.edu)

Class with Constructor Example

- **Can overload constructors just like other functions**
  - C++11 supports a feature called member initialization
- **Various constructor definition**

```
1   #include <iostream>
2   using namespace std;
3
4   class Box
5   {
6       int width  = 10;
7       int height = 10;
8       int depth  = 10;
9
10  public:
11      Box( int, int, int );
12      Box( int, int );
13      Box( int );
14      Box( );
15      void show();
16  };
```

```
17  Box::Box( int w, int h, int d )
18  {
19      width = w; height = h; depth = d;
20  }
21
22  Box::Box( int w, int h )
23      : width(w), height(h), depth(0)
24  { }
25
26  Box::Box( int w ) : Box( w, 0, 0 )
27  { }
28
29  Box::Box() { }
```

Class with Constructor Example

- Calling constructors

```
30  void Box::show()
31  {
32      if(height==width&&width==depth) cout<<"Cube\t : "<<width<<", "<<height<<", "<<depth;
33      else if(!height)cout<<"Line\t : "<<width;
34      else if(!depth) cout<<"Quad\t : "<<width<<", "<<height;
35      else            cout<<"Box\t : "<<width<<", "<<height<<", "<<depth;
36  }
37  int main()
38  {
39      Box box( 10, 20, 30 );
40      Box quad( 50, 50 );
41      Box line( 20 );
42      Box cube;
43
44      box.show();  cout<<endl;
45      quad.show(); cout<<endl;
46      line.show(); cout<<endl;
47      cube.show(); cout<<endl;
48  }
```

```
Box     : 10, 20, 30
Quad    : 50, 50
Line    : 20
Cube    : 10, 10, 10
```

Exercise 1

- **Class variables**
  - name, age, adult(bool)
- **Constructor overload**
  - person(name, age, adult), person(name, age)
- **Class functions**
  - isAdult() returns true when a person is adult and returns false otherwise
  - getName() returns name

```cpp
#include <iostream>
#include <string>
using namespace std;

class person{
private:
    string name;
    int age;
    bool adult;
public:
    person( string, int, bool );
    person( string, int );
    string getName();
    bool isAdult();
};
```

```cpp
int main()
{
    person a( "Tom", 35, true );
    person b( "Daniel", 17 );

    string result;
    string adult = " is adult.";
    string kid   = " is not adult.";
    result = a.isAdult() ? adult : kid;
    cout<<a.getName()<<result<<endl;
    result = b.isAdult() ? adult : kid;
    cout<<b.getName()<<result<<endl;

    return 0;
}
```

```
Tom is adult.
Daniel is not adult.
```

Copy constructor and destructor

- **Copy constructor**
  - Automatically called when a class object declared and initialized to other object.
  - But default copy constructor is shallow copy
- **Destructor**
  - Automatically called when an object become out of scope.

```cpp
#include <iostream>
using namespace std;

class DayOfYear{
private:
    int month;
    int day;
public:
    void ShowDate();
    DayOfYear(int a, int b):month(a), day(b){}
    DayOfYear(const DayOfYear& other);
    ~DayOfYear();
};
void DayOfYear::ShowDate(){
    cout<< month << "월 " << day << "일" <<endl;
}
DayOfYear::DayOfYear(const DayOfYear& other){
    this->month = other.month;
    this->day = other.day;
    cout << "call copy constructor" << endl;
}
DayOfYear::~DayOfYear(){
    cout << "call destructor" << endl;
}
int main(){
    DayOfYear birthday(2,5);
    birthday.ShowDate();

    DayOfYear today = birthday;
    today.ShowDate();

    return 0;
}
```

```
2월 5일
call copy constructor
2월 5일
call destructor
call destructor
```

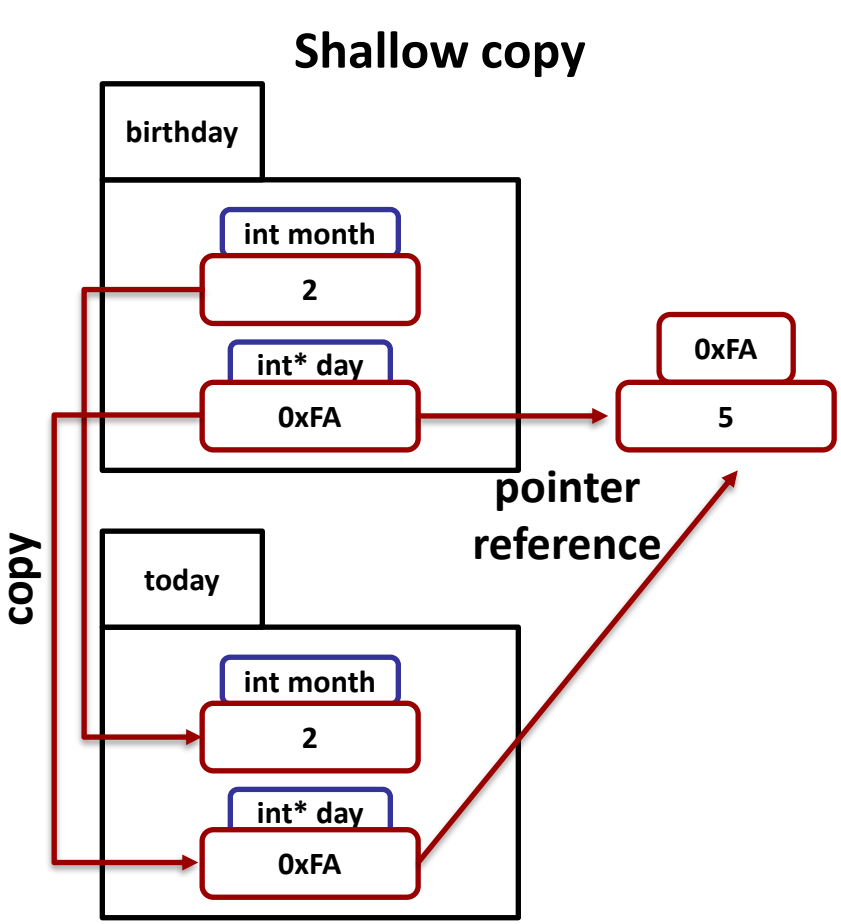
Problem of shallow copy

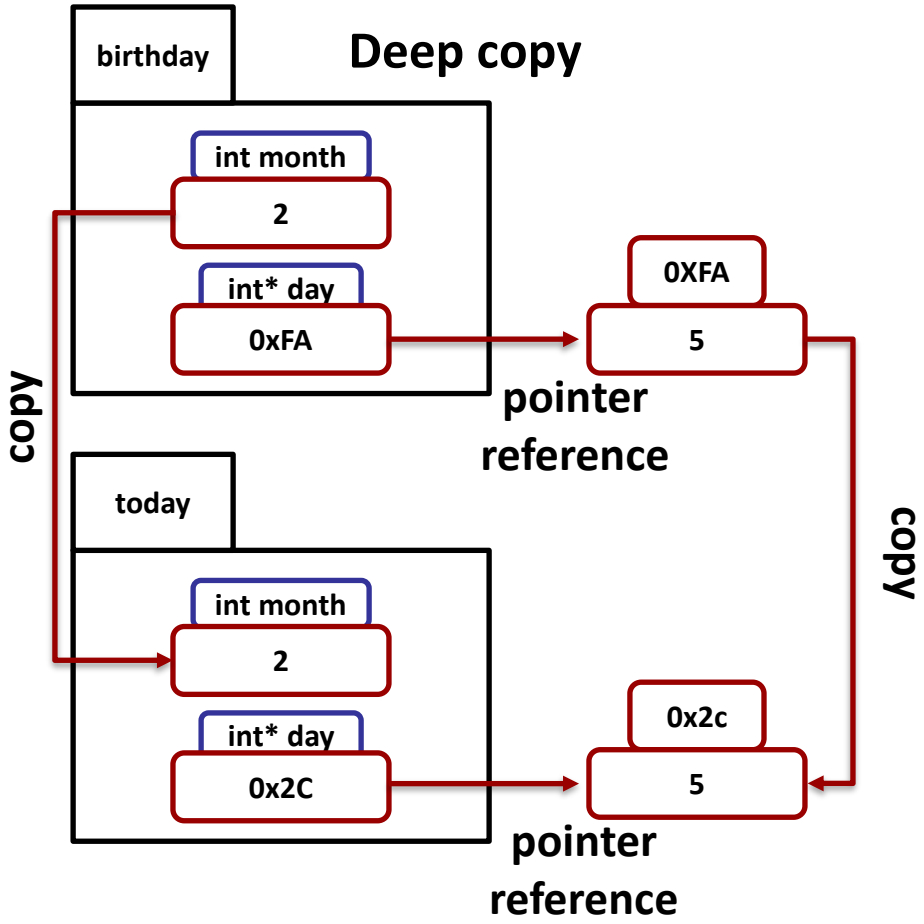
- Default copy constructor is shallow copy
- Releasing dynamic memory is necessary in destructor

```cpp
#include <iostream>
using namespace std;

class DayOfYear{
private:
    int month;
    int* day = new int;
public:
    void ShowDate();
    DayOfYear(int a, int b):month(a){*day=b;}
    DayOfYear(const DayOfYear& other);
    ~DayOfYear();
};
void DayOfYear::ShowDate(){
    cout<< month << "월 " << *day << "일" <<endl;
}
DayOfYear::DayOfYear(const DayOfYear& other){
    this->month = other.month;
    this->day = other.day;
    cout << "call copy constructor" << endl;
}
DayOfYear::~DayOfYear(){
    delete day;
    cout << "call destructor" << endl;
}
int main(){
    DayOfYear birthday(2,5);
    birthday.ShowDate();

    DayOfYear today = birthday;
    today.ShowDate();

    return 0;
}
```

```
2월 5일
call copy constructor
2월 5일
call destructor
free(): double free detected in tcache 2
Aborted
```

Shallow copy vs Deep copy






```
DayOfYear::DayOfYear(const DayOfYear& other){
    this->month = other.month;
    this->day = other.day;
    cout << "call copy constructor" << endl;
}
```

```
DayOfYear::DayOfYear(const DayOfYear& other){
    this->month = other.month;
    *(this->day) = *(other.day);
    cout << "call copy constructor" << endl;
}
```

Deep copy constructor

- Default copy constructor is shallow copy
- Releasing dynamic memory is necessary in destructor

```cpp
#include <iostream>
using namespace std;

class DayOfYear{
private:
    int month;
    int* day = new int;
public:
    void ShowDate();
    DayOfYear(int a, int b):month(a){*day=b;}
    DayOfYear(const DayOfYear& other);
    ~DayOfYear();
};
void DayOfYear::ShowDate(){
    cout<< month << "월 " << *day << "일" <<endl;
}
DayOfYear::DayOfYear(const DayOfYear& other){
    this->month = other.month;
    *(this->day) = *(other.day);
    cout << "call copy constructor" << endl;
}
DayOfYear::~DayOfYear(){
    delete day;
    cout << "call destructor" << endl;
}
int main(){
    DayOfYear birthday(2,5);
    birthday.ShowDate();

    DayOfYear today = birthday;
    today.ShowDate();

    return 0;
}
```

```
2월 5일
call copy constructor
2월 5일
call destructor
call destructor
```

Exercise 2

- **Class variables and constructor, copy constructor, destructor**
  - name, age, adult(bool) / Person(string name, int age)
- **Define member functions**
  - setName(string n) sets name with a string argument
  - getName() returns name
  - setAge(int a) sets age with an int argument
  - isAdult() returns true when a person is adult and returns false otherwise

```cpp
#include <iostream>
#include <string>
using namespace std;

class Person{
private:
    string name;
    int* age = new int;
    bool adult;
public:
    Person(string name, int age);
    Person(const Person& other);
    ~Person();
    void setName(string n);
    string getName();
    void setAge(int a);
    bool isAdult();
};
```

```cpp
int main()
{
    Person Taehun("taehun",26);
    Person Clone = Taehun;

    Clone.setName("dooyoung");
    Clone.setAge(24);

    string result = Clone.isAdult() ? "adult" : "kid";
    cout<<Clone.getName()<<" is "<<result<<endl;
    return 0;
}
```

```
call copy constructor
dooyoung is adult
call destructor
call destructor
```

The const Parameter Modifier Example

- **Protect argument**
  - Makes parameter “read-only”

```
#include <iostream>
using namespace std;
class Box
{
public:
    int width, height;
    const int depth = 10;
    Box( int,int);
    void increase(int &x);
    void show() const;
};
Box::Box( int w, int h ) : width(w), height(h){ }
void Box::increase(int &x){
    x += 1;
}
void Box::show() const
{
cout<<"width: "<<width<<" height: "<<height<<" depth: "<<depth<<endl;
}
```

```
int main()
{
    Box box( 30,20);
    box.show();
    box.increase(box.width);
    box.show();
    // box.increase(box.depth);
    const Box const_box(50,40);
    const_box.show();
    // const_box.increase(box.width);
    return 0;
}
```

```
width: 30 height: 20 depth: 10
width: 31 height: 20 depth: 10
width: 50 height: 40 depth: 10
```

Static Members Example

- **Static member variables and functions**
  - All objects of class “share” one copy
  - Only static members can be referenced in static function

```cpp
#include <iostream>
using namespace std;
class DayOfYear{
private:
    // static int year=2021;
    static int year;
    int month;
    int day;
public:
    DayOfYear(int a, int b):month(a), day(b){}
    void ShowDate();
    void IncreaseYear();};
void DayOfYear::ShowDate(){
    cout<< year << "년 " << month <<
    "월 " << day << "일" <<endl;}
void DayOfYear::IncreaseYear(){year++;}
int DayOfYear::year=2021;
int main(){
    DayOfYear christmas(12,25);
    christmas.ShowDate();
    DayOfYear today(10,27);
    today.ShowDate();

    christmas.IncreaseYear();
    christmas.ShowDate();
    today.ShowDate();
    return 0;}
```

```
2021년 12월 25일
2021년 10월 27일
2022년 12월 25일
2022년 10월 27일
```

```cpp
#include <iostream>
using namespace std;
class DayOfYear{
private:
    // static int year=2021;
    static int year;
    int month;
    int day;
public:
    DayOfYear(int a, int b):month(a), day(b){}
    void ShowDate();
    static void IncreaseYear(); };
void DayOfYear::ShowDate(){
    cout<< year << "년 " << month <<
    "월 " << day << "일" <<endl; }
void DayOfYear::IncreaseYear(){year++;}
int DayOfYear::year=2021;
int main(){
    DayOfYear christmas(12,25);
    christmas.ShowDate();
    DayOfYear today(10,27);
    today.ShowDate();

    DayOfYear::IncreaseYear();
    christmas.ShowDate();
    today.ShowDate();
    return 0; }
```

```
2021년 12월 25일
2021년 10월 27일
2022년 12월 25일
2022년 10월 27일
```

Assignment

- **Make company class company and use exercise 1 class(person)**
- **Class variables**
  - Name, operating(bool), numCustomer
- **Class functions**
  - company(string) is a constructor
  - void showNumCustomer() const prints the number of membership customer
  - static void offOperating() turns off operating status
  - static bool isOperating() returns operating status
  - Void signUp(person) increases numCustomer when the customer is adult and prints whether signed up or not

Assignment

```
class company{
private:
    string  name;
    static bool operating;
    int numCustomer = 0;
public:
    company( string );
    void showNumCustomer() const;
    static void offOperating();
    static bool isOperating();
    void signUp( person );
};
```

```
int main()
{
    company a( "cacao" );
    while( company::isOperating() )
    {
        cout<<"Enter your name and age."<<endl;
        string name; int age;
        cin>>name>>age;
        person customer( name,age );
        a.signUp( customer );
        a.showNumCustomer();

        cout<<"Is it still operating tile?"<<endl;
        char ans; cin>>ans;
        if( ans=='n'||ans=='N')
            company::offOperating();
        cout<<endl;
    }
    cout<<"Operating hours are over."<<endl;
    return 0;
}
```

```
Enter your name and age.
alpha 23
Signed up.
Number of VIP membership : 1
Is it still operating tile?
y

Enter your name and age.
beta 17
beta is not adult. Can not sign up.
Number of VIP membership : 1
Is it still operating tile?
y

Enter your name and age.
gamma 30
Signed up.
Number of VIP membership : 2
Is it still operating tile?
n

Operating hours are over.
```