Copyright Warning

ViSLAB
High-Performance Visualization, Communication, Computing

Object Oriented Design (creating/deleting objects)

Week 6-2

- timing of object creation/deletion
- local object (inside a function)
- global object (outside a function)
- static object (“static”)
- static members

object v.s general var

- variable with a class as its data-type
  - a copy of the class is created on the memory
  - == object.
- timing of creation and deletion is the same as a general variable

scope of variables

```
   // global variable
int gVar;

int foo() {
   // local variable
   int aVar;
   ...
   ...
   return 0;
}

int bar() {
   // local variable
   int bVar;
   ...
   ...
   return 0;
}
```

```
#include <iostream.h>

// global variable
int gVar = 987;

int foo() { // called from main()
   cout << "foo() is called." << endl;
   // local variable
   int aVar;
   cout << "aVar is : " << aVar << endl;
   aVar = 654;
   cout << "exits foo()." << endl;
   return 0;
}

int main() {
   cout << "main() started." << endl;
   cout << "gVar is : " << gVar << endl;
   foo();
   foo();
   cout << "main() ends." << endl;
   return 0;
}
```

Timing for Obj is the same

- an object created in a function can be used within the function
  - "Local Object"
- an object created outside of a function can be used from anywhere
  - "Global Object"

scope of objects

```
//Global Object
MyClass gObj;

int foo() {
   //local object
   MyClass aClass;
   ...
   ...
   return 0;
}

int bar() {
   //local object
   MyClass bClass;
   ...
   ...
   return 0;
}
```

```
#include <iostream.h>

class MyClass{
public:
    int myValue;
    MyClass() { cout << "MyClass() is called" << endl;}
    MyClass(int value):myValue(value) {cout <<"myClass(int) is called" << endl;};
    ~MyClass() {cout << "~MyClass() is called" << endl;};
};

//Global Object
MyClass gObj(987);

int foo() { // called from main()
    cout << "foo() is called." << endl;
    // local Object
    MyClass aObj;
    cout << "aObj.myValue is : " << aObj.myValue << endl;
    aObj.myValue = 654;
    cout << "exits foo()." << endl;
    return 0;
}

int main() {
    cout << "main() started." << endl;
    cout << "gObj.myValue is : " << gObj.myValue << endl;
    foo();
    foo();
    cout << "main() ends." << endl;
    return 0;
}
```

Static Variable

- when you use “static” keyword for a variable in a function
- memory is allocated at the program launch
- released at the end of the program
- scope is still within the function

Scope of Static Var

```
int foo() {
   static int aVar = 654;
   cout << "satic var = " << aVar << endl;
   aVar++;
   return 0;
}
```

Static Object

- when you use “static” keyword for an object in a function
- memory is allocated at the program launch
- released at the end of the program
- scope is still within the function

Scope of Static Var

```
int foo() {
   static MyClass aObj(654);
   cout << "satic obj.myValue = " << aObj.myValue << endl;
   aObj.myValue++;
   return 0;
}
```

```
#include <iostream.h>

class MyClass{
public:
    int myValue;
    MyClass():myValue(0) {cout << "MyClass() is called" << endl;}
    MyClass(int value):myValue(value) {cout <<"myClass(int) is called" << endl;};
    ~MyClass() {cout << "~MyClass() is called" << endl;};
};

int foo() { // called from main()
    cout << "foo() is called." << endl;
    // local Object
    static MyClass aObj(654);
    cout << "aObj.myValue is : " << aObj.myValue << endl;
    aObj.myValue++;
    cout << "exits foo()." << endl;
    return 0;
}

int main() {
    cout << "main() started." << endl;
    foo();
    foo();
    cout << "main() ends." << endl;
    return 0;
}
```

Static member variable

- is a member variable with “static” keyword.

in Java...

- In Java, a Class can have variables
  - "Class Variables"...."static" keyword.

```
class MyClass {
      static int MyValue;
      int realMyValue;
};

MyClass.MyValue = 100;
```

Class Variable in C++?

```
class MyClass {
public:
    static int MyValue;
    int realMyValue;
};

int MyClass::MyValue;

MyClass::MyValue = 100;
```

Where should I define?

in MyClass.h

```
class MyClass {
public:
     static int MyValue;
     int realMyValue;
};

int MyClass::MyValue;
```

Where should I define?

in MyClass.h

```
class MyClass {
public:
     static int MyValue;
     int realMyValue;
};

int MyClass::MyValue;
```

Where should I define?

in MyClass.cpp

```
#include “MyClass.h”

int MyClass::MyValue;
```

Use Static Member Var

- Counting instantiated objects

```
class MyClass{
public:
   static int MyClassObjCount;
   int myValue;
   MyClass():myValue(0) {MyClassObjCount ++;};
   MyClass(int value):myValue(value) {MyClassObjCount++;};
   ~MyClass() {cout << "~MyClass() is called" << endl;};
};
int MyClass::MyClassObjCount = 0;
```

Dynamic Object

- In C
  - malloc(), free()
- In C++
  - new, delete : a part of lang spec.

new

```
MyClass *ptr = new MyClass;
```

or

```
MyClass *ptr;
ptr = new MyClass;
```

or

```
MyClass *ptr = new MyClass(456);
```

or

```
MyClass *ptr;
ptr = new MyClass(456);
```

new

```
MyClass *ptr = new MyClass;

ptr->myValue = 456

cout << ptr->myValue << endl;
```

new...where is the object?

```
int foo() {
  MyClass *ptr = new MyClass;
  ptr->myValue = 456;
  ...
  return 0;
}
```

new...then delete

- When you don't need it you need to delete it.

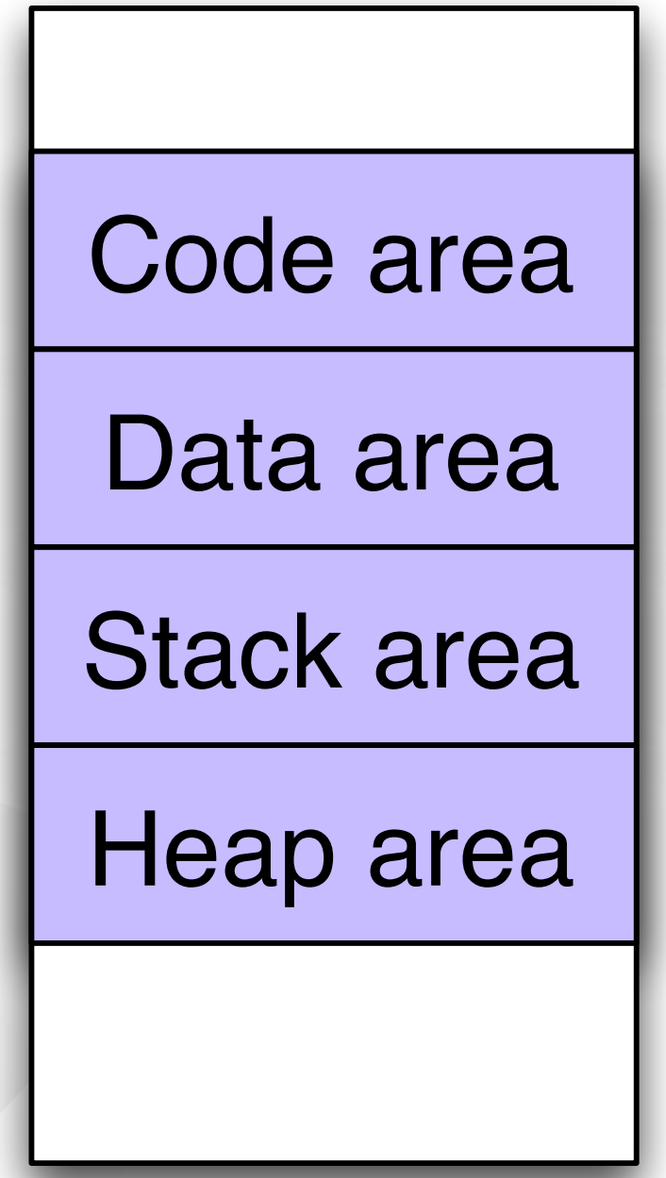
```
int foo() {
  MyClass *ptr = new MyClass;
  ptr->myValue = 456;
  ...
   delete ptr;
  return 0;
}
```

new...then delete

```
int foo() {
   MyClass *ptr = new MyClass;
   ptr->myValue = 456;
   ...
   delete ptr;
   return 0;
}
```

```
int foo() {
   MyClass myC;
   myC.myValue = 456;
   ...
   return 0;
}
```

A program



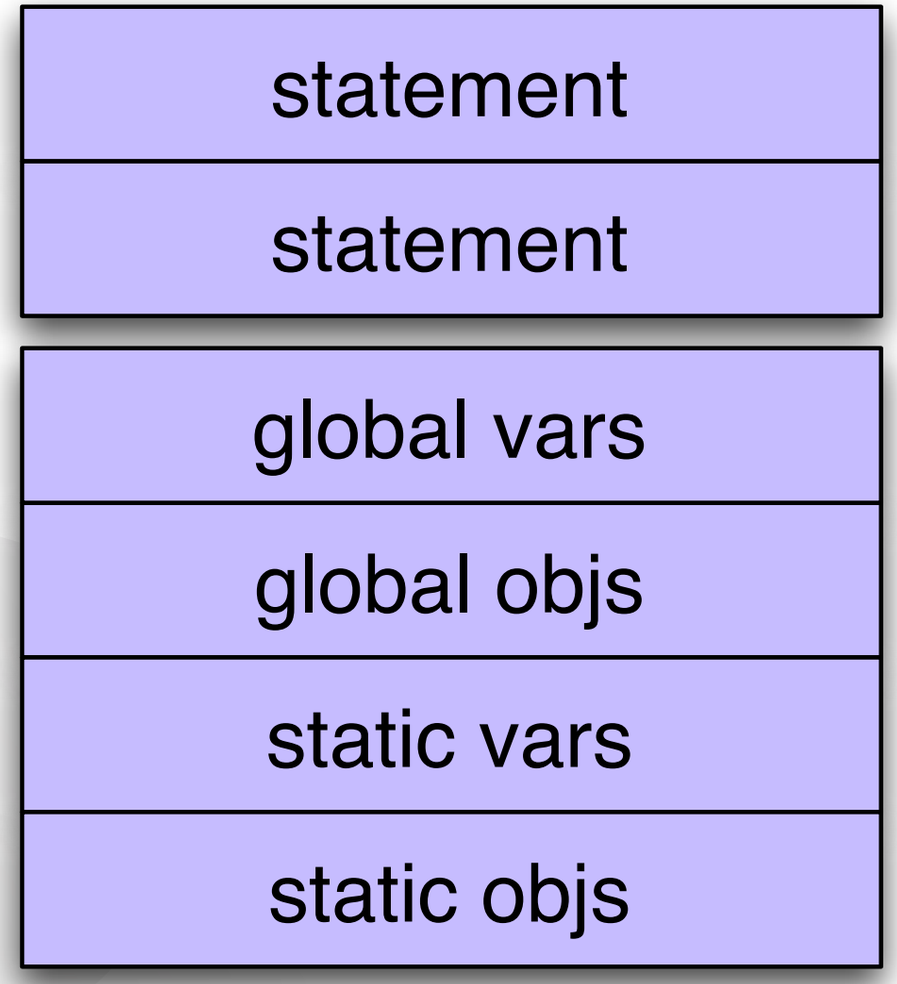
commands/statements

global variables/objects/static

args, local vars/objects

new-ed objects/vars

code/data area



commands/statements

global variables/objects/static

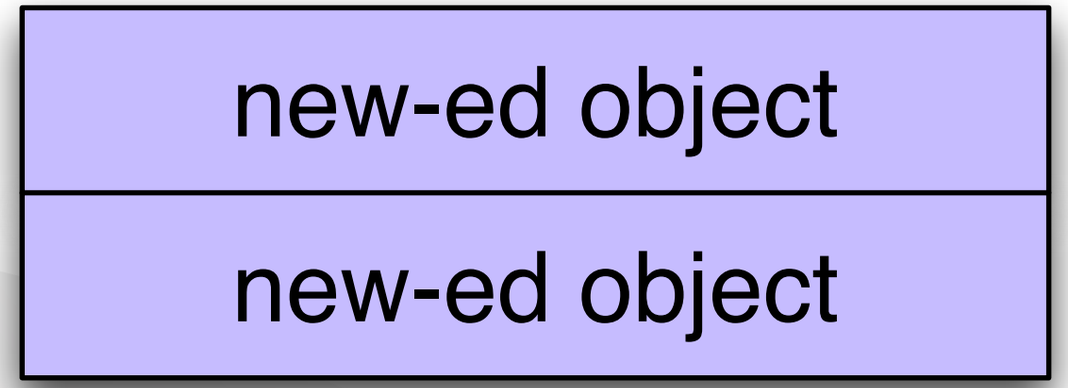
stack

- contents will dynamically change.

args, local vars/objects

heap

- contents will dynamically change.



new-ed objects/vars

extra... &

- usually see as “Pass by Reference”

```
int bar(int &x) {
    x = 10;
    return 0;
}
```

extra... &

- but you can use it anywhere...

```
int main() {
   int a = 10;
   int &b = a;
   b = 20;
   cout << "a = " << a << ", b = " << b << endl;
   return 0;
}
```

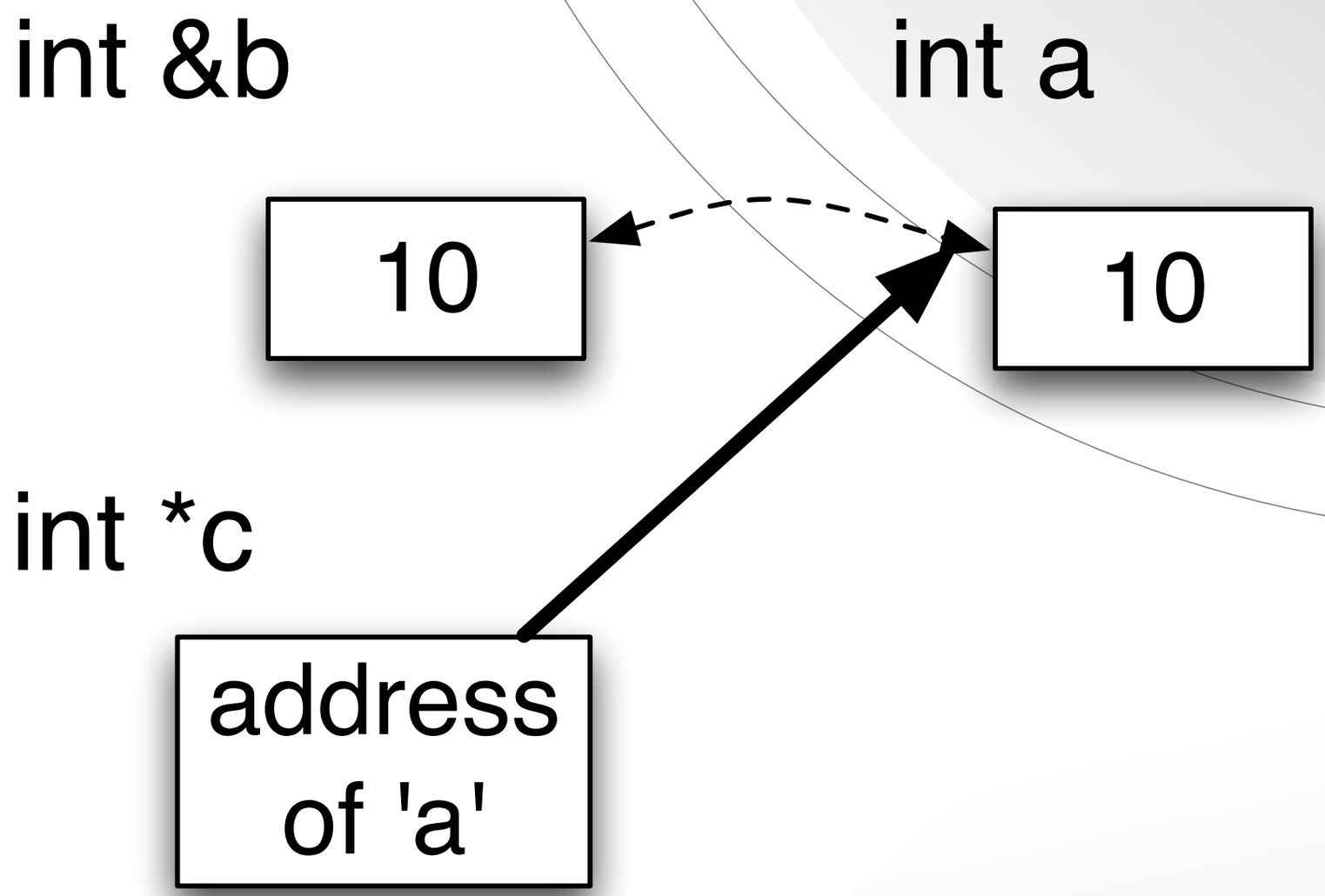
extra... &

- but you can use it anywhere...

```
int main() {
   int a = 10;
   int *b = &a;
   *b = 20;
   cout << "a = " << a << ", b = " << b << endl;
   return 0;
}
```

extra... &

```
int main() {
int a = 10;
int &b = a;
int *c = &a;
   b = 20;
   *c = 20;
   return 0;
}
```



extra... &

```
int main() {
int a = 10;
int &b = a;
int *c = &a;
b++;
c++
return 0;
}
```