

C/C++ Program Design

CS205

Prof. Shiqi Yu (于仕琪)

yusq@sustech.edu.cn

http://faculty.sustech.edu.cn/yusq/





Classes and Objects





Structures

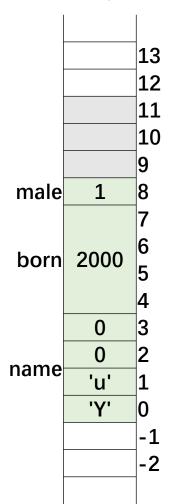
• A struct in C is a type consisting of a sequence of data members.

Some functions/statements are needed to operate the data members of

an object of a struct type.

```
struct Student
{
    char name[4];
    int born;
    bool male;
};

struct Student stu;
strcpy(stu.name, "Yu");
stu.born = 2000;
stu.male = true;
```







Classes

 You should be very careful to manipulated the data members in a struct object.

• Can we improve struct to a better one?

• Yes, it is class! We can put some member fun jons

```
firstclass.cpp
```

```
class Student
{
  public:
    char name[4];
  int born;
  bool male;
  void setName(const char * s)
  {
    strncpy(name, s, sizeof(name));
  }
  void setBorn(int b)
  { ...
```





Access Specifiers

- You can protect data members by access specifier private.
- Then data member can only be accessed by well designed member functions.

```
access-attribute.cpp
                                                    Student yu;
            class Student
                                                    yu.born = 2001;
             private:
              char name[4];
              int born;
              bool male;
             public:
              void setName(const char * s)
                strncpy(name, s, sizeof(name));
              void setBorn(int b)
```



void printInfo();

Member Functions

• A member function can be defined inside or outside class.

```
inline function inline void Student::setGender(bool isMale)
class Student
                                      male = isMale;
 private:
                                    void Student::printInfo()
  char name 4;
  int born;
                                      cout << "Name: " << name << endl;
  bool male;
                                      cout << "Born in " << born << endl;
 public:
                                      cout << "Gender: " << (male ? "Male" : "Female") << endl;
  void setName(const char * s)
    strncpy(name, s, sizeof(name));
  void setBorn(int b)
    born = b;
  void setGender(bool isMale);
```



File Structures

• The source code can be placed into multiple files

```
void Student::setGender(bool isMale)
                                                                                             student.cpp
student.hpp
class Student
                                       male = isMale;
 private:
                                     void Student::printInfo()
  char name[4];
  int born;
                                       cout << "Name: " << name << endl;
  bool male;
                                       cout << "Born in " << born << endl;
 public:
                                       cout << "Gender: " << (male ? "Male" : "Female") << endl;
  void setName(const char * s)
    strncpy(name, s, sizeof(name));
  void setBorn(int b)
    born = b;
  void setGender(bool isMale);
  void printInfo();
```



Constructors and Destructors





Constructors

- Different from struct in C, a constructor will be invoked when creating an object of a class.
 - struct in C: allocate memory
 - class in C++: allocate memory & invoke a constructor
- But ... No constructor is defined explicitly in previous examples.
 - The compiler will generate one with empty body





Constructors

```
class Student
                                     • The same name with the class.
 private:

    Have no return value

  // ...
 public:
  Student()
    name[0] = 0;
    born = 0;
    male = false;
  Student(const char * initName, int initBorn, bool isMale)
    setName(initName);
    born = initBorn;
    male = isMale;
```





Constructors

• The members can also be initialized as follows

```
Student(const char * initName): born(0), male(true)
{
   setName(initName);
}
```



Destructors

- The destructor will be invoked when the object is destroyed.
- Be formed from the class name preceded by a tilde (~)
- Have no return value, no parameters

```
class Student
  // ...
 public:
  Student()
    name = new char[1024]{0};
    born = 0;
    male = false;
    cout << "Constructor: Person()" << endl;</pre>
  ~Student()
    delete [] name;
               destructor.cpp
```





Destructors

What is the difference between the following two lines?

```
delete class1;
delete []class1;
```





this Pointer





Why is this needed?

How does a member function know which name?

```
Student yu = Student{"Yu", 2000, true};
Student amy = Student{"Amy", 2000, true};
yu.setName("yu");
amy.setName("Amy");
```

name: "Yu"

born: 2000

male: true

name: "Amy"

born: 2001

male: false

```
void setName(const char * s)
{
   strncpy(name, s, 1024);
}
```





this Pointer

- All methods in a function have a this pointer.
- It is set to the address of the object that invokes the method.

```
void setBorn(int b)
{
   born = b;
}
```

```
void setBorn(int b)
{
   this->born = b;
}
```

```
void setBorn(int born)
{
    this->born = born;
}
```





const and static Members





const Variables

• Statements for constants

```
#define VALUE 100

const int value = 100;
const int * p_int;
int const * p_int;
int * const p_int;

void func(const int *);
void func(const int &);
```





const Members

- const member variables behavior similar with normal const variables
- const member functions promise not to modify member variables.

```
class Student
 private
  const int BMI = 24;
 public:
  Student()
    BMI = 25;//can it be modified?
    // ...
  int getBorn() const
    born++; //Can it be modified?
    return born;
         const.cpp
```





static members

static members are not bound to class instances.

```
class Student
 private:
  static size_t student_total; // declaration only
 public:
  Student()
    student total++;
  ~Student()
    student_total--;
  static size_t getTotal() {return student_total;}
// definition it here
size_t Student::student_total = 0;
```

student_total: 0 3

name: "Yu" name: "Amy" born: 2000 born: 2001 male: true male: false

name: "Tom"
born: 2001
male: true



static.cpp