Classes

Week 2

{C++}

Yang-Cheng Chang
Yuan-Ze University
yczhang@saturn.yzu.edu.tw

Copy Constructors

- When copies of objects are made
  - A variable is declared which is initialized from another object

```
~~person q("Mickey");    // constructor is used to build q.~~
person r(p);            // copy constructor is used to build r.
person p = q;           // copy constructor is used to initialize in declaration.
```

  - A value parameter is initialized from its corresponding argument

```
class book{
        person author;
        book(person m):author(m){}      // copy constructor initializes formal value parameter.
};
```

Copy Constructors

```
class person {
      private:
            char* name;
            int age;
      public:
            person(char* nm, int ae);               // default constructor
            ~person();                              // default destructor
            person(person &m);                      // copy constructor
            person& operator=( person &m)           // assignment operator
};
```

Don't write a copy constructor if shallow copies are ok

- If the object has no pointers to dynamically allocated memory, a shallow copy is probably sufficient.
  - Therefore the default copy constructor, default assignment operator, and default destructor are ok and you don't need to write your own.

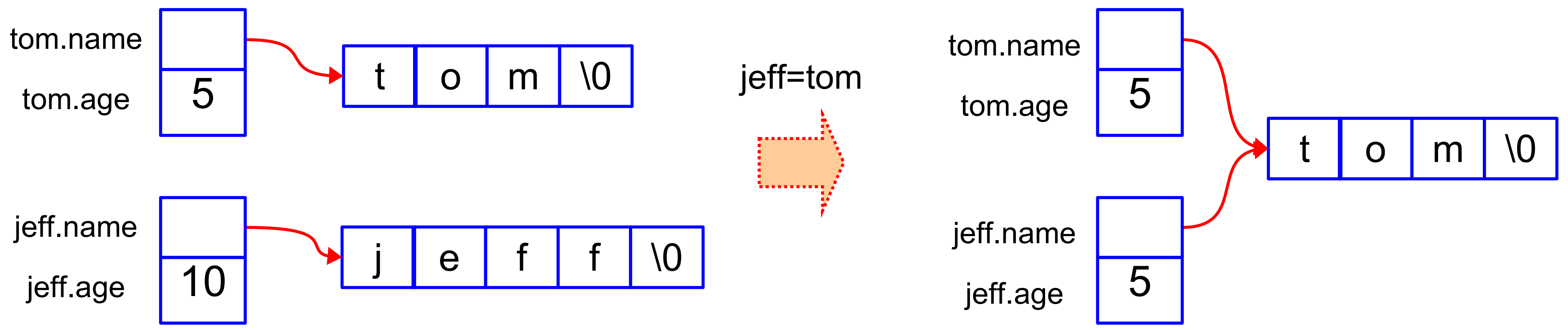
```
class person {
        private:
                string name;
                int age;
        public:
                person(string nm, int ae);              // default constructor
};
```

Shallow Copies

- A shallow copy of an object copies all of the member field values.
  - This works well if the fields are values, but ⋯..
- The default copy constructor and assignment operator make shallow copies.

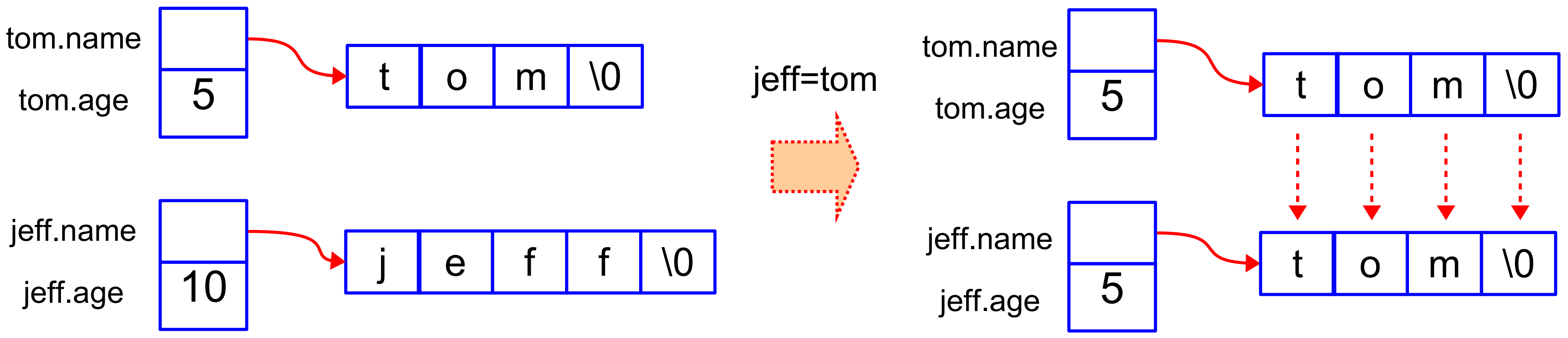
Shallow Copies

- The pointer will be copied. but the memory it points to will not be copied
  - The field in both the original object and the copy will then point to the same dynamically allocated memory



Deep Copies

- A deep copy copies all fields, and makes copies of dynamically allocated memory pointed to by the fields.



Deep copies need ...

- If an object has pointers to dynamically allocated memory, and the dynamically allocated memory needs to be copied

```
class person {
        private:
                char* name;
                int age;
        public:
                person(char* nm, int ae);                       // default constructor
};
// default constructor
person::person(char *nm, int ae){
        name = new char[strlen(nm) + 1];
        strcpy(name, nm);
        age = ae;
}
```

Deep copies need ...

- A class that requires deep copies generally needs
  - A constructor to either make an initial allocation or set the pointer to NULL.
  - A destructor to delete the dynamically allocated memory.
  - A copy constructor to make a copy of the dynamically allocated memory.
  - An overloaded assignment operator to make a copy of the dynamically allocated memory.

Assignment Operator =

```
class person {
        private:
                char* name;
                int age;
        public:
                person(char* nm, int ae);              // default constructor
                ~person();                             // default destructor
                person(person &m);                     // copy constructor
                person& operator=( person &m)  // assignment operator
};
```

Difference between copy constructor and assignment

- A copy constructor is used to initialize a newly declared variable from an existing variable

```
person r(p);            // copy constructor is used to build r.
person p = q;           // copy constructor is used to initialize in declaration.
```

- The assignment operator is used to change an existing instance to have the same values as the right-side value
  - The instance p has to be destroyed and re-initialized if it has internal dynamic memory.

```
p = q;                  // Assignment operator, no constructor or copy constructor.
```

Difference between copy constructor and assignment

- A copy constructor is simpler than assignment
  - No need to test to see if it is being initialized from itself
  - No need to clean up (eg, delete) an existing value
  - A reference to itself is not returned

```
person::person(person& m){
        ……………………………….
}
```

Copy constructor

```
person& person::operator=(person& m){
        ……………………………….
        return *this
}
```

Assignment operator

Rule

Don't write a copy constructor if the object has no pointers to dynamically allocated memory

If you need a copy constructor, you also need a destructor and assignment operator

Assignment 2

- Complete all the member functions of C++ class person
- You can download the declaration of class person

```
class person {
    private:
        // member variables
        char* name;
        int age;
    public:
        // constructor
        person( char* nm = "noname", int ae = 5 );
        // copy constructor
        person( person& m );
        // destructor
        ~person();
        // assignment operator
        person& operator=( person& m );
        // member functions
        void setName( char* nm );
        void setAge( int ae );
        string getName();
        int getAge();
};
```