

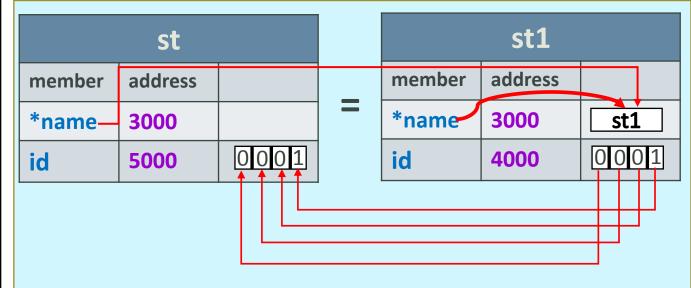
```
class student {
    int id;
    char* name;
public:
    student(char* p , int q) {
    id=q;
    name = new char[strlen(p)];
    strcpy(name,p);
    cout << "Constructing: "<<name<<endl;</pre>
    ~student(){
    cout << "Destructing "<<name<<endl;</pre>
    delete [] name;
    int getId() {return id;}
} ;
```

```
void func(student st) {
                                  OUTPUT??
cout << st.getId()<<endl;</pre>
                                 Constructing st1
int main() {
student st1("St1",1);
func(stl);
cout<<"Finish"<<endl;
                                    st1
                          member
                                  address
                          *name
                                  3000
                                            st1
                                           0001
                          id
                                  4000
```

```
class student {
    int id;
    char* name;
public:
    student(char* p , int q) {
    id=q;
    name = new char[strlen(p)];
    strcpy(name,p);
    cout << "Constructing: "<<name<<endl;</pre>
    ~student(){
    cout << "Destructing "<<name<<end1;</pre>
    delete [] name;
    int getId() {return id;}
};
```

```
void func(student st) {
cout << st.getId() << endl;
}

int main() {
student st1("St1",1);
func(st1);
cout << "Finish" << endl;
}</pre>
st = st1(bitwise copy)
```



```
class student {
    int id;
    char* name;
public:
    student(char* p , int q) {
    id=q;
    name = new char[strlen(p)];
    strcpy(name,p);
    cout << "Constructing: "<<name<<endl;</pre>
    ~student(){
    cout << "Destructing "<<name<<end1;</pre>
    delete [] name;
    int getId() {return id;}
} ;
```

```
void func(student st) {
cout << st.getId() << endl;
}

int main() {
student st1("St1",1);
func(st1);
cout << "Finish" << endl;
}</pre>
OUTPUT??
Constructing st1
1
```

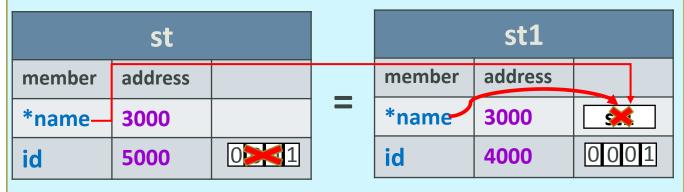
st				st1		
member	address			member	address	
*name—	3000		=	*name	3000	st1
id	5000	0001		id	4000	0001

```
class student {
    int id;
    char* name;
public:
    student(char* p , int q) {
    id=q;
    name = new char[strlen(p)];
    strcpy(name,p);
    cout << "Constructing: "<<name<<endl;</pre>
    ~student(){
    cout << "Destructing "<<name<<end1;</pre>
    delete [] name;
    int getId() {return id;}
} ;
```

```
void func(student(st)) {
cout << st.getId() << endl;
}

int main() {
student st1("St1",1);
func(st1);
cout<< "Finish" << endl;
}</pre>
OUTPUT??
Constructing st1
1
Destructing st1

int main() {
student st1("St1",1);
func(st1);
}
```



```
class student {
    int id;
    char* name;
public:
    student(char* p , int q) {
    id=q;
    name = new char[strlen(p)];
    strcpy(name,p);
    cout << "Constructing: "<<name<<endl;</pre>
    ~student(){
    cout << "Destructing "<<name<<end1;</pre>
    delete [] name;
    int getId() {return id;}
} ;
```

```
void func(student st) {
cout << st.getId() << endl;
}

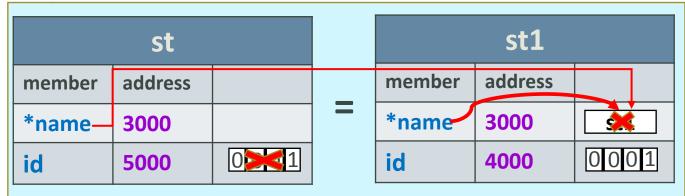
int main() {
student st1("St1",1);
func(st1);

cout << "Finish" << endl;

OUTPUT??

Constructing st1

1
Destructing st1
Finish</pre>
```



```
class student {
    int id;
    char* name;
public:
    student(char* p , int q) {
    id=q;
    name = new char[strlen(p)];
    strcpy(name,p);
    cout << "Constructing: "<<name<<endl;</pre>
    ~student(){
    cout << "Destructing "<<name<<end1;</pre>
    delete [] name;
    int getId() {return id;}
} ;
```

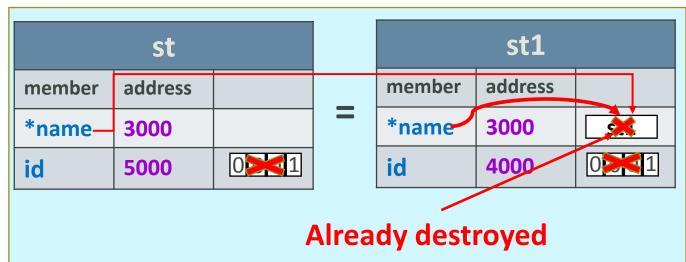
```
void func(student st) {
cout << st.getId() << endl;
}

int main() {
student st1("St1",1);
func(st1);
cout<< "Finish"<< endl;

OUTPUT??

Constructing st1

1
Destructing st1
Finish
Destructing st1</pre>
```



A New Solution – User Defined Copy Constructor

- The *default copy constructor* makes a bitwise copy.
- Copy Constructor invoked in 3 ways
 - When an object is used to initialize another in a declaration statement(student st2=st1;)
 - When an object is passed as a parameter to a function
 - When a temporary object is created for use as a return value by a function
- User defined copy constructor specifies exactly what occurs when a copy of an object is made

Copy Constructor Syntax

```
■Most Common Form –
        classname (const classname &obj){
        // body
Here obj is a reference to an object that is being used to initialize another object.
Copy Constructor of the student class –
student (const student &ob){
      id = obj.id;
      name = new char[strlen(obj.name)+1];
      strcpy(name,obj.name);
```

Copy Constructor Invocation

```
class student {
    int id;
    char* name;
public:
    student(char* p , int q) {
    id=q;
    name = new char[strlen(p)];
    strcpy(name,p);
    cout << "Constructing: "<<name<<endl;</pre>
    student(const student &obj) {
        id = obj.id;
        name = new char[strlen(obj.name)+1];
        strcpy(name, obj.name);
    ~student(){
    cout << "Destructing "<<name<<end1;</pre>
    delete [] name;
    int getId() {return id;}
};
```

```
void func(student st) {
cout << st.getId() << endl;
}
int main() {
    student st1("St1", 1):
    func(st1);
cout << "Finish" << endl;
}</pre>
```

Copy Constructor Invocation

- Student s1 = s2; // y explicitly initializing x
- func(s2); // y passed as a parameter
- s1 = func(s1); // y receiving a returned object

```
int main() {
  student st1("St1",1), st2("st2",2);
  student st3 = st1;
  st3 = func(st1);
  cout<<"Finish"<<endl;
}</pre>
Copy
Invocation

Copy
Invocation

Copy
Invocation

Preserved

Copy
Invocation

Copy
Invocation

Preserved

Copy
Invocation

Copy
Invocation

Copy
Invocation

Preserved

Preserved

Copy
Invocation

Copy
Invocation

Copy
Invocation

Preserved

Copy
Invocation

Copy
Invoc
```

Copy Constructor Invocation(initialization)

Copy Constructor
Invocation (passing as parameter)

Copy Constructor
Invocation(return from function)

```
class student {
    int id;
    char* name;
public:
    student(char* p , int q) {
    id=q;
    name = new char[strlen(p)];
    strcpy(name,p);
    cout << "Constructing: "<<name<<endl;</pre>
    student(const student &obj) {
        id = obj.id;
        name = new char[strlen(obj.name)+1];
        strcpy(name, obj.name);
        cout<<"Copy Constructor calling "<<name<<endl;</pre>
    ~student(){
    cout << "Destructing "<<name<<endl;|delete [] name;</pre>
    int getId() {return id;}
```

```
student func(student st) {
student temp("temp", 3);
return temp;
}

int main() { 1
student st1("St1", 1), st2("st2", 2);
student st3 = st1;
st3 = func(st1);
cout<<"Finish"<<endl;
}
1.Constructing: St1 (In main)</pre>
```

```
class student {
    int id;
    char* name;
public:
    student(char* p , int q) {
    id=q;
    name = new char[strlen(p)];
    strcpy(name,p);
    cout << "Constructing: "<<name<<endl;</pre>
    student(const student &obj) {
        id = obj.id;
        name = new char[strlen(obj.name)+1];
        strcpy(name, obj.name);
        cout<<"Copy Constructor calling "<<name<<end1; 2.Constructing: St2 (In main)</pre>
    ~student(){
    cout << "Destructing "<<name<<endl;|delete [] name;</pre>
    int getId() {return id;}
```

```
student func(student st) {
student temp("temp", 3);
return temp;
int main() \{(1)\}
student st1 ("St1",1), st2("st2",2);
student st3 = st1;
st3 = func(st1);
cout<<"Finish"<<endl;
1.Constructing: St1 (In main)
```

```
class student {
    int id;
    char* name;
public:
    student(char* p , int q) {
    id=q;
    name = new char[strlen(p)];
    strcpy(name,p);
    cout << "Constructing: "<<name<<endl;</pre>
    student(const student &obj) {
        id = obj.id;
        name = new char[strlen(obj.name)+1];
        strcpy(name, obj.name);
        cout<<"Copy Constructor calling "<<name<<end1; 2.Constructing: St2(In main)</pre>
    ~student(){
    cout << "Destructing "<<name<<endl;|delete [] name;</pre>
    int getId() {return id;}
```

```
student func(student st) {
student temp("temp", 3);
return temp;
int main() \{(1)\}
student st1 ("st1",1), st2("st2",2);
student st3 = st1;(3)
st3 = func(st1);
cout<<"Finish"<<endl;
1.Constructing: St1(In main)
3.Copy Constructor Calling St1
(In main)
```

```
class student {
    int id;
    char* name;
public:
    student(char* p , int q) {
    id=q;
    name = new char[strlen(p)];
    strcpy(name,p);
    cout << "Constructing: "<<name<<endl;</pre>
    student(const student &obj) {
        id = obj.id;
        name = new char[strlen(obj.name)+1];
        strcpy(name, obj.name);
        cout<<"Copy Constructor calling "<<name<<end1; 2.Constructing: St2(In main)</pre>
    ~student(){
    cout << "Destructing "<<name<<endl;|delete [] name;</pre>
    int getId() {return id;}
```

```
student func(student st
student temp("temp", 3);
return temp;
int main() \{(1)
student st1 ("St1", 1), st2("st2", 2);
student st3 = st1;(3)
st3 = func(st1);
cout<<"Finish"<<end]
1.Constructing: St1(In main)
3.Copy Constructor Calling St1
(In main)
4.Copy Constructor Calling St1
(In function parameter)
```

```
class student {
    int id;
    char* name;
public:
    student(char* p , int q) {
    id=q;
    name = new char[strlen(p)];
    strcpy(name,p);
    cout << "Constructing: "<<name<<endl;</pre>
    student(const student &obj) {
        id = obj.id;
        name = new char[strlen(obj.name)+1];
        strcpy(name, obj.name);
        cout<<"Copy Constructor calling "<<name<<end1; 2.Constructing: St2(In main)</pre>
    ~student(){
    cout << "Destructing "<<name<<endl;|delete [] name;</pre>
    int getId() {return id;}
```

```
student func(student st
student temp("temp",3);
return temp;
int main() {(1)
student st1 ("St1", 1), st2("st2", 2);
student st3 = st1;(3)
st3 = func(st1);
cout<<"Finish"<<end]
1.Constructing: St1(In main)
3.Copy Constructor Calling St1
l (In main)
4.Copy Constructor Calling St1
(In function parameter)
6.Constructing: temp(In local)
```

```
class student {
    int id;
    char* name;
public:
    student(char* p , int q) {
    id=q;
    name = new char[strlen(p)];
    strcpy(name,p);
    cout << "Constructing: "<<name<<endl;</pre>
    student(const student &obj) {
        id = obj.id;
        name = new char[strlen(obj.name)+1];
        strcpy(name, obj.name);
        cout<<"Copy Constructor calling "<<name<<endl</pre>
    ~student(){
    cout << "Destructing "<<name<<endl;|delete [] name;</pre>
    int getId() {return id;}
```

```
student func(student st
 student temp("temp",3);
 return temp; (7)
 int main() \{(1)
 student st1 ("St1", 1), st2("st2", 2);
 student st3 = st1;(3)
 st3 = Func(st1);
 cout<<"Finish"<<end.
1.Constructing: St1(In main)
2.Constructing: St2(In main)
3.Copy Constructor Calling St1
(In main)
4.Copy Constructor Calling St1
(In function parameter)
6.Constructing: temp(In local)
7.Copy Constructor Calling temp(in
local)
```

```
class student {
                               The copy is done before the
    int id;
                               called function exits, and copies
    char* name;
                               the then-existing local variable
public:
                               into the return value.
    student(char* p , int o
    id=q;
                               The called function has access to
    name = new char[strlen
                               the memory the return value will
    strcpy(name,p);
                               occupy, even though
                                                        that
    cout << "Constructing:</pre>
                               memory is not "in scope" when
                               the copy is being made, it's still
    student (const student
                               available.
         id = obj.id;
         name = new char[str
         strcpy (name, obj.name);
         cout<<"Copy Constructor calling "<<name<<endl</pre>
    ~student(){
    cout << "Destructing "<<name<<endl;|delete [] name;</pre>
    int getId() {return id;}
```

```
student func(student st
student temp("temp", 3); (6)
return temp; (7)
int main() \{(1)
student st1 ("st1",1), st2("st2",2);
student st3 = st1;(3)
st3 = func(st1);
 cout<<"Finish"<<end
1.Constructing: St1(In main)
2.Constructing: St2(In main)
3.Copy Constructor Calling St1
(In main)
4.Copy Constructor Calling St1
(In function parameter)
6.Constructing: temp(In local)
7.Copy Constructor Calling temp(in
local)
```

```
class student {
    int id;
    char* name;
public:
    student(char* p , int q) {
    id=q;
    name = new char[strlen(p)];
    strcpy(name,p);
    cout << "Constructing: "<<name<<endl;</pre>
    student(const student &obj) {
        id = obj.id;
        name = new char[strlen(obj.name)+1];
         strcpy(name, obj.name);
        cout<<"Copy Constructor calling "<<name<<endl</pre>
                                                             (In main)
    ~student(){
    cout << "Destructing "<<name<<endl;|delete [] name;</pre>
    int getId() {return id;}
                                                             temp(in local)
```

```
student func(student st
student temp("temp",3);
return temp; (7)
int main() \{(1)
 student st1 ("St1", 1), st2("st2", 2);
 student st3 = st1;(3)
 st3 = func(st1);
 cout<<"Finish"<<end
1.Constructing: St1(In main)
                             8. Destructing temp(local)
2.Constructing: St2(In main)
3.Copy Constructor Calling St1
4.Copy Constructor Calling St1
(In function parameter)
6.Constructing: temp(In local)
7.Copy Constructor Calling
```

```
class student {
    int id;
    char* name;
public:
    student(char* p , int q) {
    id=q;
    name = new char[strlen(p)];
    strcpy(name,p);
    cout << "Constructing: "<<name<<endl;</pre>
    student(const student &obj) {
        id = obj.id;
        name = new char[strlen(obj.name)+1];
        strcpy(name, obj.name);
        cout<<"Copy Constructor calling "<<name<<endl</pre>
    ~student(){
    cout << "Destructing "<<name<<endl;|delete [] name;</pre>
    int getId() {return id;}
```

```
student func(student st
 student temp("temp", 3); (6)
 return temp; (7)
int main() {(1)
 student st1 ("St1", 1), st2("st2", 2);
 student st3 = st1;(3)
 st3 = func(st1);
 cout<<"Finish"<<endl
1.Constructing: St1(In main)
                             8.Destructing temp(local)
                             9.Destructing St1(local)
2.Constructing: St2(In main)
3.Copy Constructor Calling St1
(In main)
4.Copy Constructor Calling St1
(In function parameter)
6.Constructing: temp(In local)
```

7.Copy Constructor Calling

temp(in local)

```
class student {
                                                               student func(student st
    int id;
                                                               student temp("temp", 3); 6
    char* name;
                                                               return temp; (7)
public:
    student(char* p , int q) {
    id=q;
                                                               int main() {(1)
    name = new char[strlen(p)];
                                                               student st1 ("St1", 1), st2("st2", 2);
    strcpy(name,p);
    cout << "Constructing: "<<name<<endl;</pre>
                                                               student st3 = st1;(3)
                                                           10 st3 = func(st1);
    student(const student &obj) {
                                                               cout<<"Finish"<<endl
         id = obj.id;
                                                               1. Constructing: St1(In main)
                                                                                          8. Destructing temp(local)
         name = new char[strlen(obj.name)+1];
         strcpy(name, obj.name);
                                                              2.Constructing: St2(In main)
                                                                                          9. Destructing St1(local)
         cout<<"Copy Constructor calling "<<name<<endl</pre>
                                                              3.Copy Constructor Calling St1
                                                                                          10. Destructing
                                                              (In main)
                                                                                          temp(return value)
    ~student(){
                                                              4.Copy Constructor Calling St1
    cout << "Destructing "<<name<<endl;delete [] name;</pre>
                                                              (In function parameter)
                                                              6.Constructing: temp(In local)
    int getId() {return id;}
                                                              7. Copy Constructor Calling
                                                              temp(in local)
```

```
class student {
                                                               student func (student st)
    int id;
                                                               student temp("temp", 3); 6
    char* name;
                                                               return temp; (7)
public:
    student(char* p , int q) {
    id=q;
                                                               int main() \{(1)
    name = new char[strlen(p)];
                                                              student st1 ("St1",1), st2("st2",2);
    strcpy(name,p);
    cout << "Constructing: "<<name<<endl;</pre>
                                                               student st3 = st1;(3)
                                                           10 st3 = func(st1);
    student(const student &obj) {
                                                               cout<<"Finish"<<endl
         id = obj.id;
                                                               1.Constructing: St1(In main)
                                                                                          8.Destructing temp(local)
         name = new char[strlen(obj.name)+1];
                                                               2.Constructing: St2(In main)
                                                                                          9. Destructing St1(local)
         strcpy(name, obj.name);
         cout<<"Copy Constructor calling "<<name<<endl</pre>
                                                               3.Copy Constructor Calling St1
                                                                                          10. Destructing
                                                               (In main)
                                                                                          temp(return value)
                                                               4.Copy Constructor Calling St1
    ~student(){
                                                                                          Finish
    cout << "Destructing "<<name<<endl;|delete [] name;</pre>
                                                               (In function parameter)
                                                                                          11. Destructing
                                                               6.Constructing: temp(In local)
                                                                                          temp(main)
    int getId() {return id;}
                                                               7. Copy Constructor Calling
                                                               temp(in local)
```

```
class student {
                                                             student func(student st
    int id;
                                                             student temp("temp", 3);
    char* name;
                                                             return temp;
public:
    student(char* p , int q) {
    id=q;
                                                             int main(){(1)
                                                             name = new char[strlen(p)];
    strcpy(name,p);
                                                             student *st3 = st1;
    cout << "Constructing: "<<name<<endl;</pre>
                                                             st3 = Func(st1);
    student(const student &obj) {
                                                             cout<<"Finish"<<end.
        id = obj.id;
                                                             1.Constructing: St1(In main)
                                                                                       8.Destructing temp(local)
        name = new char[strlen(obj.name)+1];
        strcpy(name, obj.name);
                                                             2.Constructing: St2(In main)
                                                                                       9. Destructing St1(local)
        cout<<"Copy Constructor calling "<<name<<endl</pre>
                                                             3. Copy Constructor Calling St1
                                                                                       10. Destructing
                                                             (In main)
                                                                                       temp(return value)
                                                             4.Copy Constructor Calling St1
    ~student(){
                                                                                       Finish
    cout << "Destructing "<<name<<endl;|delete [] name;</pre>
                                                            (In function parameter)
                                                                                       11. Destructing
                                                             6.Constructing: temp(In local)
                                                                                       temp(main)
    int getId() {return id;}
                                                             7. Copy Constructor Calling
                                                                                       12. Destructing St2(main)
                                                             temp(in local)
```

```
class student {
    int id;
    char* name;
public:
    student(char* p , int q) {
    id=q;
    name = new char[strlen(p)];
    strcpy(name,p);
    cout << "Constructing: "<<name<<endl;</pre>
    student(const student &obj) {
        id = obj.id;
        name = new char[strlen(obj.name)+1];
        strcpy(name, obj.name);
        cout<<"Copy Constructor calling "<<name<<endl</pre>
    ~student(){
    cout << "Destructing "<<name<<endl;|delete [] name;</pre>
    int getId() {return id;}
                                                            temp(in local)
```

```
student func student st
 student temp ("temp", 3); (6)
 return temp;
 int main() {(1)
int main() {(1)
    student st1("St1",1), st2("st2",2);
 student st3 = st1(3)
 st3 = func(st1);
 cout<<"Finish"<<end
                              8. Destructing temp(local)
 1.Constructing: St1(In main)
                              9. Destructing St1(local)
2.Constructing: St2(In main)
                              10. Destructing
3.Copy Constructor Calling St1
                              temp(return value)
(In main)
                              Finish
4.Copy Constructor Calling St1
                              11 Destructing temp(main)
(In function parameter)
                              12. Destructing St2(main)
6.Constructing: temp(In local)
                              13. Destructing St1(main)
7. Copy Constructor Calling
```

Class Member of Another Class

```
class date {
    int day;
    int mon;
    int year;
public:
    date() {day=0; mon=0; year=0; }
    date(int d, int m, int y) {
    day = d; mon = m; year = y;
    void setDay(int d) {day=d;}
    void setMon(int m) {mon=m;}
    void setYear(int y) {year=y;}
    int getDay() {return day;}
    int getMon() {return mon;}
    int getYear() {return year;}
```

```
class student {
    int id:
    date dob; //date of birth
public:
    student(int i, int d, int m, int y) {
    id = i;
    dob. setDay(d);
    dob. setMon(m);
    dob. setYear(y);
    int getId() {return id;}
    void printDob() {
    cout<<"day : "<<dob.getDay()<<endl;</pre>
    cout<<"Mon : "<<dob.getMon()<<endl;</pre>
    cout<<"Year : "<<dob.getYear()<<endl;</pre>
};
   int main(){
    student st(21,8,8,97);
    cout<<" ID is: "<<st.getId()<<endl;</pre>
    st.printDob();
```

hank you.