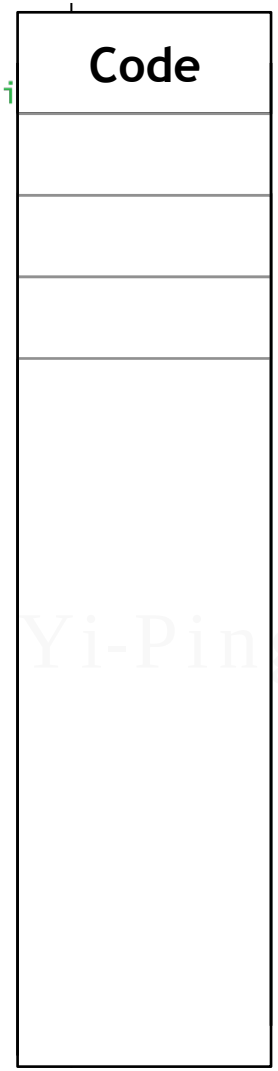
An Example of Constructors/Destructors Being Called

```
1  // Fig. 9.15: fig09_15.cpp
2  // Demonstrating the order in which constructors and
3  // destructors are called.
4  #include <iostream>
5  #include "CreateAndDestroy.h" // include CreateAndDestroy class definiti
6  using namespace std;
7
8  void create( void ); // prototype
9
10 CreateAndDestroy first( 1, "(global before main)" ); // global object
11
12 int main()
13 {
14    cout << "\nMAIN FUNCTION: EXECUTION BEGINS" << endl;
15    CreateAndDestroy second( 2, "(local automatic in main)" );
16    static CreateAndDestroy third( 3, "(local static in main)" );
17
18    create(); // call function to create objects
19
20    cout << "\nMAIN FUNCTION: EXECUTION RESUMES" << endl;
21    CreateAndDestroy fourth( 4, "(local automatic in main)" );
22    cout << "\nMAIN FUNCTION: EXECUTION ENDS" << endl;
23 } // end main
24
25 // function to create objects
26 void create( void )
27 {
28    cout << "\nCREATE FUNCTION: EXECUTION BEGINS" << endl;
29    CreateAndDestroy fifth( 5, "(local automatic in create)" );
30    static CreateAndDestroy sixth( 6, "(local static in create)" );
31    CreateAndDestroy seventh( 7, "(local automatic in create)" );
32    cout << "\nCREATE FUNCTION: EXECUTION ENDS" << endl;
33 } // end function create
```

An Example of Constructors/Destructors Being Called

1. Locations for static data are reserved

```
1  //
2  //
3  // destructors are called.
4  #include <iostream>
5  #include "CreateAndDestroy.h" // include CreateAndDestroy class definiti
6  using namespace std;
7
8  void create( void ); // prototype
9
10 CreateAndDestroy first( 1, "(global before main)" ); // global object
11
12 int main()
13 {
14    cout << "\nMAIN FUNCTION: EXECUTION BEGINS" << endl;
15    CreateAndDestroy second( 2, "(local automatic in main)" );
16    static CreateAndDestroy third( 3, "(local static in main)" );
17
18    create(); // call function to create objects
19
20    cout << "\nMAIN FUNCTION: EXECUTION RESUMES" << endl;
21    CreateAndDestroy fourth( 4, "(local automatic in main)" );
22    cout << "\nMAIN FUNCTION: EXECUTION ENDS" << endl;
23 } // end main
24
25 // function to create objects
26 void create( void )
27 {
28    cout << "\nCREATE FUNCTION: EXECUTION BEGINS" << endl;
29    CreateAndDestroy fifth( 5, "(local automatic in create)" );
30    static CreateAndDestroy sixth( 6, "(local static in create)" );
31    CreateAndDestroy seventh( 7, "(local automatic in create)" );
32    cout << "\nCREATE FUNCTION: EXECUTION ENDS" << endl;
33 } // end function create
```



An Example of Constructors/Destructors Being Called

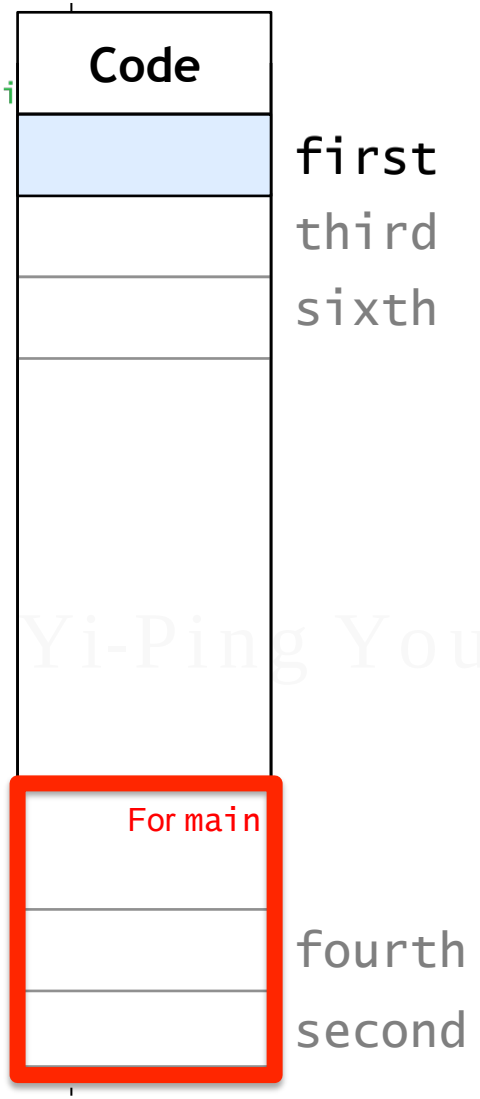
1. Locations for static data are reserved
2. Calling constructor for `first`

```
1  //
2  //
3  //
4  #include <iostream>
5  #include "CreateAndDestroy.h" // include CreateAndDestroy class definiti
6  using namespace std;
7
8  void create( void ); // prototype
9
10 CreateAndDestroy first( 1, "(global before main)" ); // global object
11
12 int main()
13 {
14    cout << "\nMAIN FUNCTION: EXECUTION BEGINS" << endl;
15    CreateAndDestroy second( 2, "(local automatic in main)" );
16    static CreateAndDestroy third( 3, "(local static in main)" );
17
18    create(); // call function to create objects
19
20    cout << "\nMAIN FUNCTION: EXECUTION RESUMES" << endl;
21    CreateAndDestroy fourth( 4, "(local automatic in main)" );
22    cout << "\nMAIN FUNCTION: EXECUTION ENDS" << endl;
23 } // end main
24
25 // function to create objects
26 void create( void )
27 {
28    cout << "\nCREATE FUNCTION: EXECUTION BEGINS" << endl;
29    CreateAndDestroy fifth( 5, "(local automatic in create)" );
30    static CreateAndDestroy sixth( 6, "(local static in create)" );
31    CreateAndDestroy seventh( 7, "(local automatic in create)" );
32    cout << "\nCREATE FUNCTION: EXECUTION ENDS" << endl;
33 } // end function create
```

An Example of Constructors/Destructors Being Called

1. Locations for static data are reserved
2. Calling constructor for `first`
3. Pushing `main`'s ARI to stack (locations for local variables and parameters are reserved)

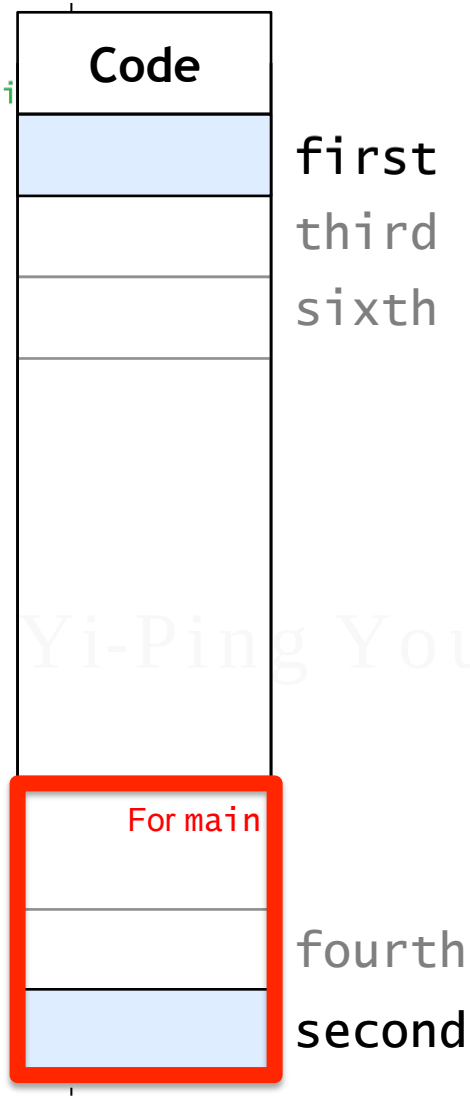
```
1
2
3
4
5
6
7
8   void create( void ); // prototype
9
10  CreateAndDestroy first( 1, "(global before main)" ); // global object
11
12  int main()
13  {
14     cout << "\nMAIN FUNCTION: EXECUTION BEGINS" << endl;
15     CreateAndDestroy second( 2, "(local automatic in main)" );
16     static CreateAndDestroy third( 3, "(local static in main)" );
17
18     create(); // call function to create objects
19
20     cout << "\nMAIN FUNCTION: EXECUTION RESUMES" << endl;
21     CreateAndDestroy fourth( 4, "(local automatic in main)" );
22     cout << "\nMAIN FUNCTION: EXECUTION ENDS" << endl;
23  } // end main
24
25  // function to create objects
26  void create( void )
27  {
28     cout << "\nCREATE FUNCTION: EXECUTION BEGINS" << endl;
29     CreateAndDestroy fifth( 5, "(local automatic in create)" );
30     static CreateAndDestroy sixth( 6, "(local static in create)" );
31     CreateAndDestroy seventh( 7, "(local automatic in create)" );
32     cout << "\nCREATE FUNCTION: EXECUTION ENDS" << endl;
33  } // end function create
```



An Example of Constructors/Destructors Being Called

1. Locations for static data are reserved
2. Calling constructor for `first`
3. Pushing `main`'s ARI to stack (locations for local variables and parameters are reserved)
4. Calling constructor for `second`

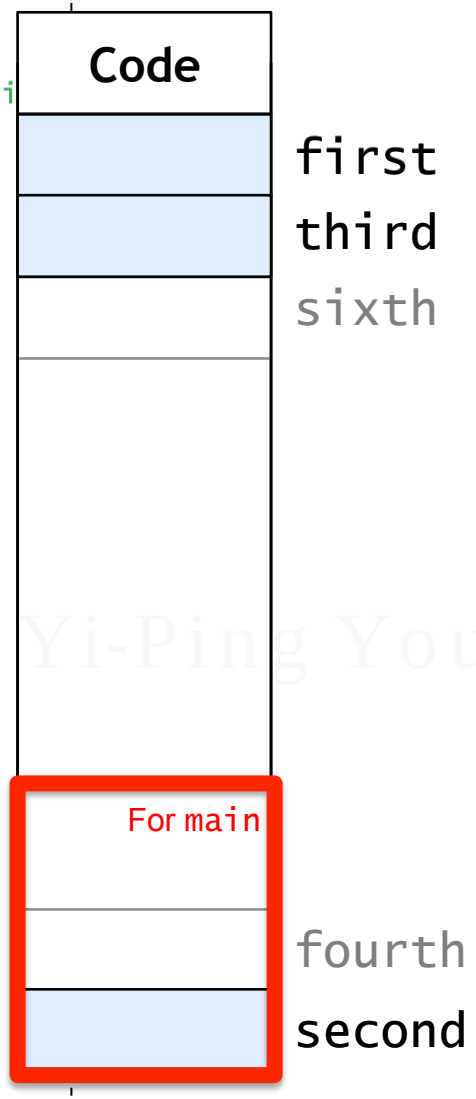
```
1
2
3
4
5
6
7
8  void create( void ); // prototype
9
10 CreateAndDestroy first( 1, "(global before main)" ); // global object
11
12 int main()
13 {
14    cout << "\nMAIN FUNCTION: EXECUTION BEGINS" << endl;
15    CreateAndDestroy second( 2, "(local automatic in main)" );
16    static CreateAndDestroy third( 3, "(local static in main)" );
17
18    create(); // call function to create objects
19
20    cout << "\nMAIN FUNCTION: EXECUTION RESUMES" << endl;
21    CreateAndDestroy fourth( 4, "(local automatic in main)" );
22    cout << "\nMAIN FUNCTION: EXECUTION ENDS" << endl;
23 } // end main
24
25 // function to create objects
26 void create( void )
27 {
28    cout << "\nCREATE FUNCTION: EXECUTION BEGINS" << endl;
29    CreateAndDestroy fifth( 5, "(local automatic in create)" );
30    static CreateAndDestroy sixth( 6, "(local static in create)" );
31    CreateAndDestroy seventh( 7, "(local automatic in create)" );
32    cout << "\nCREATE FUNCTION: EXECUTION ENDS" << endl;
33 } // end function create
```



An Example of Constructors/Destructors Being Called

1. Locations for static data are reserved
2. Calling constructor for `first`
3. Pushing `main`'s ARI to stack (locations for local variables and parameters are reserved)
4. Calling constructor for `second`
5. Calling constructor for `third`

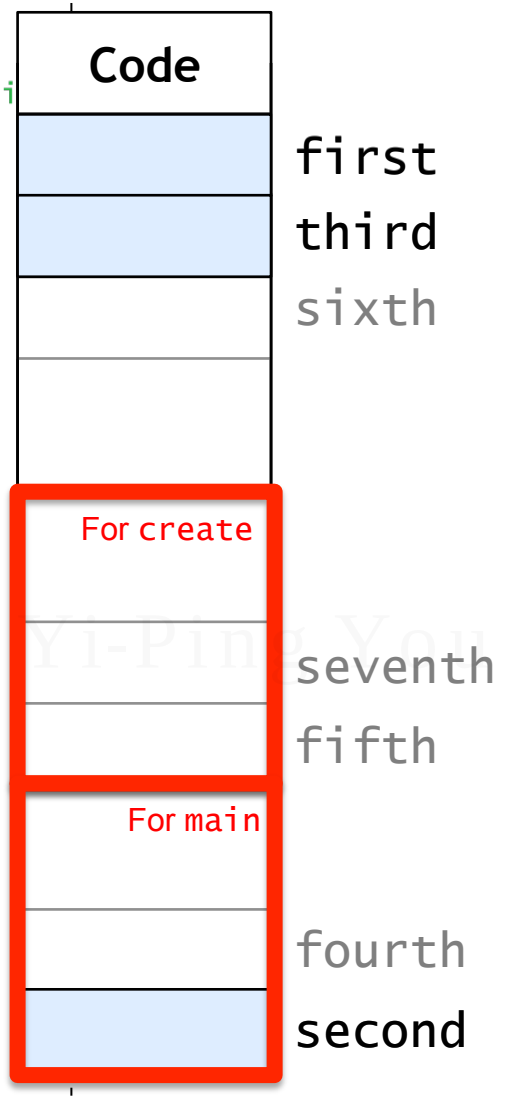
```
10  CreateAndDestroy first( 1, "(global before main)" ); // global object
11
12  int main()
13  {
14     cout << "\nMAIN FUNCTION: EXECUTION BEGINS" << endl;
15     CreateAndDestroy second( 2, "(local automatic in main)" );
16     static CreateAndDestroy third( 3, "(local static in main)" );
17
18     create(); // call function to create objects
19
20     cout << "\nMAIN FUNCTION: EXECUTION RESUMES" << endl;
21     CreateAndDestroy fourth( 4, "(local automatic in main)" );
22     cout << "\nMAIN FUNCTION: EXECUTION ENDS" << endl;
23  } // end main
24
25  // function to create objects
26  void create( void )
27  {
28     cout << "\nCREATE FUNCTION: EXECUTION BEGINS" << endl;
29     CreateAndDestroy fifth( 5, "(local automatic in create)" );
30     static CreateAndDestroy sixth( 6, "(local static in create)" );
31     CreateAndDestroy seventh( 7, "(local automatic in create)" );
32     cout << "\nCREATE FUNCTION: EXECUTION ENDS" << endl;
33  } // end function create
```



An Example of Constructors/Destructors Being Called

1. Locations for static data are reserved
2. Calling constructor for `first`
3. Pushing `main`'s ARI to stack (locations for local variables and parameters are reserved)
4. Calling constructor for `second`
5. Calling constructor for `third`
6. Pushing `create`'s ARI to stack (locations for local variables and parameters are reserved)

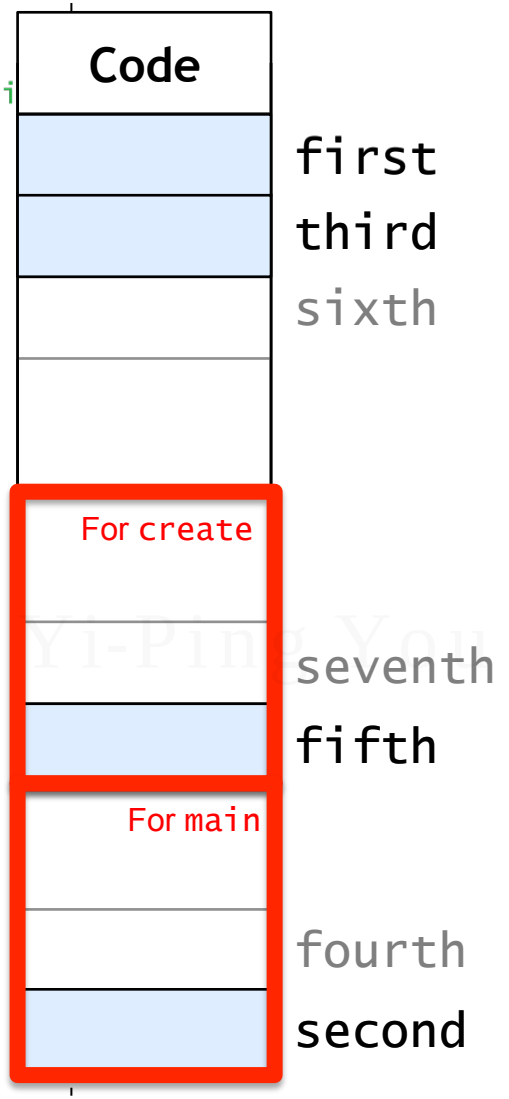
```
12 int main()
13 {
14    cout << "\nMAIN FUNCTION: EXECUTION BEGINS" << endl;
15    CreateAndDestroy second( 2, "(local automatic in main)" );
16    static CreateAndDestroy third( 3, "(local static in main)" );
17
18    create(); // call function to create objects
19
20    cout << "\nMAIN FUNCTION: EXECUTION RESUMES" << endl;
21    CreateAndDestroy fourth( 4, "(local automatic in main)" );
22    cout << "\nMAIN FUNCTION: EXECUTION ENDS" << endl;
23 } // end main
24
25 // function to create objects
26 void create( void )
27 {
28    cout << "\nCREATE FUNCTION: EXECUTION BEGINS" << endl;
29    CreateAndDestroy fifth( 5, "(local automatic in create)" );
30    static CreateAndDestroy sixth( 6, "(local static in create)" );
31    CreateAndDestroy seventh( 7, "(local automatic in create)" );
32    cout << "\nCREATE FUNCTION: EXECUTION ENDS" << endl;
33 } // end function create
```



An Example of Constructors/Destructors Being Called

1. Locations for static data are reserved
2. Calling constructor for `first`
3. Pushing `main`'s ARI to stack (locations for local variables and parameters are reserved)
4. Calling constructor for `second`
5. Calling constructor for `third`
6. Pushing `create`'s ARI to stack (locations for local variables and parameters are reserved)
7. Calling constructor for `fifth`

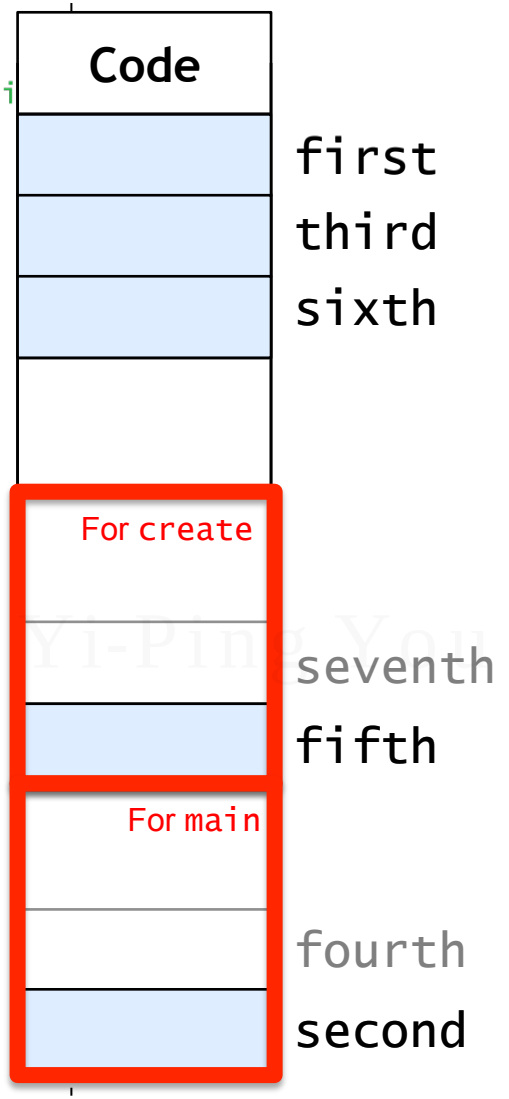
```
{
   cout << "\nMAIN FUNCTION: EXECUTION BEGINS" << endl;
   CreateAndDestroy second( 2, "(local automatic in main)" );
   static CreateAndDestroy third( 3, "(local static in main)" );

   create(); // call function to create objects

   cout << "\nMAIN FUNCTION: EXECUTION RESUMES" << endl;
   CreateAndDestroy fourth( 4, "(local automatic in main)" );
   cout << "\nMAIN FUNCTION: EXECUTION ENDS" << endl;
} // end main

// function to create objects
void create( void )
{
   cout << "\nCREATE FUNCTION: EXECUTION BEGINS" << endl;
   CreateAndDestroy fifth( 5, "(local automatic in create)" );
   static CreateAndDestroy sixth( 6, "(local static in create)" );
   CreateAndDestroy seventh( 7, "(local automatic in create)" );
   cout << "\nCREATE FUNCTION: EXECUTION ENDS" << endl;
} // end function create
```



An Example of Constructors/Destructors Being Called

1. Locations for static data are reserved
2. Calling constructor for `first`
3. Pushing `main`'s ARI to stack (locations for local variables and parameters are reserved)
4. Calling constructor for `second`
5. Calling constructor for `third`
6. Pushing `create`'s ARI to stack (locations for local variables and parameters are reserved)
7. Calling constructor for `fifth`
8. Calling constructor for `sixth`

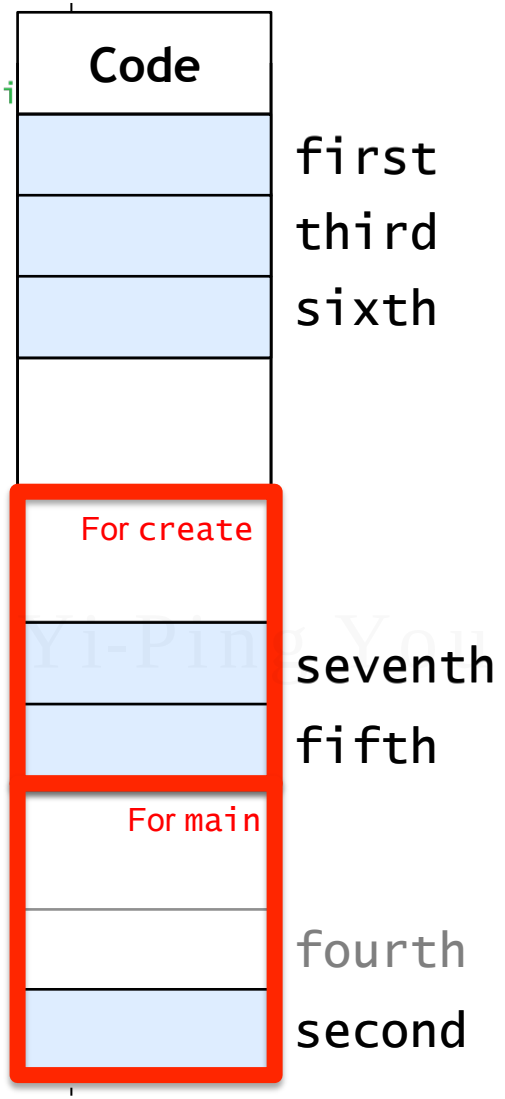
```
14     cout << "\nMAIN FUNCTION: EXECUTION BEGINS" << endl;
15     CreateAndDestroy second( 2, "(local automatic in main)" );
16     static CreateAndDestroy third( 3, "(local static in main)" );
17
18     create(); // call function to create objects
19
20     cout << "\nMAIN FUNCTION: EXECUTION RESUMES" << endl;
21     CreateAndDestroy fourth( 4, "(local automatic in main)" );
22     cout << "\nMAIN FUNCTION: EXECUTION ENDS" << endl;
23  } // end main
24
25  // function to create objects
26  void create( void )
27  {
28     cout << "\nCREATE FUNCTION: EXECUTION BEGINS" << endl;
29     CreateAndDestroy fifth( 5, "(local automatic in create)" );
30     static CreateAndDestroy sixth( 6, "(local static in create)" );
31     CreateAndDestroy seventh( 7, "(local automatic in create)" );
32     cout << "\nCREATE FUNCTION: EXECUTION ENDS" << endl;
33  } // end function create
```



An Example of Constructors/Destructors Being Called

1. Locations for static data are reserved
2. Calling constructor for `first`
3. Pushing `main`'s ARI to stack (locations for local variables and parameters are reserved)
4. Calling constructor for `second`
5. Calling constructor for `third`
6. Pushing `create`'s ARI to stack (locations for local variables and parameters are reserved)
7. Calling constructor for `fifth`
8. Calling constructor for `sixth`
9. Calling constructor for `seventh`

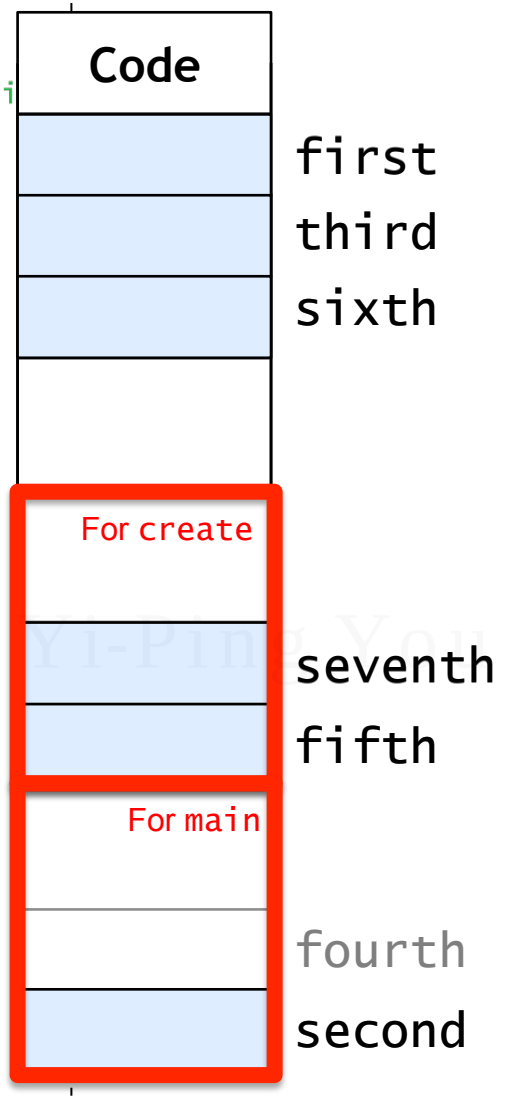
```
15     CreateAndDestroy second( 2, "(local automatic in main)" );
16     static CreateAndDestroy third( 3, "(local static in main)" );
17
18     create(); // call function to create objects
19
20     cout << "\nMAIN FUNCTION: EXECUTION RESUMES" << endl;
21     CreateAndDestroy fourth( 4, "(local automatic in main)" );
22     cout << "\nMAIN FUNCTION: EXECUTION ENDS" << endl;
23  } // end main
24
25  // function to create objects
26  void create( void )
27  {
28     cout << "\nCREATE FUNCTION: EXECUTION BEGINS" << endl;
29     CreateAndDestroy fifth( 5, "(local automatic in create)" );
30     static CreateAndDestroy sixth( 6, "(local static in create)" );
31     CreateAndDestroy seventh( 7, "(local automatic in create)" );
32     cout << "\nCREATE FUNCTION: EXECUTION ENDS" << endl;
33  } // end function create
```



An Example of Constructors/Destructors Being Called

1. Locations for static data are reserved
2. Calling constructor for `first`
3. Pushing `main`'s ARI to stack (locations for local variables and parameters are reserved)
4. Calling constructor for `second`
5. Calling constructor for `third`
6. Pushing `create`'s ARI to stack (locations for local variables and parameters are reserved)
7. Calling constructor for `fifth`
8. Calling constructor for `sixth`
9. Calling constructor for `seventh`
10. Calling destructor for `seventh`
11. Calling destructor for `fifth`

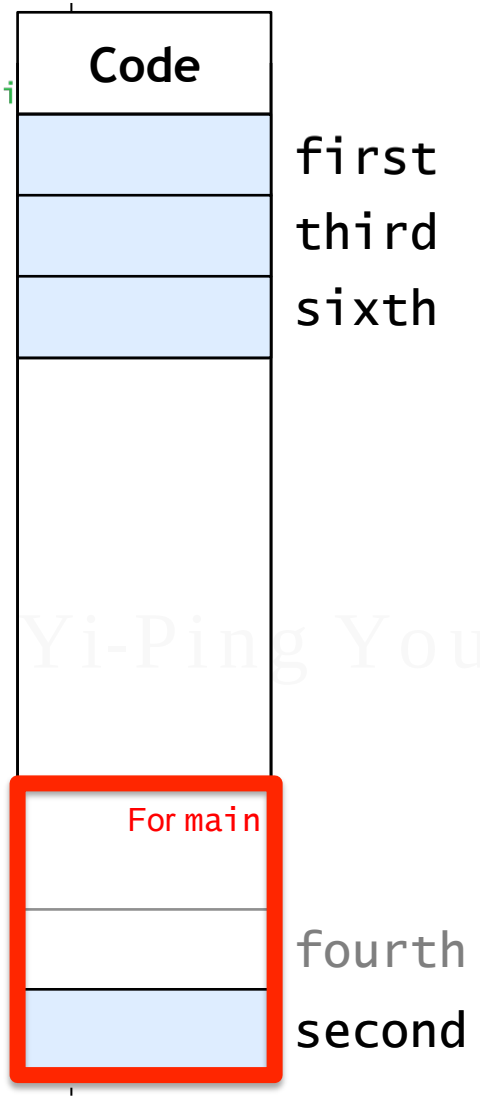
```
18     create(); // call function to create objects
19
20     cout << "\nMAIN FUNCTION: EXECUTION RESUMES" << endl;
21     CreateAndDestroy fourth( 4, "(local automatic in main)" );
22     cout << "\nMAIN FUNCTION: EXECUTION ENDS" << endl;
23  } // end main
24
25  // function to create objects
26  void create( void )
27  {
28     cout << "\nCREATE FUNCTION: EXECUTION BEGINS" << endl;
29     CreateAndDestroy fifth( 5, "(local automatic in create)" );
30     static CreateAndDestroy sixth( 6, "(local static in create)" );
31     CreateAndDestroy seventh( 7, "(local automatic in create)" );
32     cout << "\nCREATE FUNCTION: EXECUTION ENDS" << endl;
33  } // end function create
```



An Example of Constructors/Destructors Being Called

1. Locations for static data are reserved
2. Calling constructor for `first`
3. Pushing `main`'s ARI to stack (locations for local variables and parameters are reserved)
4. Calling constructor for `second`
5. Calling constructor for `third`
6. Pushing `create`'s ARI to stack (locations for local variables and parameters are reserved)
7. Calling constructor for `fifth`
8. Calling constructor for `sixth`
9. Calling constructor for `seventh`
10. Calling destructor for `seventh`
11. Calling destructor for `fifth`
12. Popping `create`'s ARI from stack

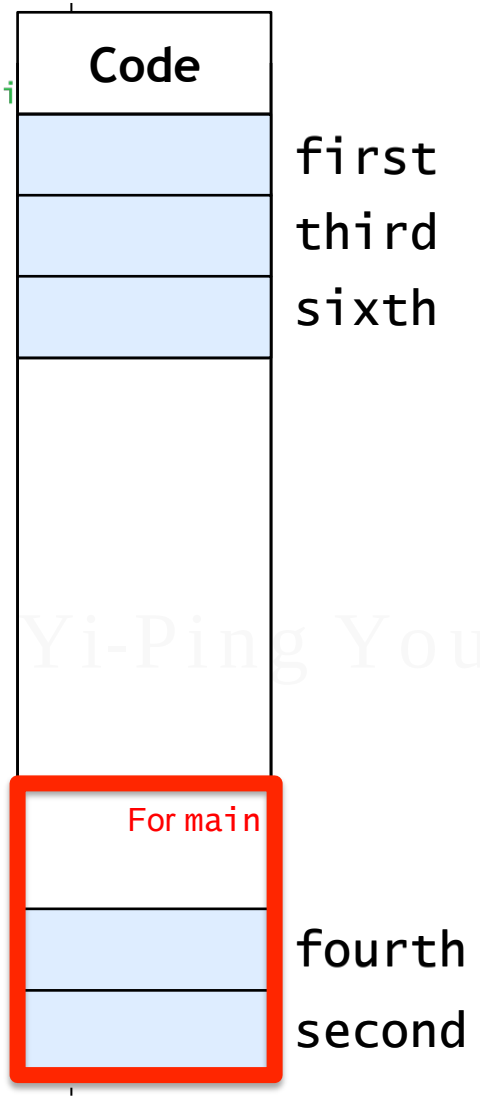
```
   cout << "\nMAIN FUNCTION: EXECUTION RESUMES" << endl;
   CreateAndDestroy fourth( 4, "(local automatic in main)" );
   cout << "\nMAIN FUNCTION: EXECUTION ENDS" << endl;
} // end main

// function to create objects
void create( void )
{
   cout << "\nCREATE FUNCTION: EXECUTION BEGINS" << endl;
   CreateAndDestroy fifth( 5, "(local automatic in create)" );
   static CreateAndDestroy sixth( 6, "(local static in create)" );
   CreateAndDestroy seventh( 7, "(local automatic in create)" );
   cout << "\nCREATE FUNCTION: EXECUTION ENDS" << endl;
} // end function create
```



An Example of Constructors/Destructors Being Called

1. Locations for static data are reserved
2. Calling constructor for `first`
3. Pushing `main`'s ARI to stack (locations for local variables and parameters are reserved)
4. Calling constructor for `second`
5. Calling constructor for `third`
6. Pushing `create`'s ARI to stack (locations for local variables and parameters are reserved)
7. Calling constructor for `fifth`
8. Calling constructor for `sixth`
9. Calling constructor for `seventh`
10. Calling destructor for `seventh`
11. Calling destructor for `fifth`
12. Popping `create`'s ARI from stack
13. Calling constructor for `fourth`

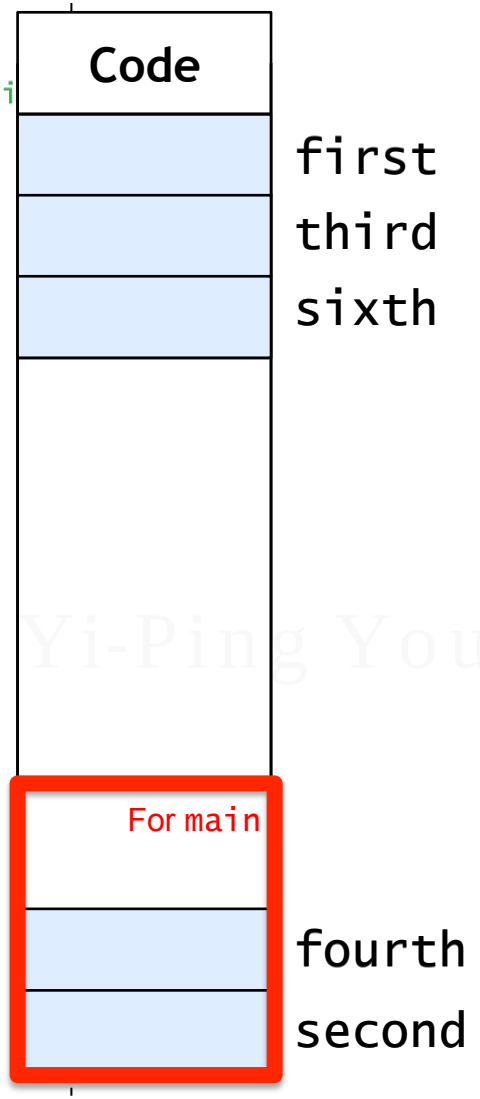
```
20    cout << "\nMAIN FUNCTION: EXECUTION RESUMES" << endl;
21    CreateAndDestroy fourth( 4, "(local automatic in main)" );
22    cout << "\nMAIN FUNCTION: EXECUTION ENDS" << endl;
23 } // end main
24
25 // function to create objects
26 void create( void )
27 {
28    cout << "\nCREATE FUNCTION: EXECUTION BEGINS" << endl;
29    CreateAndDestroy fifth( 5, "(local automatic in create)" );
30    static CreateAndDestroy sixth( 6, "(local static in create)" );
31    CreateAndDestroy seventh( 7, "(local automatic in create)" );
32    cout << "\nCREATE FUNCTION: EXECUTION ENDS" << endl;
33 } // end function create
```



An Example of Constructors/Destructors Being Called

1. Locations for static data are reserved
2. Calling constructor for `first`
3. Pushing `main`'s ARI to stack (locations for local variables and parameters are reserved)
4. Calling constructor for `second`
5. Calling constructor for `third`
6. Pushing `create`'s ARI to stack (locations for local variables and parameters are reserved)
7. Calling constructor for `fifth`
8. Calling constructor for `sixth`
9. Calling constructor for `seventh`
10. Calling destructor for `seventh`
11. Calling destructor for `fifth`
12. Popping `create`'s ARI from stack
13. Calling constructor for `fourth`
14. Calling destructor for `fourth`
15. Calling destructor for `second`

```
} // end main

// function to create objects
void create( void )
{
   cout << "\nCREATE FUNCTION: EXECUTION BEGINS" << endl;
   CreateAndDestroy fifth( 5, "(local automatic in create)" );
   static CreateAndDestroy sixth( 6, "(local static in create)" );
   CreateAndDestroy seventh( 7, "(local automatic in create)" );
   cout << "\nCREATE FUNCTION: EXECUTION ENDS" << endl;
} // end function create
```



An Example of Constructors/Destructors Being Called

1. Locations for static data are reserved
2. Calling constructor for `first`
3. Pushing `main`'s ARI to stack (locations for local variables and parameters are reserved)
4. Calling constructor for `second`
5. Calling constructor for `third`
6. Pushing `create`'s ARI to stack (locations for local variables and parameters are reserved)
7. Calling constructor for `fifth`
8. Calling constructor for `sixth`
9. Calling constructor for `seventh`
10. Calling destructor for `seventh`
11. Calling destructor for `fifth`
12. Popping `create`'s ARI from stack
13. Calling constructor for `fourth`
14. Calling destructor for `fourth`
15. Calling destructor for `second`
16. Popping `main`'s ARI from stack


```
// function to create objects
void create( void )
{
   cout << "\nCREATE FUNCTION: EXECUTION BEGINS" << endl;
   CreateAndDestroy fifth( 5, "(local automatic in create)" );
   static CreateAndDestroy sixth( 6, "(local static in create)" );
   CreateAndDestroy seventh( 7, "(local automatic in create)" );
   cout << "\nCREATE FUNCTION: EXECUTION ENDS" << endl;
} // end function create
```

An Example of Constructors/Destructors Being Called

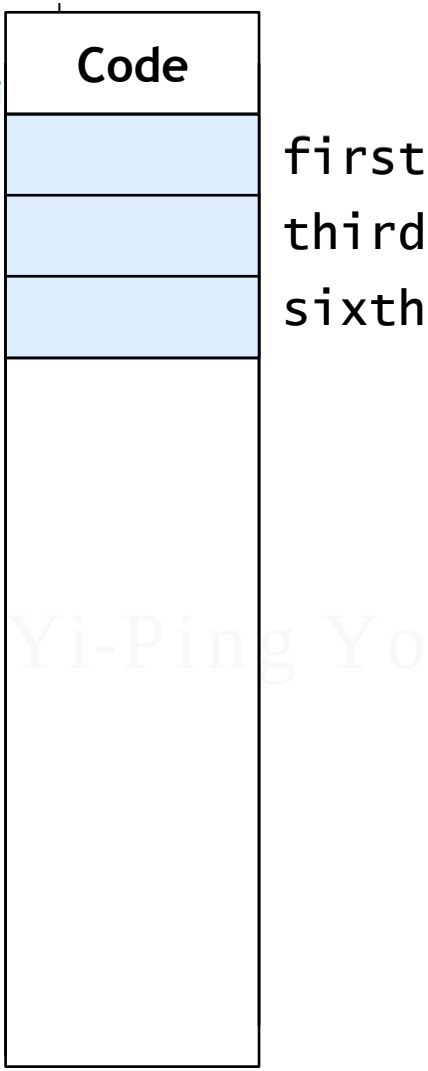
1. Locations for static data are reserved
2. Calling constructor for `first`
3. Pushing `main`'s ARI to stack (locations for local variables and parameters are reserved)
4. Calling constructor for `second`
5. Calling constructor for `third`
6. Pushing `create`'s ARI to stack (locations for local variables and parameters are reserved)
7. Calling constructor for `fifth`
8. Calling constructor for `sixth`
9. Calling constructor for `seventh`
10. Calling destructor for `seventh`
11. Calling destructor for `fifth`
12. Popping `create`'s ARI from stack
13. Calling constructor for `fourth`
14. Calling destructor for `fourth`
15. Calling destructor for `second`
16. Popping `main`'s ARI from stack
17. Calling destructor for `sixth`
18. Calling destructor for `third`
19. Calling destructor for `first`


```
28      cout << "\nCREATE FUNCTION: EXECUTION BEGINS" << endl;
29      CreateAndDestroy fifth( 5, "(local automatic in create)" );
30      static CreateAndDestroy sixth( 6, "(local static in create)" );
31      CreateAndDestroy seventh( 7, "(local automatic in create)" );
32      cout << "\nCREATE FUNCTION: EXECUTION ENDS" << endl;
33   } // end function create
```

An Example of Constructors/Destructors Being Called

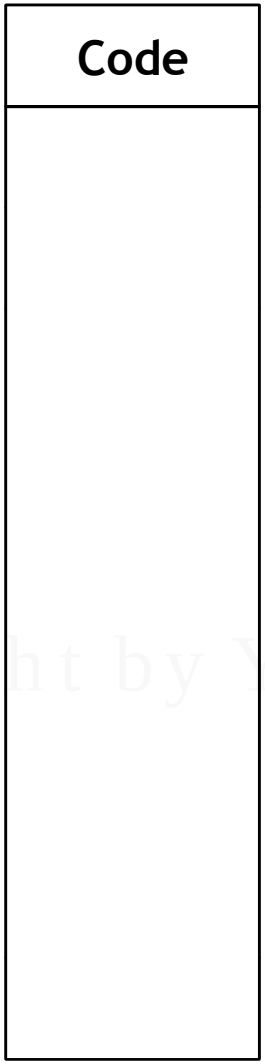
1. Locations for static data are reserved
2. Calling constructor for `first`
3. Pushing `main`'s ARI to stack (locations for local variables and parameters are reserved)
4. Calling constructor for `second`
5. Calling constructor for `third`
6. Pushing `create`'s ARI to stack (locations for local variables and parameters are reserved)
7. Calling constructor for `fifth`
8. Calling constructor for `sixth`
9. Calling constructor for `seventh`
10. Calling destructor for `seventh`
11. Calling destructor for `fifth`
12. Popping `create`'s ARI from stack
13. Calling constructor for `fourth`
14. Calling destructor for `fourth`
15. Calling destructor for `second`
16. Popping `main`'s ARI from stack
17. Calling destructor for `sixth`
18. Calling destructor for `third`
19. Calling destructor for `first`
20. Program terminates

```
    CreateAndDestroy fifth( 5, "(local automatic in create)" );
    static CreateAndDestroy sixth( 6, "(local static in create)" );
    CreateAndDestroy seventh( 7, "(local automatic in create)" );
    cout << "\nCREATE FUNCTION: EXECUTION ENDS" << endl;
} // end function create
```



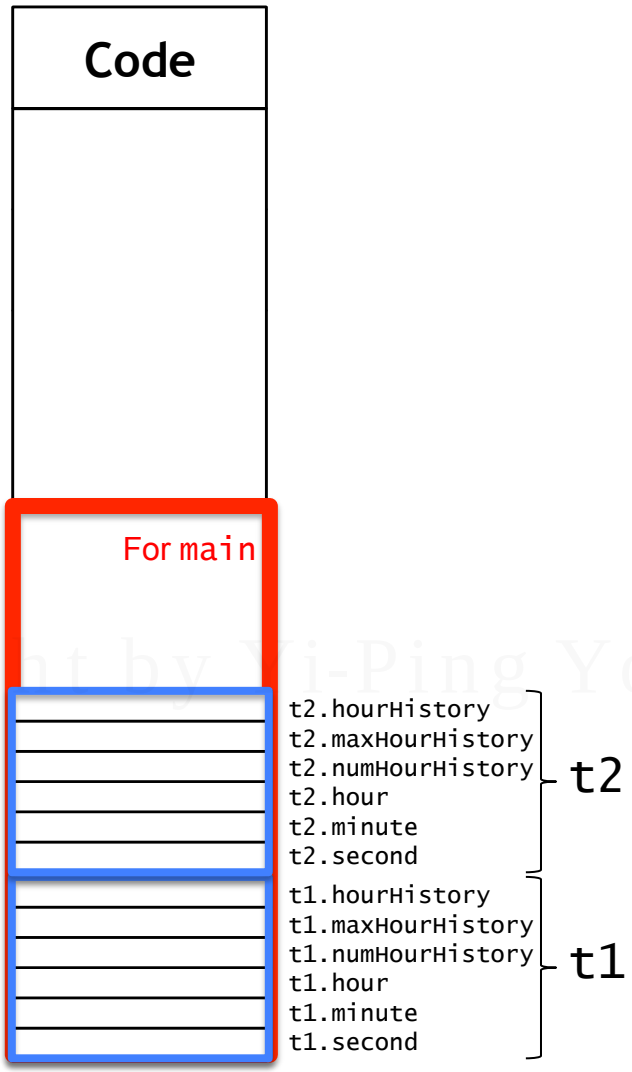
Memberwise Assignment: An Example

```
#include “time.h”

int main() {
  Time t1(3,10,10,10);
  Time t2(5);
  t2 = t1;

  return 0;
}
```



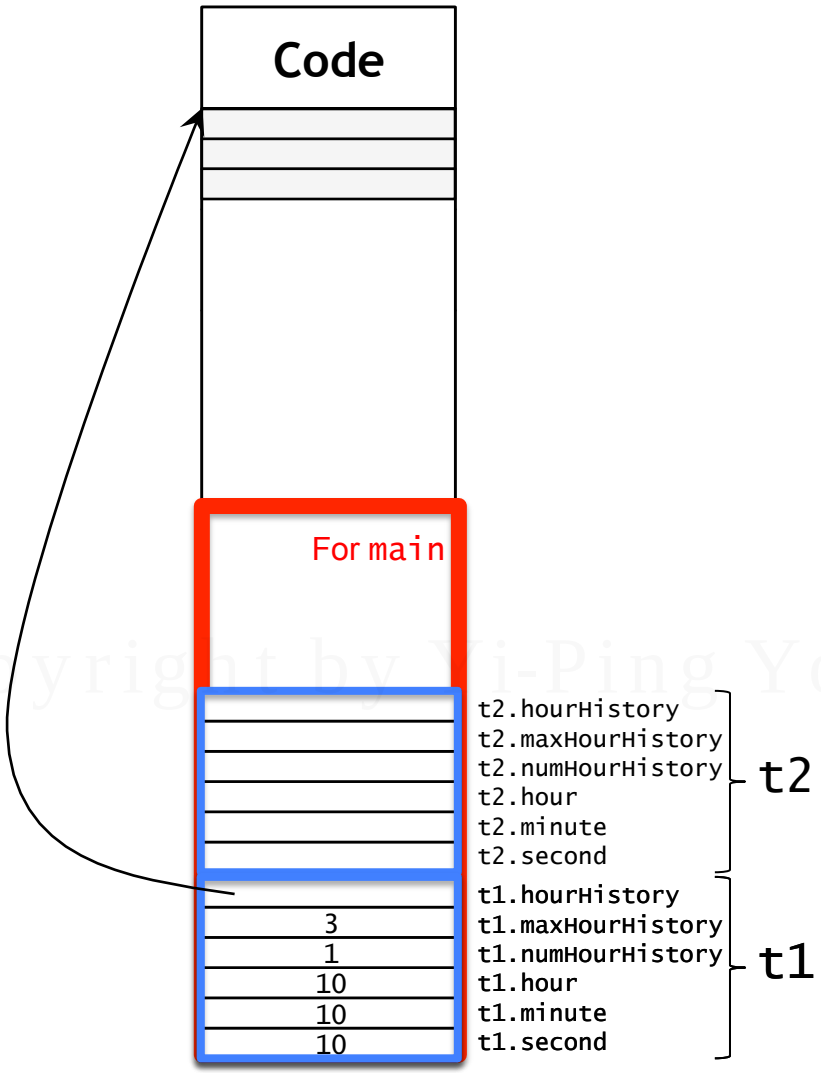
Memberwise Assignment: An Example

```
#include "time.h"

int main() {
  Time t1(3,10,10,10);
  Time t2(5);
  t2 = t1;

  return 0;
}
```



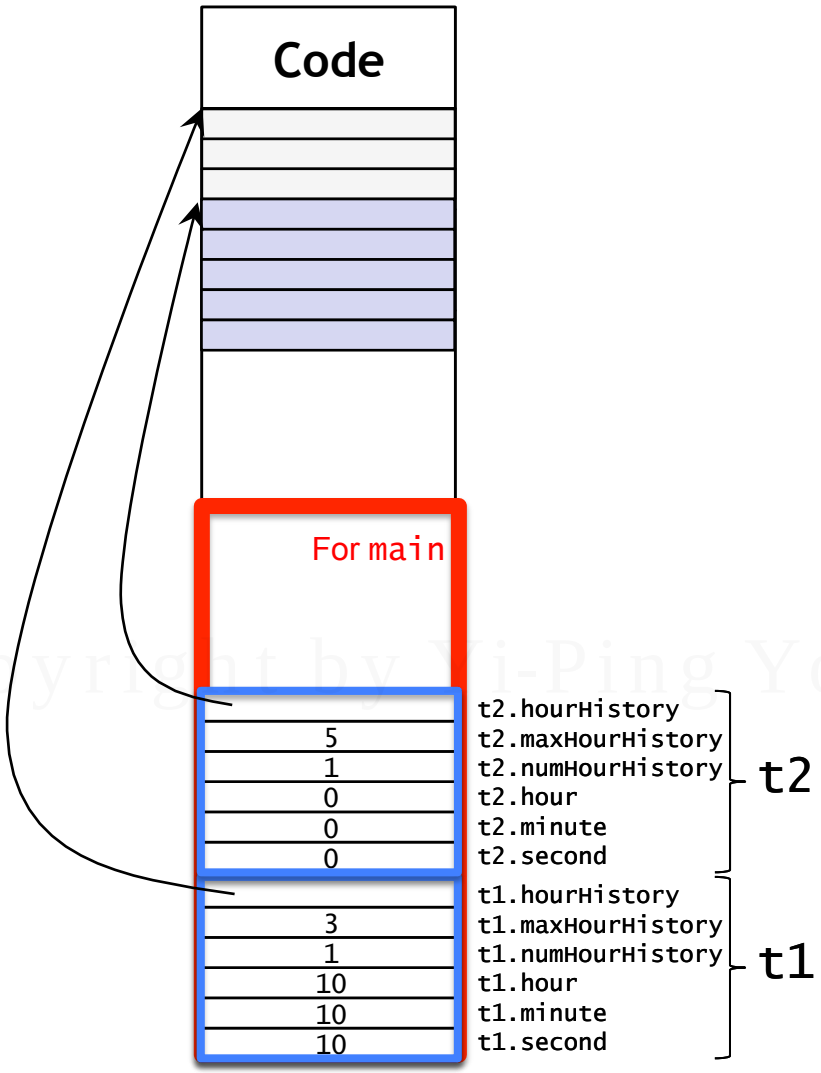
Memberwise Assignment: An Example

```
#include "time.h"

int main() {
  Time t1(3,10,10,10);
  Time t2(5);
  t2 = t1;

  return 0;
}
```



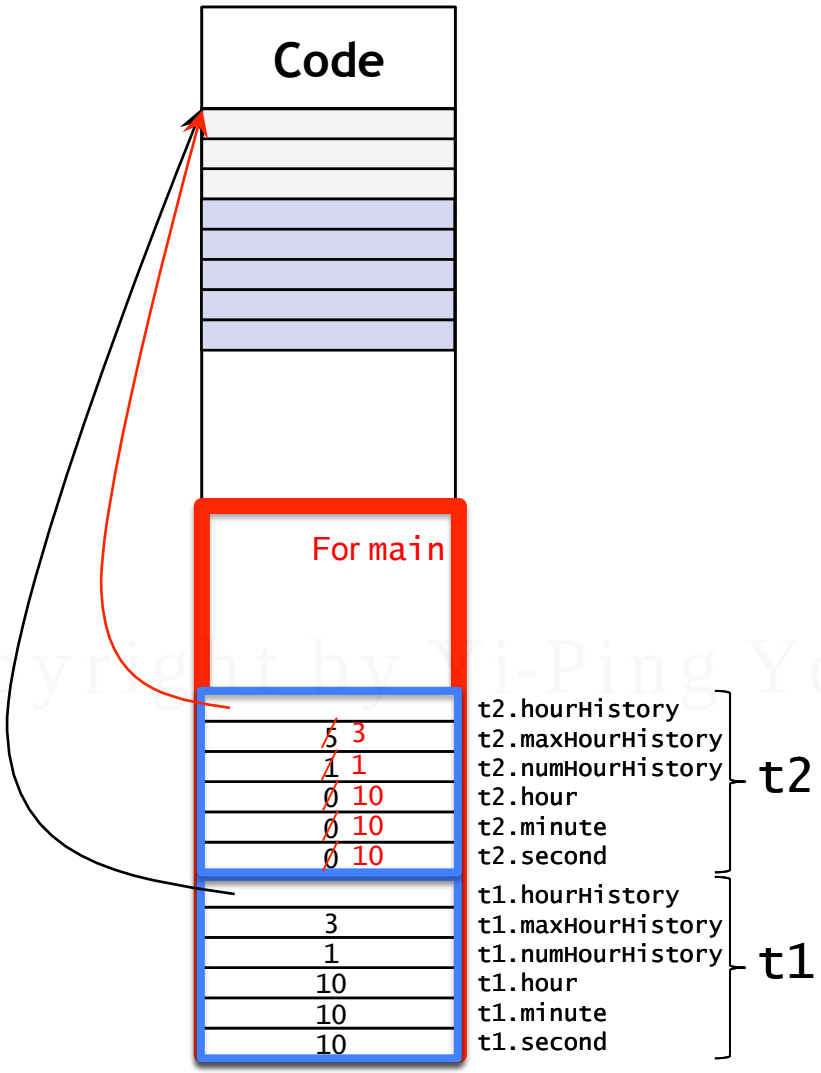
Memberwise Assignment: An Example

```
#include "time.h"

int main() {
  Time t1(3,10,10,10);
  Time t2(5);
  t2 = t1;

  return 0;
}
```



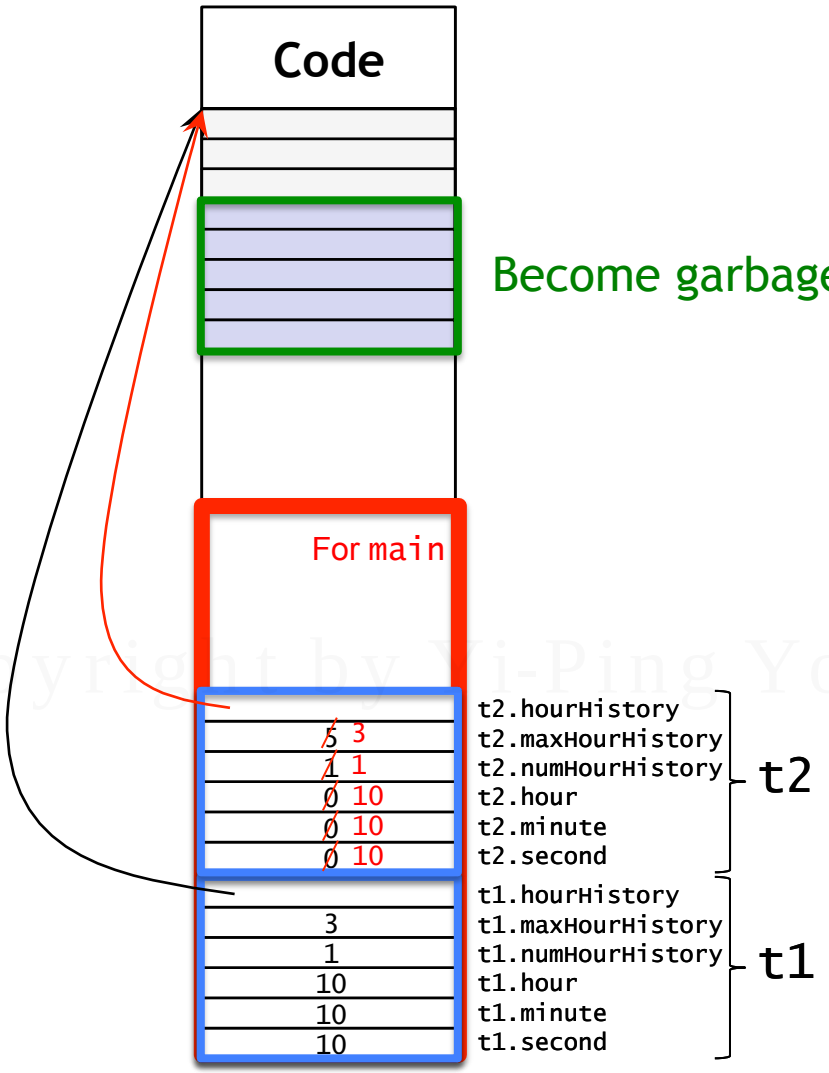
Memberwise Assignment: An Example

```
#include "time.h"

int main() {
  Time t1(3,10,10,10);
  Time t2(5);
  t2 = t1;

  return 0;
}
```



Memberwise Assignment: An Example

```
#include "time.h"

int main() {
  Time t1(3,10,10,10);
  Time t2(5);
  t2 = t1;

  return 0;
}
```

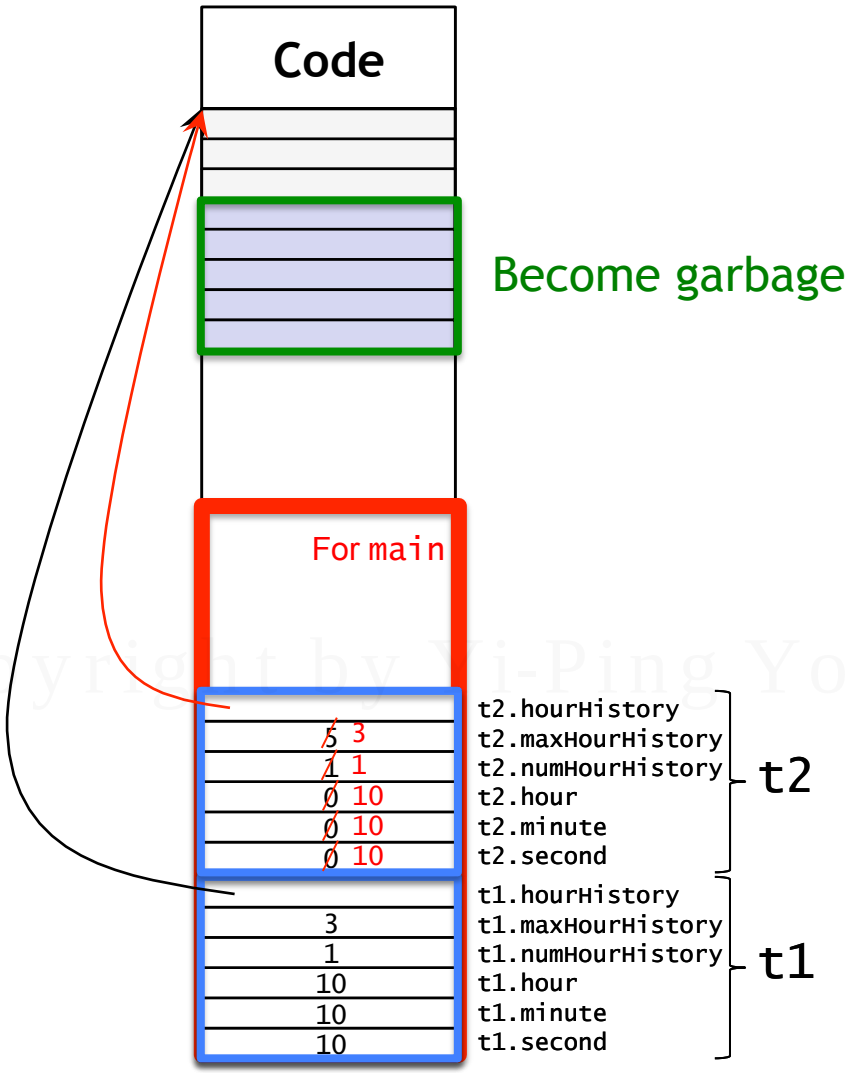


Memberwise Assignment: An Example

```
#include "time.h"

int main() {
  Time t1(3,10,10,10);
  Time t2(5);
  t2 = t1;

  return 0;
}
```

time.h

```
Time::~Time() {
  delete [] hourHistory;
}
```

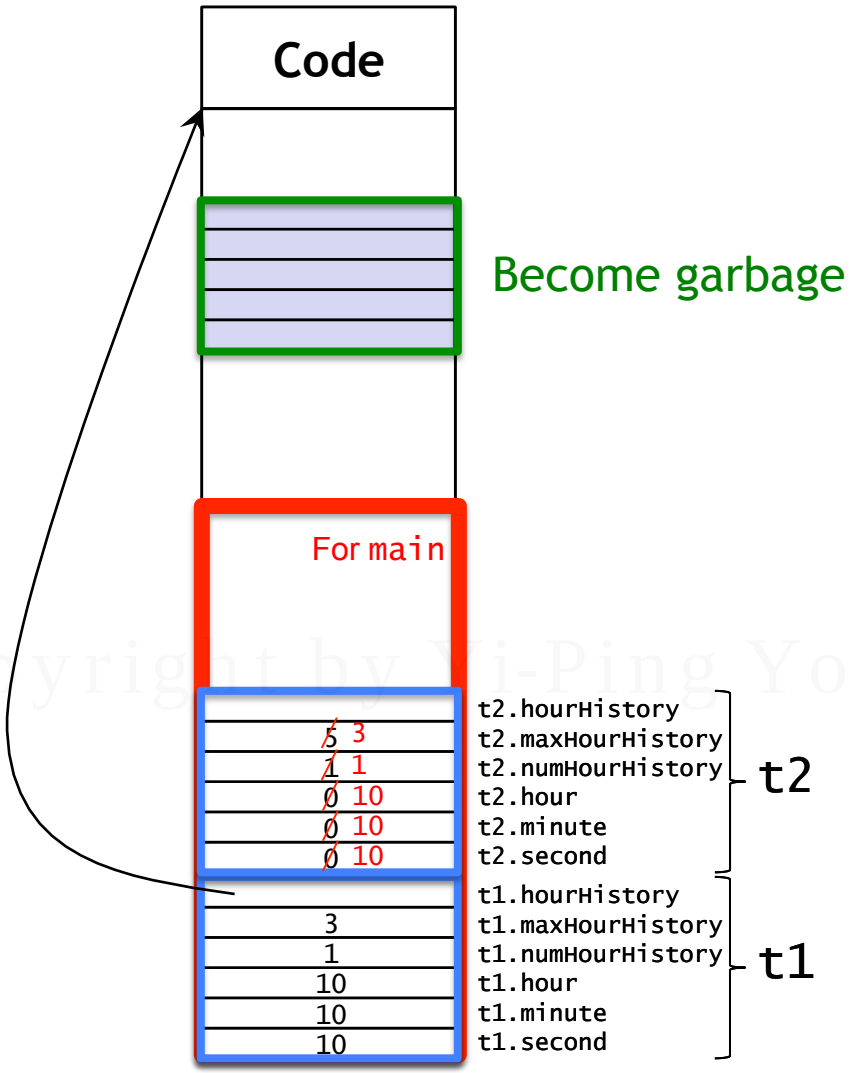


Memberwise Assignment: An Example

```
#include "time.h"

int main() {
  Time t1(3,10,10,10);
  Time t2(5);
  t2 = t1;

  return 0;
}
```

time.h

```
Time::~Time() {
  delete [] hourHistory;
}
```

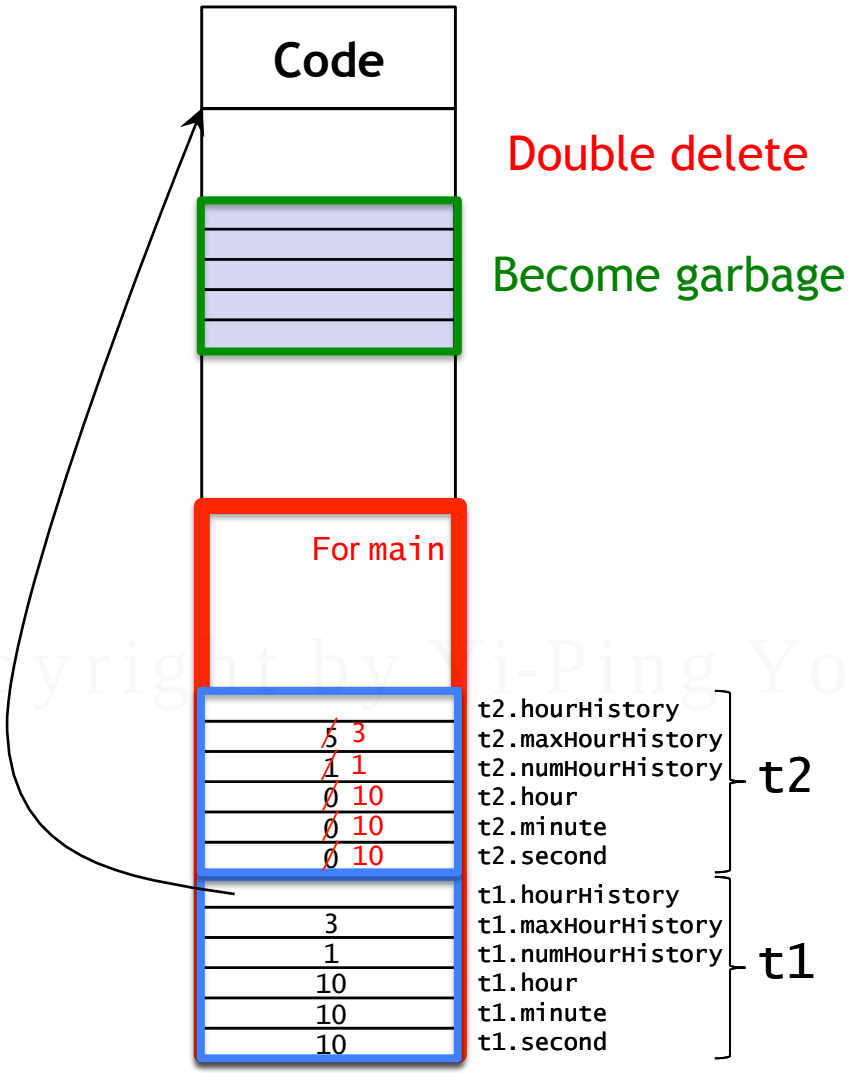


Memberwise Assignment: An Example

```
#include "time.h"

int main() {
  Time t1(3,10,10,10);
  Time t2(5);
  t2 = t1;

  return 0;
}
```

time.h

```
Time::~Time() {
  delete [] hourHistory;
}
```

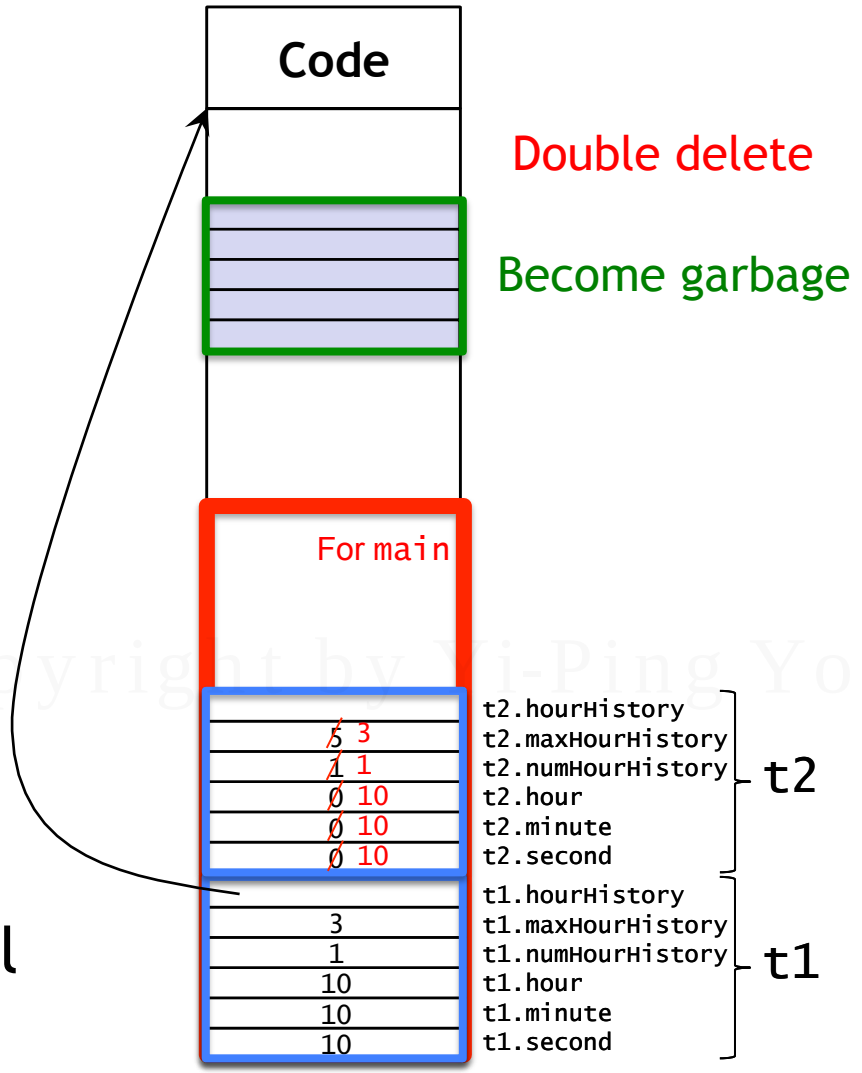


Memberwise Assignment: An Example

```
#include "time.h"

int main() {
  Time t1(3,10,10,10);
  Time t2(5);
  t2 = t1;

  return 0;
}
```

time.h

```
Time::~Time() {
  delete [] hourHistory;
}
```



- Solutions to this problem will be discussed in Chapter 11 (Operator Overloading)