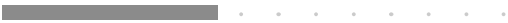


⋮

Identifier Scope and Lifetime

- **Scope**
 - Set of lines in a program
 - Where name refers to that object or function
- **Lifetime**
 - When memory allocated to object

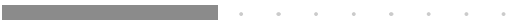


Handwriting lines for notes corresponding to the first section.

⋮

Global Declarations

- **Identifier declared outside any block** (compound statement)
 - Outside any function
- **Identifier scope**
 - Starting at declaration
 - Until end of the file
 - Known as “file scope”

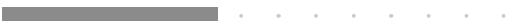


Handwriting lines for notes corresponding to the second section.

⋮

Local Declarations

- **Identifier declared inside a block**
- **Identifier scope**
 - Starting at declaration
 - Until end of the block enclosing the declaration



Handwriting lines for notes corresponding to the third section.

Formal Function Arguments

- Identifier treated as if declared inside function
- Identifier scope
 - From beginning of function body
 - To end of function body

Scope Example

```

int y = 38;          void f(int s)
void f(int s);     {
void main()        { int r = 12;
{                  s = r + t;
  int z = 47;      int I = 27;
  {               s *= I;
    const int a=90; }
    z += a;      }
  }
  z = 2 * y;      What is the scope
  f(1,2);         of each identifier?
}

```

Resolving Name Conflicts

- Scopes can overlap
 - Different identifiers
- What if name is the same?
 - Name refers to innermost scope
 - Outer scope objects are hidden

Name Conflict Example

```

void Func();
int xx = 20;
void main()
{
  cout << xx;
  {
    double xx=1.0;
    cout << xx;
  }
  Func();
}

void Func()
{
  int xx = 5;
  cout << xx;
}

```

What is the output?
What is the lifetime of each object?

Static Objects

- **Two kinds of local objects**
 - Automatic (all we've seen so far)
 - Static
- **Differ in lifetime**
 - Auto: definition to end of scope
 - Static: definition to end of execution

Static/Automatic Example

<pre> void Func(); void main() { Func(); Func(); } void Func() { int n=0; n += 1; cout << n; } </pre>	<pre> void Func(); void main() { Func(); Func(); } void Func() { static int n=0; n += 1; cout << n; } </pre>
---	--

What is the output?

Function Arguments

```
void Out2(int x, int y);  
void main()  
{  
  int x = 11;  
  int y = 22;  
  Out2(y, x);  
}  
  
void Out2(int x, int y)  
{  
  cout << x << y;  
}
```

x:
y:

x:
y:
