

CSE 3302
Programming Languages



Semantics

Chengkai Li, Weimin He
Spring 2008

Lecture 5 - Semantics, Spring 2008 CSE3302 Programming Languages, UT-Arlington
©Chengkai Li, Weimin He, 2008

1

Names



- Names: identify language entities
 - variables, procedures, functions, constants, data types, ...
- Attributes: properties of names
- Examples of attributes:
 - Data type:
`int n = 5;` (data type: integer)
`int itself is a name`
 - Value: (value: 5)
 - Location:
`int* y;`
`y = new int;`
 - Parameters, return value: `int f(int n) { ... }`
 - ...

Lecture 5 - Semantics, Spring 2008 CSE3302 Programming Languages, UT-Arlington
©Chengkai Li, Weimin He, 2008

2

Binding



- Binding:** associating attributes to names
 - declarations
 - assignments
 - declarations (prototype) and definition of a function
- The bindings can be explicit or implicit**
 - e.g. `int x;`
 - Explicit binding: the data type of `x`
 - Implicit binding: the location of `x` (static or dynamic, depending on where the declaration is)

Lecture 5 - Semantics, Spring 2008 CSE3302 Programming Languages, UT-Arlington
©Chengkai Li, Weimin He, 2008

3

Binding Time



- Binding Time:** the time when an attribute is bound to a name.
 - Static binding** (static attribute): occurs before execution
 - Language definition/implementation time:* The range of data type `int`
 - translation time (parsing/semantic analysis):* The data type of a variable
 - link time:* The body of external function
 - load time:* Location of global variable
 - Dynamic binding** (dynamic attribute): occurs during execution
 - entry/exit from procedure or program:* the value of local variables

Lecture 5 - Semantics, Spring 2008 CSE3302 Programming Languages, UT-Arlington
©Chengkai Li, Weimin He, 2008

4

Where can declarations happen?



- Blocks ({}), begin-end, ... Algol descendants: C/C++, Java, Pascal, Ada, ...)
 - e.g., C
 - Function body
 - Anywhere a statement can appear (compound statement)
- External/global
- Structured data type
- Class

Lecture 5 - Semantics, Spring 2008 CSE3302 Programming Languages, UT-Arlington
©Chengkai Li, Weimin He, 2008

5

C++ Example



```

const int Maximum = 100;
struct FullName {string LastName, string FirstName};

class Student {
private:
    struct FullName name; int Age;
public:
    void setValue(const int a, struct FullName name);
    int TStudent();
    ...
};

void Student::setAge(const int a, string lName, string fName) {
    int i;
    Age = a;
    {
        int j;
        name.LastName = lName;
        name.FirstName = fName;
    }
}

```

Lecture 5 - Semantics, Spring 2008 CSE3302 Programming Languages, UT-Arlington
©Chengkai Li, Weimin He, 2008

6

Scope of Binding



- **Scope of Binding:** the region of the program where the binding is maintained (is valid, applies).
 - **Block-structured language**
- lexical scope (static scope):* from the declaration to the end of the block containing the declaration.
- dynamic scope :* introduced later.

Lecture 5 - Semantics, Spring 2008

CSE3302 Programming Languages, UT-Arlington
©Chengkai Li, Weimin He, 2008

7

Example

```
int x;
void p(void) {
    char y;
    . .
    { int i;
    . .
    }
}
void q(void) {
    double z;
    . .
}
main() {
    int w[10];
    . .
}
```

The diagram illustrates the static scope of variables in a C-like language. The variable `x` is in global scope. Inside function `p`, `y` is local, and `i` is local to the block within `p`. Inside function `q`, `z` is local. In the `main` function, `w` is local, and `main` itself is the current block. The variable `p` is a function object, `q` is another function object, and `main` is the entry point.

CSE3302 Programming Languages, UT-Arlington
©Chengkai Li, Weimin He, 2008

8

Declaration before Use



```
void p(void) {
    int x;
    . .
    char y; ] y ] x
    . .
}
```

Exception in OO languages: Scope of local declarations inside a class declaration includes the whole class.

```
public class {
    public int getValue() { return value; }
    int value;
}
```

Lecture 5 - Semantics, Spring 2008

CSE3302 Programming Languages, UT-Arlington
©Chengkai Li, Weimin He, 2008

9

Scope Hole



- **Scope Hole:** Declarations in nested blocks take precedence over the previous declarations. That is, binding becomes **invisible/hidden**.

```
int x;

void p(void) {
    char x;
    x = 'a';
    . .
}

main() {
    x = 2;
    . .
}
```

The diagram shows a variable `x` declared at the top level. Inside function `p`, there is another declaration `x` with type character. This inner `x` shadows the outer `x`, making the outer `x` invisible or "hidden".

CSE3302 Programming Languages, UT-Arlington
©Chengkai Li, Weimin He, 2008

10

Access Hidden Declarations



- scope resolution operator :: (C++)

```
int x;

void p(void) {
    char x;
    x = 'a';
    ::x=42;
    . .
}

main() {
    x = 2;
    . .
}
```

The diagram illustrates the use of the scope resolution operator `::` in C++. It shows a global variable `x` and a local variable `x` in function `p`. The expression `::x=42;` uses the resolution operator to refer to the global `x`, effectively hiding the local one. In the `main` function, the global `x` is accessed directly.

Lecture 5 - Semantics, Spring 2008

CSE3302 Programming Languages, UT-Arlington
©Chengkai Li, Weimin He, 2008

11

Hide a Declaration



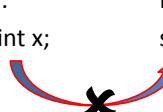
- File 1:
`extern int x;`

File 2:
`int x;`



- File 1:
`extern int x;`

File 2:
`static int x;`



CSE3302 Programming Languages, UT-Arlington
©Chengkai Li, Weimin He, 2008

12

Symbol Table



- Symbol Table: maintain bindings. Can be viewed as functions that map names to their attributes.

```

graph LR
    subgraph SymbolTable [SymbolTable]
        direction TB
        Names[Names] --> Attributes[Attributes]
    end
  
```

Lecture 5 - Semantics, Spring 2008 CSE3302 Programming Languages, UT-Arlington ©Chengkai Li, Weimin He, 2008 13

Static vs. Dynamic Scope



- Static scope (lexical scope):**
 - scope maintained statically (during compilation)
 - follow the layout of source codes
 - used in most languages
- Dynamic scope:**
 - scope maintained dynamically (during execution)
 - follow the execution path
 - few languages use it (The bindings cannot be determined statically, may depend on user input).
 - Lisp: considered a bug by its inventor.
 - Perl: can choose lexical or dynamic scope

Lecture 5 - Semantics, Spring 2008 CSE3302 Programming Languages, UT-Arlington ©Chengkai Li, Weimin He, 2008 14

Static Scope



```

int x = 1;
char y = 'a';

void p(void) {
    double x=2.5;
    printf("%c\n",y);
}

void q(void) {
    int y = 42;
    printf("%d\n",x);
    p();
}

main() {
    char x = 'b';
    q();
}
  
```

X integer, global
Y character, global

Lecture 5 - Semantics, Spring 2008 CSE3302 Programming Languages, UT-Arlington ©Chengkai Li, Weimin He, 2008 15

Static Scope



The symbol table in p:
the bindings available in p

```

int x = 1;
char y = 'a';

void p(void) {
    double x=2.5;
    printf("%c\n",y);
}

void q(void) {
    int y = 42;
    printf("%d\n",x);
    p();
}

main() {
    char x = 'b';
    q();
}
  
```

X double, local to p
Y Character, global

Lecture 5 - Semantics, Spring 2008 CSE3302 Programming Languages, UT-Arlington ©Chengkai Li, Weimin He, 2008 16

Static Scope



The symbol table in q:
the bindings available in q

```

int x = 1;
char y = 'a';

void p(void) {
    double x=2.5;
    printf("%c\n",y);
}

void q(void) {
    int y = 42;
    printf("%d\n",x);
    p();
}

main() {
    char x = 'b';
    q();
}
  
```

X integer, global
Y integer, local to q
character, global

Lecture 5 - Semantics, Spring 2008 CSE3302 Programming Languages, UT-Arlington ©Chengkai Li, Weimin He, 2008 17

Static Scope



The symbol table in main:
the bindings available in main

```

int x = 1;
char y = 'a';

void p(void) {
    double x=2.5;
    printf("%c\n",y);
}

void q(void) {
    int y = 42;
    printf("%d\n",x);
    p();
}

main() {
    char x = 'b';
    q();
}
  
```

X character, local to main
Y character, global

Lecture 5 - Semantics, Spring 2008 CSE3302 Programming Languages, UT-Arlington ©Chengkai Li, Weimin He, 2008 18

Static Scope

The symbol table in previous slides are built during compilation

- The bindings are used in generating the machine code
- Result:
 - 1**
 - a**
- E.g., semantics of **q**:

```
void q(void) {
    int y = 42;
    printf("%d\n", x);
    p();
}
```

The symbol table in **q**:
the bindings available in **q**

X	integer, global
Y	integer, local to q

character, global

Lecture 5 - Semantics, Spring 2008 CSE3302 Programming Languages, UT-Arlington ©Chengkai Li, Weimin He, 2008 19

Practice for Static Scope

int x, y;

```
void g(void) {
    x = x + 1;
    y = x + 1;
}

void f(void) {
    int x;
    y = y + 1;
    x = y + 1;
    g();
}

main() {
    x = 1;
    y = 2;
    f();
    g();
    printf("x=%d,y=%d\n", x, y);
}
```

Question 1:
Draw the symbol table at the given points in the program, using static scope?

Question 2:
What does the program print, using static scope?

Lecture 5 - Semantics, Spring 2008 CSE3302 Programming Languages, UT-Arlington ©Chengkai Li, Weimin He, 2008 20

What if dynamic scope is used?

int x = 1;
char y = 'a';

```
void p(void) {
    double x=2.5;
    printf("%c\n", y);
}

void q(void) {
    int y = 42;
    printf("%d\n", x);
    p();
}

main() {
    char x = 'b';
    q();
}
```

The symbol table in **main**:
the bindings available in **main**

X	character, 'b', local to main
Y	integer, 1, global
Z	character, 'a', global

Lecture 5 - Semantics, Spring 2008 CSE3302 Programming Languages, UT-Arlington ©Chengkai Li, Weimin He, 2008 21

What if dynamic scope is used?

int x = 1;
char y = 'a';

```
void p(void) {
    double x=2.5;
    printf("%c\n", y);
}

void q(void) {
    int y = 42;
    printf("%d\n", x);
    p();
}

main() {
    char x = 'b';
    q();
}
```

The symbol table in **main**:
the bindings available in **main**

X	character, 'b', local to main
Y	integer, 1, global
Z	character, 'a', global

Lecture 5 - Semantics, Spring 2008 CSE3302 Programming Languages, UT-Arlington ©Chengkai Li, Weimin He, 2008 22

What if dynamic scope is used?

int x = 1;
char y = 'a';

```
void p(void) {
    double x=2.5;
    printf("%c\n", y);
}

void q(void) {
    int y = 42;
    printf("%d\n", x);
    p();
}

main() {
    char x = 'b';
    q();
}
```

The symbol table in **q**:
the bindings available in **q**

X	character, 'b', local to main
Y	integer, 1, global
Z	integer, 42, local to q
W	character, 'a', global

Lecture 5 - Semantics, Spring 2008 CSE3302 Programming Languages, UT-Arlington ©Chengkai Li, Weimin He, 2008 23

What if dynamic scope is used?

int x = 1;
char y = 'a';

```
void p(void) {
    double x=2.5;
    printf("%c\n", y);
}

void q(void) {
    int y = 42;
    printf("%d\n", x);
    p();
}

main() {
    char x = 'b';
    q();
}
```

The symbol table in **p**:
the bindings available in **p**

X	double, 2.5, local to p
Y	character, 'b', local to main
Z	integer, 1, global

The symbol table in **q**:
the bindings available in **q**

X	integer, 42, local to q
Y	character, 'a', global
Z	character, 'b', local to main
W	*

Lecture 5 - Semantics, Spring 2008 CSE3302 Programming Languages, UT-Arlington ©Chengkai Li, Weimin He, 2008 24

Practice for Dynamic Scope 

```

int x,y;

void g(void) {
    x = x + 1;
    y = x + 1;
}

void f(void) {
    int x;
    y = y + 1;
    x = y + 1;
    g();
}

main() {
    x = 1;
    y = 2;
    f();
    g();
    printf("x=%d,y=%d\n",x,y);
}

```

Point 1 → **Question 1:** Draw the symbol table at the given points in the program, using dynamic scope?

Point 2 → **Question 2:** What does the program print, using dynamic scope?

Point 3 → **Question 3:**

Lecture 5 - Semantics, Spring 2008 CSE3302 Programming Languages, UT-Arlington
©Chengkai Li, Weimin He, 2008 25