C/Java Syntax

Slides by Mark Hancock
(adapted from notes by Craig Schock)

Lecture 02 Summary

- Keywords
- Variable Declarations
- Data Types
- Operators
- Statements
  - if, switch, while, do-while, for
- Functions

By the end of this lecture, you will be able to identify the different parts of a C program. You will also be able to create a simple C program.

What is a keyword?

What are some of the keywords in Python?

Keywords in C (all 32 of them)

```
auto        double      int         struct
break       else        long        switch
case        enum        register    typedef
char        extern      return      union
const       float       short       unsigned
continue    for         signed      void
default     goto        sizeof      volatile
do          if          static      while
```

Variable Declarations

In Python…

- which lines are okay?

```
1var = 5
_pi = 3.1415
str+ing = “hello”
while = 3.3
x = ‘0’
myStr2 = “I’m a string”
```

C Variable Names

- Start with a letter or underscore
- Subsequent characters can also be numbers
- Can't use reserved keywords
- Case sensitive
  - `mystring` is not the same as `myString`

C Declarations

- Must specify the *type* of the variable (which will never change)

In Python…

```
x = 5  # x starts out as an integer

… more code …

x = “now I’m a string”
```

In C…

```
int x = 5;

… more code …

x = “Nooooooooooooo!”
```

Data Types

- Integral Types
  - `char`
  - `short`
  - `int`
  - `long`
  - `long long`
- Floating Point Types
  - `float`
  - `double`

Why isn't there just one `int` and one `float` type?

`sizeof` Operator

```
main()
{
    printf(“int: %d bytes\n", sizeof(int));
}
```

In C…

- which lines are okay?

```
int 1var = 5
float _pi = 3.1415
short str+ing = “hello”
double while = 3.3
char x = ‘0’
int myStr2 = “I’m a string”
```

Operators

What are the operators?

```
int x = 5;
int y = x + 10;
int z = (x + 20) / y;

if (z < x)
    z = x * y;
```

Operators in C

Operator	Description
++ --	postfix increment and decrement
()	function call
[]	array subscription
->	element selection through pointer
++ --	prefix increment and decrement
+ -	unary plus and minus
! ~	logical not, bitwise not
(*type*)	type cast
*	indirection/dereferencing of pointer
&	address of
sizeof	get the size of element
* / %	multiplication, division, modulus

Operators in C (cont’d)

Operator	Description
+ -	addition and subtraction
<< >>	bitwise shift left and right
< <= > >=	less than, less than equals, greater than, greater than equals
== !=	equals, not equals
&	bitwise AND
^	bitwise XOR (exclusive-or)
|	Bitwise OR
&&	Logical AND
||	Logical OR
cond?*t*:*f*	ternary command
= += -= *= /= %=	Assignment operators

Increment and Decrement

```
int i = 0;
int j = 10;

i = i + 1;
i++;

j = j - 1;
j--;
```

What will this output?

```
main()
{
    int x = 5;
    int y = 20;

    x++;
    y--;

    printf("x = %d\n", x);
    printf("y = %d\n", y);
}
```

Prefix vs. Postfix

- Prefix = first thing that happens (before)
  - e.g., `++i`

- Postfix = last thing that happens (after)
  - e.g., `i++`

Prefix vs. Postfix Example

```
main()
{
    int x = 5;
    int y = 5;
    int a;
    int b;

    a = x++; /* postfix */
    b = ++y; /* prefix */

    printf("a = %d\n", a);
    printf("b = %d\n", b);
    printf("x = %d\n", x);
    printf("y = %d\n", y);
}
```

What will this output?

```
main()
{
    int i = 70;
    int j = 42;

    int a = i++ * ++j;

    printf("i = %d\n", i);
    printf("j = %d\n", j);
    printf("a = %d\n", a);
}
```

Statements

In Python…

```
def farenheit_to_celcius(temp):
    return (temp - 32.0) * (5.0 / 9.0)

1 print "Enter the temperature in Farenheit:"
2 temp = input()

if temp < -459.67:
    print "It can’t possibly be that cold!"
else:
    tempInCelcius = farenheit_to_celcius(temp)
    print "In Farenheit: %f" % temp
    print "In Celcius: %f" % tempInCelcius
```

Enter “72”

```
def farenheit_to_celcius(temp):
  5 return (temp - 32.0) * (5.0 / 9.0)

1 print "Enter the temperature in Farenheit:"
2 temp = input()

3 if temp < -459.67:
      print "It can’t possibly be that cold!"
  else:
  4 tempInCelcius = farenheit_to_celcius(temp)
  6 print "In Farenheit: %f" % temp
  7 print "In Celcius: %f" % tempInCelcius
```

In C…

- Simple statements end with a semi-colon (;)
- Any *white space* is ignored by the computer
  - spaces, tabs, new lines
  - but is very helpful to *people* who read the code

`if` Statement Syntax

```
if (<conditional>)
    <what-to-do-if-true>;
else
    <what-to-do-if-false>;
```

Example

```
main()
{
    int x = 3;

    if (x % 2 == 0)
        printf("x is even\n");
    else
        printf("x is odd\n");
}
```

Example

```
main()
{
int x = 3;

if (x % 2 == 0)
printf("x is even\n");
else
printf("x is odd\n");
}
```

Example

```
main(){ int x = 3; if (x % 2 == 0)printf(
"x is even\n"); else printf("x is odd\n");}
```

How would we write this in C?

```
if temp < -459.67:
    print "It can’t possibly be that cold!"
else:
    tempInCelcius = farenheit_to_celcius(temp)
    print "In Farenheit: %f" % temp
    print "In Celcius: %f" % tempInCelcius
```

Would this work?

```
if (temp < -459.67)
    printf("It can’t possibly be that cold!\n");
else
    tempInCelcius = farenheit_to_celcius(temp);
    printf("In Farenheit: %f\n", temp);
    printf("In Celcius: %f", tempInCelcius);
```

Compare it to this

```
if (temp < -459.67)
    printf("It can’t possibly be that cold!\n");
else
    tempInCelcius = farenheit_to_celcius(temp);
printf("In Farenheit: %f\n", temp);
printf("In Celcius: %f", tempInCelcius);
```

Blocks

- Statements can be grouped together into a *compound* statement by enclosing them in curly braces ({})

Adjusted `if` Statement Syntax

```
if (<conditional>)
{
    <what-to-do-if-true>;
    <possibly>;
    <containing>;
    <multiple lines>;
}
else
{
    <what-to-do-if-false>;
    <also-more-lines>;
}
```

How would we write this in C?

```
if temp < -459.67:
    print "It can’t possibly be that cold!"
else:
    tempInCelcius = farenheit_to_celcius(temp)
    print "In Farenheit: %f" % temp
    print "In Celcius: %f" % tempInCelcius
```

Answer

```
if (temp < -459.67)
{
    printf("It can’t possibly be that cold!\n");
}
else
{
    tempInCelcius = farenheit_to_celcius(temp);
    printf("In Farenheit: %f\n", temp);
    printf("In Celcius: %f", tempInCelcius);
}
```

Consider this code...

```
if (x > 50)
    if (y > 200)
        z = x * y;
else
    printf("error!\n");
```

- Which `if` does the `else` belong to?

Good Practice

```
if (x > 50)
{
    if (y > 200)
    {
        z = x * y;
    }
    else
    {
        printf ("error!\n");
    }
}
```

`else if` and `switch` Statements

What would this do?

```
int x = 3;

if (x == 1)
    printf("One!\n");
else
    if (x == 2)
        printf("Two!\n");
    else
        if (x == 3)
            printf("Three!\n");
        else
            printf("Not one, two, or three");
```

What would this do?

```
int x = 3;

if (x == 1)
    printf("One!\n");
else if (x == 2)
    printf("Two!\n");
else if (x == 3)
    printf("Three!\n");
else
    printf("Not one, two, or three");
```

`switch` Statement Syntax

```
switch (<variable>)
{
    case <value-one>:
        <code>;
        <code>;
        break;

    case <value-two>:
        <code>;
        <code>;
        break;

    default:
        <code>;
        break;
}
```

Example

```
int x = 3;

switch (x)
{
    case 1:
        printf("One!\n");
        break;
    case 2:
        printf("Two!\n");
        break;
    case 3:
        printf("Three!\n");
        break;
    default:
        printf("Invalid choice!\n");
        break;
}
```

Loops

Example

- Factorial
  - $n! = n \cdot (n-1) \cdot (n-2) \cdot \ldots \cdot 3 \cdot 2 \cdot 1$

Complete this Python program

```
print "Enter a number greater than zero:"
n = input()

...
```

`while` Syntax

```
while (<condition>)
    <statement>
```

What do you think this would do?

```
main()
{
    int i = 0;

    while (i < 10)
        printf("i = %d\n", i);
        i++;
}
```

Without the infinite loop

```
main()
{
    int i = 0;

    while (i < 10)
    {
        printf("i = %d\n", i);
        i++;
    }
}
```

Complete this C Program

```
main()
{
    int n;

    printf("Enter a number greater than zero: ");
    scanf("%d", n);

    ...
}
```

`do-while` Syntax

```
do
{
    <statement>;
    <statement2>;
} while (<condition>);
```

Compare

```
int x = 5;

while (x > 0)
{
    printf("%d\n",x);
    x = x - 1;
}
```

```
int x = 5;

do
{
    printf("%d\n",x);
    x = x - 1;
} while (x > 0);
```

`for` Syntax

```
for (<a>; <b>; <c>)
{
    <statements>
}
```

- <a> = Initialization of looping variable
- <b> = Condition
- <c> = Modification of looping variable

For Loops

```
<a>
while (<b>)
{
    <statements>
    <c>
}
```

```
for (<a>; <b>; <c>;)
{
    <statements>
}
```

Example

```
int i;
for (i = 0; i < 5; i++)
{
    printf("%d\n", i);
}
```

Complete this C Program

```
main()
{
    int n;

    printf("Enter a number greater than zero: ");
    scanf("%d", n);

    ...
}
```

Functions

Functions in C

- Must specify what type is returned
  - if there is no return statement, must return `void`

- Must specify the type of each parameter

Function Syntax

```
<return-type> function_name(<parameters>)
{
    <statements>
}
```

Examples

```
int add(int x, int y)
{
    return x + y;
}

void print_int(int i)
{
    printf("%d\n", i);
}
```

Remember from before

```
def farenheit_to_celcius(temp):
    return (temp - 32.0) * (5.0 / 9.0)
```

What would it look like in C?

```
def farenheit_to_celcius(temp):
    return (temp - 32.0) * (5.0 / 9.0)
```

What would it look like in C?

```
float farenheit_to_celcius(float temp)
{
    return (temp - 32.0) * (5.0 / 9.0);
}
```

What is `main`?

Variants of `main`

```
void main()
{
}

int main()
{
    return 0;
}

int main(int argc, char **argv)
{
    return 1;
}
```

Write this whole program in C

```
def farenheit_to_celcius(temp):
    return (temp - 32.0) * (5.0 / 9.0)

print "Enter the temperature in Farenheit:"
temp = input()

if temp < -459.67:
    print "It can’t possibly be that cold!"
else:
    tempInCelcius = farenheit_to_celcius(temp)
    print "In Farenheit: %f" % temp
    print "In Celcius: %f" % tempInCelcius
```

Identify the different parts

```
int max(int a, int b)
{
    if (a > b)
        return a;
    else
        return b;
}

void main()
{
    int first;
    int second;
    int bigger;

    printf("Enter a number: ");
    scanf("%d", &first);
    printf("Enter another number: ");
    scanf("%d", &second);

    bigger = max(first, second);
    printf("The bigger number is: %d\n", bigger);
}
```

Keywords

```
int max(int a, int b)
{
    if (a > b)
        return a;
    else
        return b;
}

void main()
{
    int first;
    int second;
    int bigger;

    printf("Enter a number: ");
    scanf("%d", &first);
    printf("Enter another number: ");
    scanf("%d", &second);

    bigger = max(first, second);
    printf("The bigger number is: %d\n", bigger);
}
```

Variable Declarations

```
int max(int a, int b)
{
    if (a > b)
        return a;
    else
        return b;
}

void main()
{
    int first;
    int second;
    int bigger;

    printf("Enter a number: ");
    scanf("%d", &first);
    printf("Enter another number: ");
    scanf("%d", &second);

    bigger = max(first, second);
    printf("The bigger number is: %d\n", bigger);
}
```

Data Types

```
int max(int a, int b)
{
    if (a > b)
        return a;
    else
        return b;
}

void main()
{
    int first;
    int second;
    int bigger;

    printf("Enter a number: ");
    scanf("%d", &first);
    printf("Enter another number: ");
    scanf("%d", &second);

    bigger = max(first, second);
    printf("The bigger number is: %d\n", bigger);
}
```

Operators

```
int max(int a, int b)
{
    if (a > b)
        return a;
    else
        return b;
}

void main()
{
    int first;
    int second;
    int bigger;

    printf("Enter a number: ");
    scanf("%d", &first);
    printf("Enter another number: ");
    scanf("%d", &second);

    bigger = max(first, second);
    printf("The bigger number is: %d\n", bigger);
}
```

Statements

```
int max(int a, int b)
{
    if (a > b)
        return a;
    else
        return b;
}

void main()
{
    int first;
    int second;
    int bigger;

    printf("Enter a number: ");
    scanf("%d", &first);
    printf("Enter another number: ");
    scanf("%d", &second);

    bigger = max(first, second);
    printf("The bigger number is: %d\n", bigger);
}
```

Functions

```
int max(int a, int b)
{
    if (a > b)
        return a;
    else
        return b;
}
```

```
void main()
{
    int first;
    int second;
    int bigger;

    printf("Enter a number: ");
    scanf("%d", &first);
    printf("Enter another number: ");
    scanf("%d", &second);

    bigger = max(first, second);
    printf("The bigger number is: %d\n", bigger);
}
```

Create a Program in C

- Input:
  - three floating point numbers
- Output:
  - the average of those three numbers

- Use:
  - `scanf` to get the input
  - `printf` to show the result
  - a function to calculate the average

Lecture 02 Summary

- Keywords
- Variable Declarations
- Data Types
- Operators
- Statements
  - if, switch, while, do-while, for
- Functions

Next Class

- Arrays and Strings