Basic Elements I

DM2111
C++ Programming

Introduction

Introduction	Array and Strings
Problem solving	Array and Strings
Basic elements of C++	Pointers
Basic elements of C++	Pointers
Logic and branching	I/O operations
Repetition	Structs
Functions	Others
Functions	

Agenda

- Compilation process
- Tokens
- Identifiers
- Variables
- Expressions
- Input / Output

Compilation Process

1. Preprocessing
   - Deals with the preprocessor directives such as #include and #define
   - Tokenization
2. Compilation
   - Process the C++ code and produce an object file
   - Syntax errors
3. Linking
   - Links the object files together and produces the final compilation output
   - Definition error

Tokens

Tokens can be either

- Identifiers
- Keywords
- Literals
- Operators
- Punctuators
- Comments

Identifiers

- Identifiers are reference names
- C++ identifiers
  - Can only consist of letters, digits and _
  - Must begin with a letter or _
  - Must not have white spaces
  - Are case sensitive
  - Must not be reserved words
- Are these identifiers valid?

```
employee salary
Hello!
one+two
2nd
next
float
Float
```

Keywords / Reserved words

Some common reserved words

```
int     float  char   void
switch  while  try    throw
this    new    static true
false   short  long   return
```

Literals

```
157 // integer constant
0xFE // integer constant
'c' // character constant
0.2 // floating constant
0.2E-01 // floating constant
"dog" // string literal
```

Operators

Some common operators

```
+    -    *    /
.    ;    ?    ,
<    =    >    !
<=   !=   ==   >=
||   &&   ->   ::
```

Punctuators

Punctuators do not specify any operations that yields a value

```
! % ^ & * ( ) - + = { } | ~
 [ ] \ ; ' : " < > ? , . / #
```

Parts of a program

- Identifiers
- Keywords
- Literals
- Operators
- Punctuators
- Comments
- Preprocessor

```cpp
// my first program in C++
#include <iostream>

int main()
{
  int score = 1;
  std::cout << "Hello World!";
}
```

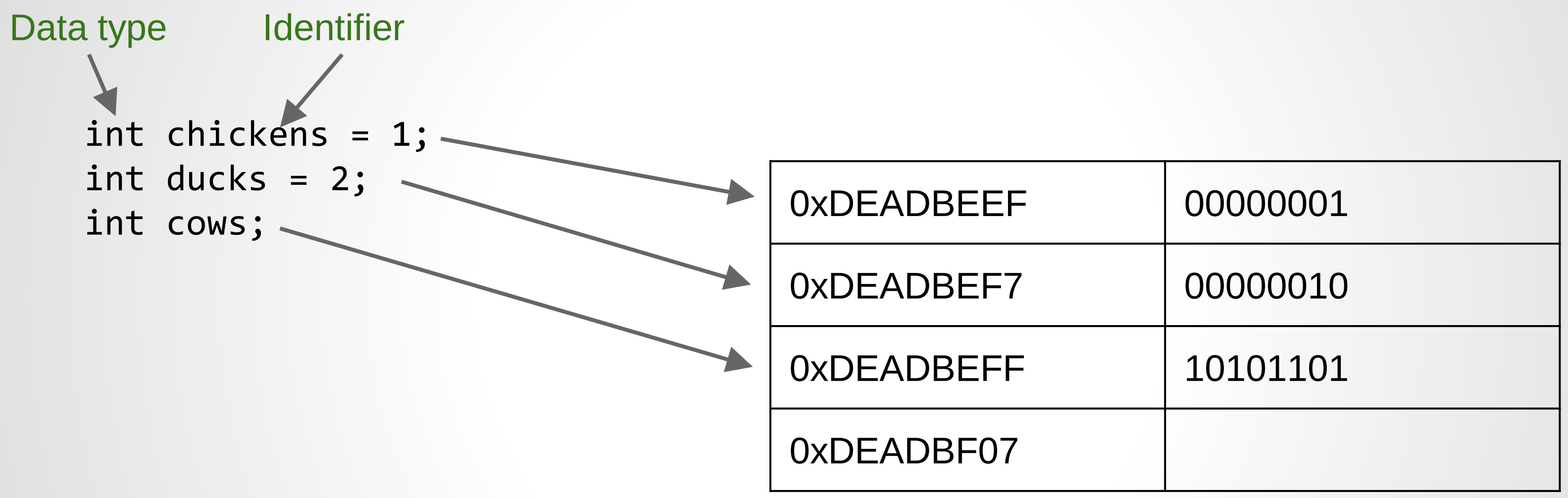
Hey girl,
You are my semicolon; always present in everything I do.

Fundamental Data Types

- Boolean
  - o True or False
- Character
  - o ‘a’, ‘A’, ‘0’, ‘-’, ‘\\’, ‘\’’
- Integer
  - o 14, 44577687, 0, -6983
- Floating point
  - o 1.4, 1.0, -4.9, 0.0

Declaration of Variables

- Allocate memory to identifier
- All variables must be declared before use



```
int chickens = 1;
int ducks = 2;
int cows;
```

0xDEADBEEF	00000001
0xDEADBEF7	00000010
0xDEADBEFF	10101101
0xDEADBF07	

Variables

- Values are assigned with the = operator

```
int num;
num = 2;
```

- A variable is said to be initialised the first time a value is placed in it.

```
int num = 2;
```

- Variables accept data based on type declared; if types don’t match, there will be a type conversion.

```
int num = 2.5; //num holds the value of 2
```

- All variables should be initialised before using; otherwise it will contain some random value.

```
int goose;
std::cout << goose; //some random number
```

Expressions

- An expression eventually yields a value
- Mixed expression – an expression that has operands of different data types

- Examples

```
2 + 3

3 / 5

3 * (5.2 + 4.7)

10 + value * 2
```

Input / Output

- printf - C function

```
int printf ( const char * format, ... );
```

```
#include <stdio.h>

int main()
{
    printf ("Hello ");
    printf ("World!");
    printf ("\n");

    printf ("How's\nlife?");
    return 0;
}
```

Output

```
Hello World!
How's
life?
```

Input / Output

- printf

```
int val1 = 5, val2 = 6;

printf ("value = %d\n", val1);
printf ("added = %d\n", val1 + val2);
```

Output

```
value = 5
added = 11
```

Input / Output

- scanf

```
int scanf ( const char * format, ... );
```

```
int input;

scanf (”%d”, &input);
printf (”input value is %d”, input);
```

Output

```
5
input value is 5
```

Input / Output

- scanf

```
float length;

printf ("Please enter length in inches: ");
scanf ("%f", &length);
printf ("Length in cm is %f", length * 2.54);
```

Output

```
Please enter length in inches: 12
Length in cm is 30.480000
```

Input / Output

- Stream extraction (>>)
  - Used on an input stream object, usually cin
  - cin is tied to the standard input, usually the keyboard

```
float length;

printf (”Please enter length in inches: ”);
cin >> length;
printf (”Length in cm is %f\n”, length * 2.54);
```

Input / Output

- Stream insertion (<<)
  - Used on an output stream object, usually cout
  - cout is tied to the standard output, usually the screen

```
float length;

cout << ”Please enter length in inches: ”;
cin >> length;
cout << ”Length in cm is ” << length * 2.54 << endl;
```

C and C++ programs

```
//C
#include <stdio.h>

int main (void)
{
    int entry;

    printf ("Enter a number ");
    scanf("%d", &entry);
    printf ("You entered %d", entry);
    return 0;
}
```

```
//C++
#include <iostream>

int main (void)
{
    int entry;

    std::cout << "Enter a number ";
    std::cin >> entry;
    std::cout << "You entered " << entry;
    return 0;
}
```

Any fool can write code that a computer can understand.

Good programmers write code that humans can understand.

~Martin Fowler