CS1500 Algorithms and Data Structures for Engineering, FALL 2012

Virgil Pavlu, vip@ccs.neu.edu
Jose Annunziato, jannunzi@gmail.com

Rohan Garg
Morteza Dilgir cs1500hw@gmail.com
Huadong Li

http://www.ccs.neu.edu/home/vip/teach/Cpp_ENG/index.html

Intro to Programming
C++ basics

Some slides from Chinmoy Dutta

computer systems

computers are programmable - that's why they are so useful

hardware components - physical components a computer is made of.

software components - **programs** that run on a computer.

hardware

CPU - Central Processing Unit

- CU - Control Unit
- ALU - Arithmetic and Logic Unit

main memory - RAM

secondary storage - hard drive, optical drive ...

input devices - keyboard, mouse, microphone, webcam ...

output devices - monitor, printer ...

software

operating systems - manages the hardware devices and control their processes

applications - solve user specific problems

program

a set of step-by-step instructions that tells a computer how to perform a given task

machine language

a computer executes machine language instructions

fetch/decode/execute cycle

a machine instruction is a long sequence of 0's and 1's :-(

we need something more sane!!!

programming language

a high-level language of giving instruction to computers

BASIC, FORTRAN, COBOL, PASCAL, C, **C++**, Java, Javascript, Python ...

Pascal

```
  .  (*****************************************************************************
 * A simple bubble sort program.  Reads integers, one per line, and prints   *
 * them out in sorted order.  Blows up if there are more than 49.            *
 *****************************************************************************)
PROGRAM Sort(input, output);
    CONST
        (* Max array size. *)
        MaxElts = 50;
    TYPE
        (* Type of the element array. *)
        IntArrType = ARRAY [1..MaxElts] OF Integer;

    VAR
        (* Indexes, exchange temp, array size. *)
        i, j, tmp, size: integer;

        (* Array of ints *)
        arr: IntArrType;

    (* Read in the integers. *)
    PROCEDURE ReadArr(VAR size: Integer; VAR a: IntArrType);
        BEGIN
            size := 1;
            WHILE NOT eof DO BEGIN
                readln(a[size]);
                IF NOT eof THEN
                    size := size + 1
            END
        END;

    BEGIN
        (* Read *)
        ReadArr(size, arr);
```

Ruby

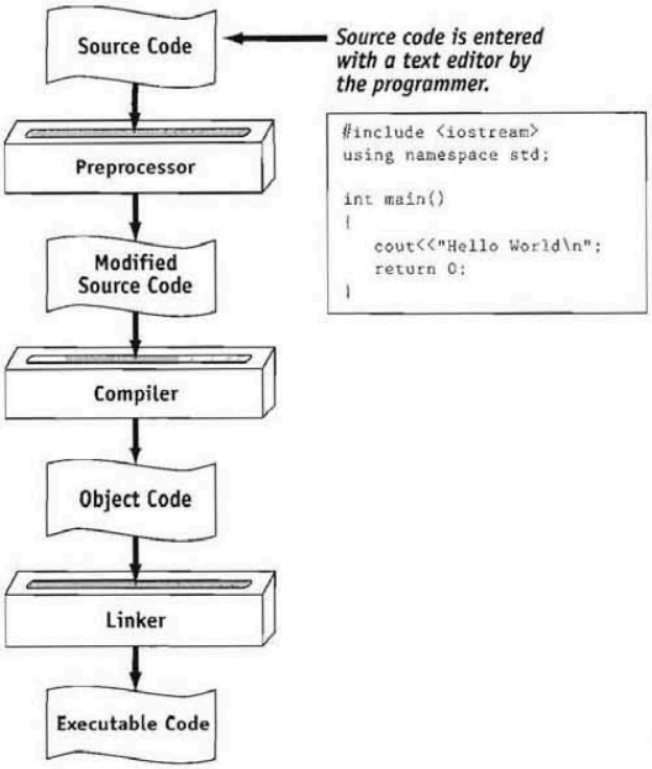
```
  # Simple for loop using a range.
for i in (1..4)
    print i," "
end
print "\n"

for i in (1...4)
    print i," "
end
print "\n"

# Running through a list (which is what they do).
items = [ 'Mark', 12, 'goobers', 18.45 ]
for it in items
    print it, " "
end
print "\n"

# Go through the legal subscript values of an array.
for i in (0...items.length)
```

Matlab

```
% script demonstrates how to print a line containing
both:
%  - literal text
%  - text converted from a numeric variable

r = input('Enter radius: ');
a = pi*r^2;
disp(['radius = ' num2str(r) ' area = ' num2str(a)]);
```

Our first C++ program

```
// this program expresses our
feelings

# include <iostream>

using namespace std;

int main(){

  cout << "I hate learning C++ at
  8AM in the morning" << endl;

  return 0;
```



compilation

Algorithm = Solution

- How do we conceptually design programs?
  - pseudocode = conceptual program, human language

```
initialize j=0,g=1;
for i = 1 to 10
        j=j+i
        g = g*i

print "sum is j, product is g"
```

C++ basics

variables

locations in memory

they have

identifier (name, case sensitive)

- restrictions on names

address

data type (this also determines size)

value (this can vary as program runs)

literals

also known as constants (contrast with variables)

a fixed value of a given data type which does not change

they can be assigned to variables

```
x=1500
```

data types

A way of categorizing variables and literals

this determines how much size is needed to store a value of a given data type

this also determines which operations are permitted on a value of a given data type

range of a data type

the set of values a data type can represent

more the space allocated for a data type, larger is the range (obvious)

integer data types

short (2 bytes)

unsigned short (2 bytes)

int (4 bytes)

unsigned int (or simply unsigned) (4 bytes)

long (4 bytes)

unsigned long (4 bytes)

floating -point data types

float (4 bytes)

double (8 bytes)

long double (8 bytes)

notes on size

Note that signed and unsigned variants are allocated equal amount of space, and hence their ranges are of the same size

the given sizes are typical; amount of space allocated may vary with system; only guarantee is that int is at least as large as short, long at least as large as int, double at least as large as float, and long double at least as large as double

character data type

char (1 byte)

represented as 1 byte integers, hence we can treat them as numbers!

conversion between the numeric value and the character done using ASCII table

boolean data type

bool

range consists of only two values

- true - represented as 1
- false - represented as 0

variable declaration

<data type> <variable identifier>;

```
int num;
double score=0;
char letter='A';
char letter2 =66;
bool flag;
```

assignment and initialization

assignment

num = 10;

letter = 'A'; (note the single quotes)

initialization (assignment while declaring)

int num = 10;

char letter = 'A';

named constants

just like variables (with identifier, address, data type and value), but the value is initialized at the time of declaration and cannot be changed

declared with keyword - const

```
const PI = 3.1415;
```

note the initialization

typically names are chosen uppercase

why use named constants

do not need to modify code at several places when the value changes (like the interest rate), just need to modify the initialization

no need to write values like that of PI repeatedly; less chance of error

makes code readable and easy to understand

binary representation (8 bits)

- positives: sign bit=0

```
  0 = 00000000
  1 = 00000001
  2 = 00000010
  3 = 00000011
  4 = 00000100
....
127 = 01111111
```

- negatives: sign bit=1, stands for -128 "two complement"

```
-128 = 10000000
-127 = -128+1 =10000001
-126 = -128+2 =10000010
-125 = -128+3 =10000011
 ...
...
```

  - range : -128 to 127