C++ is Fun – Part Deux
at Turbine/Warner Bros.!

Russell Hanson

Thanks for your patience last time, really

- We covered a lot of material quickly
- We are going to go back and use some of the topics introduced, slowly, carefully, and with examples you can try out! :)

Getting Familiar with Thinking about Programming

-- Programs can be used to accomplish certain tasks
-- The way that the tasks are accomplished usually consists of a set of instructions
-- The way this set of instructions is carried out uses basic data types like Arrays, Vectors, and Lists

-- Programs are little **models** of the world
-- They are essentially self-contained, but can interact with outside data, like weather data, stock data, game data, social network data, etc.

-- The data is stored in memory, either persistent like the hard disk, or volatile/non-persistent like RAM.

-- Interacting with a user is often important for a program. Often the user may not know how to interact with the program. The program often needs to check for multiple formats of input, and massage it into the right form for the program to perform its function.
-- A graphical user interface (GUI) can be programmed to allow the user to interact not on the console or command line, but with a more rich visual environment like we are accustomed to.

Programs and Programming Languages

CONCEPT: A program is a set of instructions a computer follows in order to perform a task. A programming language is a special language used to write computer programs.

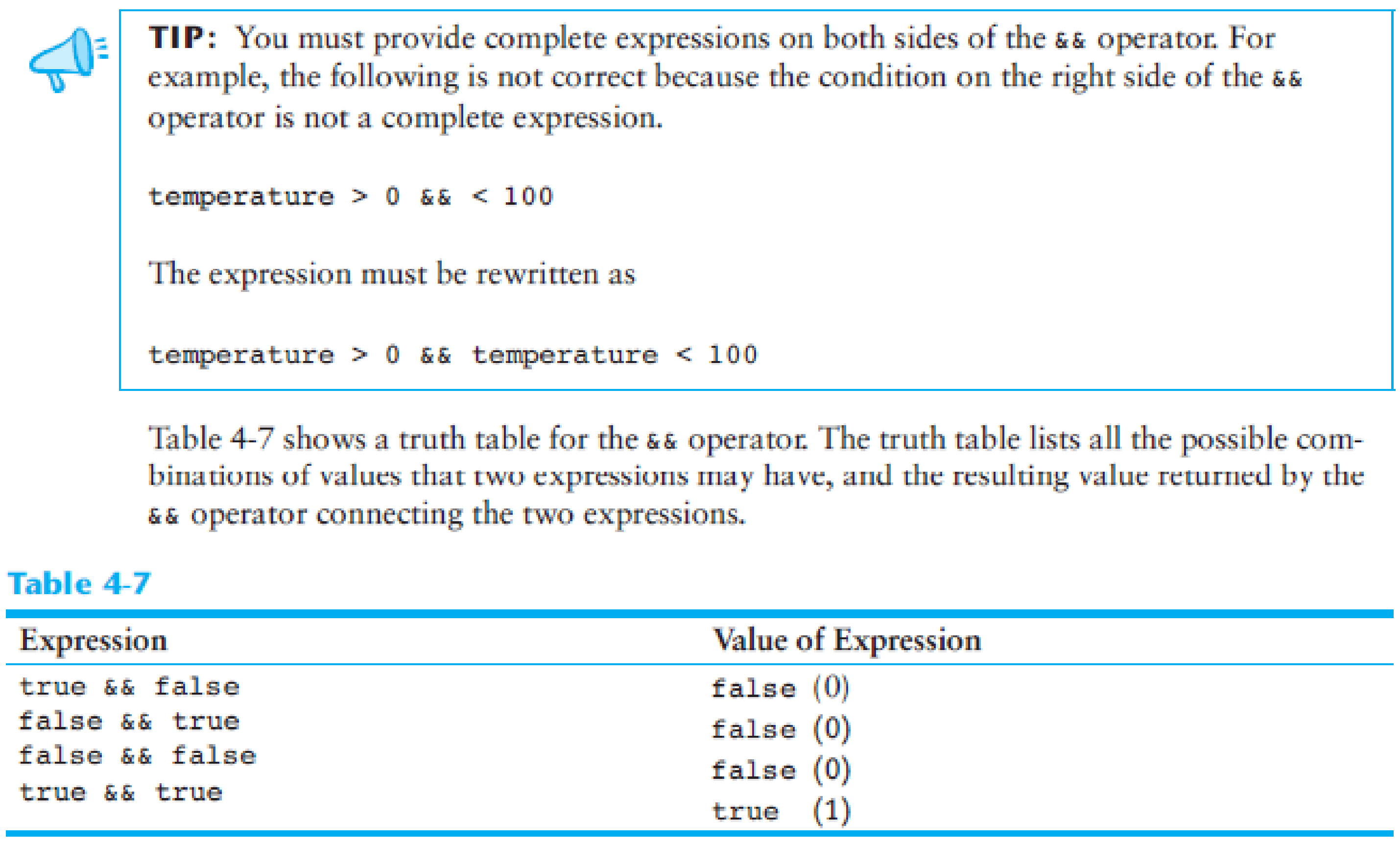
What Is a Program?

Computers are designed to follow instructions. A computer program is a set of instructions that tells the computer how to solve a problem or perform a task. For example, suppose we want the computer to calculate someone's gross pay. Here is a list of things the computer should do:

1. Display a message on the screen asking "How many hours did you work?"
2. Wait for the user to enter the number of hours worked. Once the user enters a number, store it in memory.
3. Display a message on the screen asking "How much do you get paid per hour?"
4. Wait for the user to enter an hourly pay rate. Once the user enters a number, store it in memory.
5. Multiply the number of hours by the amount paid per hour, and store the result in memory.
6. Display a message on the screen that tells the amount of money earned. The message must include the result of the calculation performed in Step 5.

Collectively, these instructions are called an *algorithm*. An algorithm is a set of well-defined steps for performing a task or solving a problem. Notice these steps are sequentially ordered. Step 1 should be performed before Step 2, and so forth. It is important that these instructions be performed in their proper sequence.

The logical AND operator &&



TIP: You must provide complete expressions on both sides of the `&&` operator. For example, the following is not correct because the condition on the right side of the `&&` operator is not a complete expression.

```
temperature > 0 && < 100
```

The expression must be rewritten as

```
temperature > 0 && temperature < 100
```

Table 4-7 shows a truth table for the `&&` operator. The truth table lists all the possible combinations of values that two expressions may have, and the resulting value returned by the `&&` operator connecting the two expressions.

Table 4-7

Expression	Value of Expression
`true && false`	`false` (0)
`false && true`	`false` (0)
`false && false`	`false` (0)
`true && true`	`true` (1)

The || Operator

The || operator is known as the logical OR operator. It takes two expressions as operands and creates an expression that is true when either of the sub-expressions are true. Here is an example of an if statement that uses the || operator:

```
if (temperature < 20 || temperature > 100)
   cout << "The temperature is in the danger zone.";
```

The cout statement will be executed if temperature is less than 20 OR temperature is greater than 100. If either relational test is true, the entire expression is true and the cout statement is executed.

TIP: You must provide complete expressions on both sides of the || operator. For example, the following is not correct because the condition on the right side of the || operator is not a complete expression.

```
temperature < 0 || > 100
```

The expression must be rewritten as

```
temperature < 0 || temperature > 100
```

Table 4-8 shows a truth table for the || operator.

Table 4-8

Expression	Value of the Expression
true || false	true (1)
false || true	true (1)
false || false	false (0)
true || true	true (1)

All it takes for an OR expression to be true is for one of the sub-expressions to be true. It doesn't matter if the other sub-expression is false or true.

Let’s write a program that uses the OR (||) or AND (&&) operators

```
#include "stdafx.h"
#include <string>
#include <iostream>
#include <map>
#include <utility>
using namespace std;
int main()
{
         string Apple = "Apple";
         string Orange = "Orange";

         string myitem = "Apple";
         if (myitem == Apple || myitem == Orange)
                  cout << "It's a fruit!\n";
         else cout << "It's not a fruit!\n";

         string myitem2 = "Cat";
         if (myitem2 == Apple || myitem2 == Orange)
                  cout << "It's a fruit!\n";
         else cout << "It's not a fruit!\n";
         return 0;
}
```

The ! Operator

The ! operator performs a logical NOT operation. It takes an operand and reverses its truth or falsehood. In other words, if the expression is true, the ! operator returns false, and if the expression is false, it returns true. Here is an `if` statement using the ! operator:

```
if (!(temperature > 100))
   cout << "You are below the maximum temperature.\n";
```

First, the expression `(temperature > 100)` is tested to be true or false. Then the ! operator is applied to that value. If the expression `(temperature > 100)` is true, the ! operator returns false. If it is false, the ! operator returns true. In the example, it is equivalent to asking "is the temperature not greater than 100?"

Table 4-9 shows a truth table for the ! operator.

Table 4-9

Expression	Value of the Expression
`!true`	`false (0)`
`!false`	`true  (1)`

Program 4-17 performs the same task as Program 4-16. The `if` statement, however, uses the ! operator to determine if the user does *not* make at least $35,000 or has *not* been on the job more than five years.

Program 4-17

```
// This program demonstrates the logical ! operator.
#include <iostream>
using namespace std;

int main()
{
   // Constants for minimum income and years
   const double MIN_INCOME = 35000.0;
   const int MIN_YEARS = 5;

   double income;   // Annual income
   int years;       // Years at the current job

   // Get the annual income
   cout << "What is your annual income? ";
   cin >> income;

   // Get the number of years at the current job.
   cout << "How many years have you worked at "
        << "your current job? ";
   cin >> years;
```

Let’s write a program that uses the NOT or ! or != operator

```
#include "stdafx.h"
#include <string>
#include <iostream>
#include <map>
#include <utility>
using namespace std;
const int TRUE = 1;
const int FALSE = 0;
int main() {
        string Apple = "Apple";
        string Orange = "Orange";

        if (Apple != Orange)
                cout << "That was predictable" << endl;

        cout << "Is this false? " << false << endl;
        cout << "Is this true? " << true << endl;
        while (!FALSE)
                cout << "Quite true" << endl;
                // Control-C will abort the program
        return 0;
}
```

Table 3-6 Algebraic and C++ Expressions

Algebraic Expression	C++ Expression
$y = 3\frac{x}{2}$	`y = x / 2 * 3;`
$z = 3bc + 4$	`z = 3 * b * c + 4;`
$a = \frac{3x + 2}{4a - 1}$	`a = (3 * x + 2) / (4 * a - 1)`

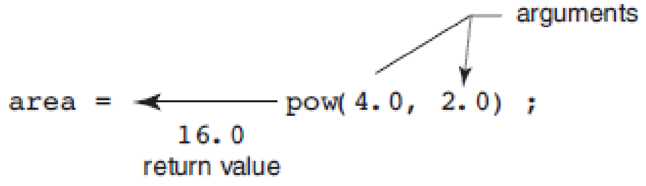
No Exponents Please!

Unlike many programming languages, C++ does not have an exponent operator. Raising a number to a power requires the use of a *library function*. The C++ library isn't a place where you check out books, but a collection of specialized functions. Think of a library function as a "routine" that performs a specific operation. One of the library functions is called `pow`, and its purpose is to raise a number to a power. Here is an example of how it's used:

```
area = pow(4.0, 2.0);
```

This statement contains a *call* to the `pow` function. The numbers inside the parentheses are *arguments*. Arguments are data being sent to the function. The `pow` function always raises the first argument to the power of the second argument. In this example, 4 is raised to the power of 2. The result is *returned* from the function and used in the statement where the function call appears. In this case, the value 16 is returned from `pow` and assigned to the variable `area`. This is illustrated in Figure 3-3.

Figure 3-3



The statement `area = pow(4.0, 2.0)` is equivalent to the following algebraic statement:

area = 4^2

Here is another example of a statement using the `pow` function. It assigns 3 times 6^3 to `x`:

```
x = 3 * pow(6.0, 3.0);
```

Multiple Assignment and Combined Assignment

CONCEPT: Multiple assignment means to assign the same value to several variables with one statement.

C++ allows you to assign a value to multiple variables at once. If a program has several variables, such as `a`, `b`, `c`, and `d`, and each variable needs to be assigned a value, such as 12, the following statement may be constructed:

```
a = b = c = d = 12;
```

The value 12 will be assigned to each variable listed in the statement.*

Combined Assignment Operators

Quite often, programs have assignment statements of the following form:

```
number = number + 1;
```

The expression on the right side of the assignment operator gives the value of `number` plus 1. The result is then assigned to `number`, replacing the value that was previously stored there. Effectively, this statement adds 1 to `number`. In a similar fashion, the following statement subtracts 5 from `number`.

```
number = number - 5;
```

If you have never seen this type of statement before, it might cause some initial confusion because the same variable name appears on both sides of the assignment operator. Table 3-8 shows other examples of statements written this way.

Table 3-8 (Assume x = 6)

Statement	What It Does	Value of x After the Statement
`x = x + 4;`	Adds 4 to x	10
`x = x - 3;`	Subtracts 3 from x	3
`x = x * 10;`	Multiplies x by 10	60
`x = x / 2;`	Divides x by 2	3
`x = x % 4`	Makes x the remainder of x / 4	2

These types of operations are very common in programming. For convenience, C++ offers a special set of operators designed specifically for these jobs. Table 3-9 shows the *combined assignment operators*, also known as *compound operators*, and *arithmetic assignment operators*.

Table 3-9

Operator	Example Usage	Equivalent to
`+=`	`x += 5;`	`x = x + 5;`
`-=`	`y -= 2;`	`y = y - 2;`
`*=`	`z *= 10;`	`z = z * 10;`
`/=`	`a /= b;`	`a = a / b;`
`%=`	`c %= 3;`	`c = c % 3;`

As you can see, the combined assignment operators do not require the programmer to type the variable name twice. Also, they give a clear indication of what is happening in the

Let’s write a program that uses the modulo operator (remainder) and assigns it to another value

```
#include "stdafx.h"
#include <string>
#include <iostream>
#include <map>
#include <utility>
using namespace std;
int main()
{
        int     dividend = 0;
        int divisor = 0;
        cout << "what's the remainder when you divide:" << endl;
        cin >> dividend;
        cout << "by:" << endl;
        cin >> divisor;
        cout << dividend % divisor;

        return 0;
}
```

Return to control structures for, and while

```
#include "stdafx.h"
// custom countdown using while
#include <iostream>
using namespace std;
int main ()
{
  int n;
  cout << "Enter the starting number > ";
  cin >> n;
  while (n>0) {
    cout << n << ", ";
    --n;
  }
  cout << "FIRE!\n";
  return 0;
}
```

For loop

```cpp
// PartDeux.cpp : Defines the entry point for the console application.
//
#include "stdafx.h"
#include <iostream>
#include <vector>
#include <string>
#include <sstream>
using namespace std;
int main() {
         int i = 0;
         string myanimal = "Elephant";
         for (i=0; i < myanimal.size(); i++)
                  cout << myanimal[i] << endl;
                 // cout << myanimal[i] << " ";
}
```

```
$ ./PartDeux.exe
E
l
e
p
h
a
n
t
```

Adding data to vectors, be careful of the maximum vector size!

```
// PartDeux.cpp : Defines the entry point for the console application.
//
#include "stdafx.h"
#include <iostream>
#include <vector>
#include <string>
using namespace std;
int main() {
   vector<string> myvector;
   myvector.push_back("The number is 10");
   myvector.push_back("The number is 20");
   myvector.push_back("The number is 30");
   cout << "Loop by index:" << endl;
   int i;
   for(i=0; i < myvector.size(); i++){
      cout << myvector[i] << endl;
   }
   cout << "What happens when we try to access an index greater than the vector's size?
   cout << "This happens:" << myvector[myvector.size()] << endl;
   cout << "Ouch.";
}
```

Joining two strings together in C++ is called concatenation, use the *stringstream*

```
#include <sstream>
#include <string>
std::stringstream ss;
ss << "Hello, world, " << myInt << niceToSeeYouString;
std::string s = ss.str();
```

```
#include "stdafx.h"
#include <iostream>
#include <vector>
#include <string>
#include <sstream>
using namespace std;
int main() {
         int myInt = 99; string niceToSeeYouString = " nice to see you!";
         stringstream ss;
         ss << "Hello, world, " << myInt << niceToSeeYouString;
         string s = ss.str();
         cout << s << endl;
}
```

Algorithms

- Algorithms can be used to sort and to search!

Quicksort
Quicksort is a fast sorting algorithm, which is used not only for educational purposes, but widely applied in practice. On the average, it has O(n log n) complexity, making quicksort suitable for sorting big data volumes. The idea of the algorithm is quite simple and once you realize it, you can write quicksort as fast as bubble sort.
Algorithm
The divide-and-conquer strategy is used in quicksort. Below the recursion step is described: **Choose a pivot value.** We take the value of the middle element as pivot value, but it can be any value, which is in range of sorted values, even if it doesn't present in the array.
Partition. Rearrange elements in such a way, that all elements which are lesser than the pivot go to the left part of the array and all elements greater than the pivot, go to the right part of the array. Values equal to the pivot can stay in any part of the array. Notice, that array may be divided in non-equal parts.
Sort both parts. Apply quicksort algorithm recursively to the left and the right parts.

Binary search algorithm

Generally, to find a value in unsorted array, we should look through elements of an array one b
one, until searched value is found. In case of searched value is absent from array, we go through al
elements. In average, complexity of such an algorithm is proportional to the length of the array.
Situation changes significantly, when array is sorted. If we know it, random access capability ca
be utilized very efficientlyto find searched value quick. Cost of searching algorithm reduces to binar
logarithm of the array length. For reference, $\log_2(1\ 000\ 000) \approx 20$. It means, that **in worst case**
algorithm makes 20 steps to find a value in sorted array of a million elements or to say, that i
doesn't present it the array.

Algorithm

Algorithm is quite simple. It can be done either recursively or iteratively:

1.get the middle element;

2.if the middle element equals to the searched value, the algorithm stops;

3.otherwise, two cases are possible:

1. searched value is less, than the middle element. In this case, go to the step 1 for the part of the array, before middle element.
2. searched value is greater, than the middle element. In this case, go to the step 1 for the part of the array, after middle element.

Now we should define, when iterations should stop. First case is when searched element is found. Second one is when subarray has no elements. In this case, we can conclude, that searched value doesn't present in the array.**Examples**

Example 1. Find 6 in {-1, 5, 6, 18, 19, 25, 46, 78, 102, 114}.

Step 1 (middle element is 19 > 6):	-1	5	6	18	**19**	25	46	78	102	114
Step 2 (middle element is 5 < 6):	-1	**5**	6	18	19	25	46	78	102	114
Step 3 (middle element is 6 == 6):	-1	5	**6**	18	19	25	46	78	102	114

Hash functions in programming

- Used to find the value of items *instantaneously*, without searching, in so-called “constant-time”
- Say you have an array with the heights of 10 basketball players, but you just want Henry’s height, but you don’t know the index in the array?

Now assumming you want a hash, and want something **blazing fast** that would work in your case, because your strings are just 6 chars long you could use this magic:

```
size_t precision = 2; //change the precision with this
size_t hash(const char* str)
{
   return (*(size_t*)str)>> precision;
}
```

CRC is for slowpokes ;)

Explanation: This works by casting the contents of the string pointer to "look like" a size_t (int32 or int64 based on the optimal match for your hardware). So the contents of the string are interpreted as a raw number, no worries about characters anymore, and you then bit-shift this the precision needed (you tweak this number to the best performance, I've found 2 works well for hashing strings in set of a few thousands).

Also the really neat part is any decent compiler on modern hardware will hash a string like this in 1 assembly instruction, hard to beat that ;)

This simple polynomial works surprisingly well. I got it from Paul Larson of Microsoft Research who studied a wide variety of hash functions and hash multipliers.

```
unsigned hash(const char* s)
{
    unsigned h = SALT;
    while (*s)
        h = h * 101 + (unsigned char*) *s++;
    return h;
}
```

SALT should be initialized to some randomly chosen value before the hashtable is created to defend against hash table attacks. If this isn't an issue for you, just use 0.

The size of the table is important too, to minimize collisions. Sounds like yours is fine.

C++ STL MAP and multiMAP:

Description, use and examples of C++ STL "pair", "map" and "multimap" associative containers. The STL associative container class is a variable sized container which supports retrieval of an element value given a search key.

- **STL pair:** A container which holds two values. The data types may or may not be different.
- **STL map:** Associative key-value pair held in balanced binary tree structure. Each key is unique.
- **STL multimap:** Same as an STL map except that duplicate keys are allowed.

```cpp
// PartDeux.cpp : Defines the entry point for the console application.
//
#include "stdafx.h"
#include <string>
#include <iostream>
#include <map>
#include <utility>
using namespace std;
int main()
{
   map<int, string> Employees;
   map<string, int> EmployeesbyName;
   EmployeesbyName["Shawn"] = 222;
   cout << "EmployeesbyName[\"Shawn\"] = " << EmployeesbyName["Shawn"] << endl;
   // 1) Assignment using array index notation
   Employees[5234] = "Mike C.";
   Employees[3374] = "Charlie M.";
   Employees[1923] = "David D.";
   Employees[7582] = "John A.";
   Employees[5328] = "Peter Q.";
   cout << "Employees[3374] = " << Employees[3374] << endl << endl;
   cout << "Map size: " << Employees.size() << endl;
   for( map<int,string>::iterator ii=Employees.begin(); ii!=Employees.end(); ++ii){
       cout << (*ii).first << ": " << (*ii).second << endl;
   }
   return 0;
}
```

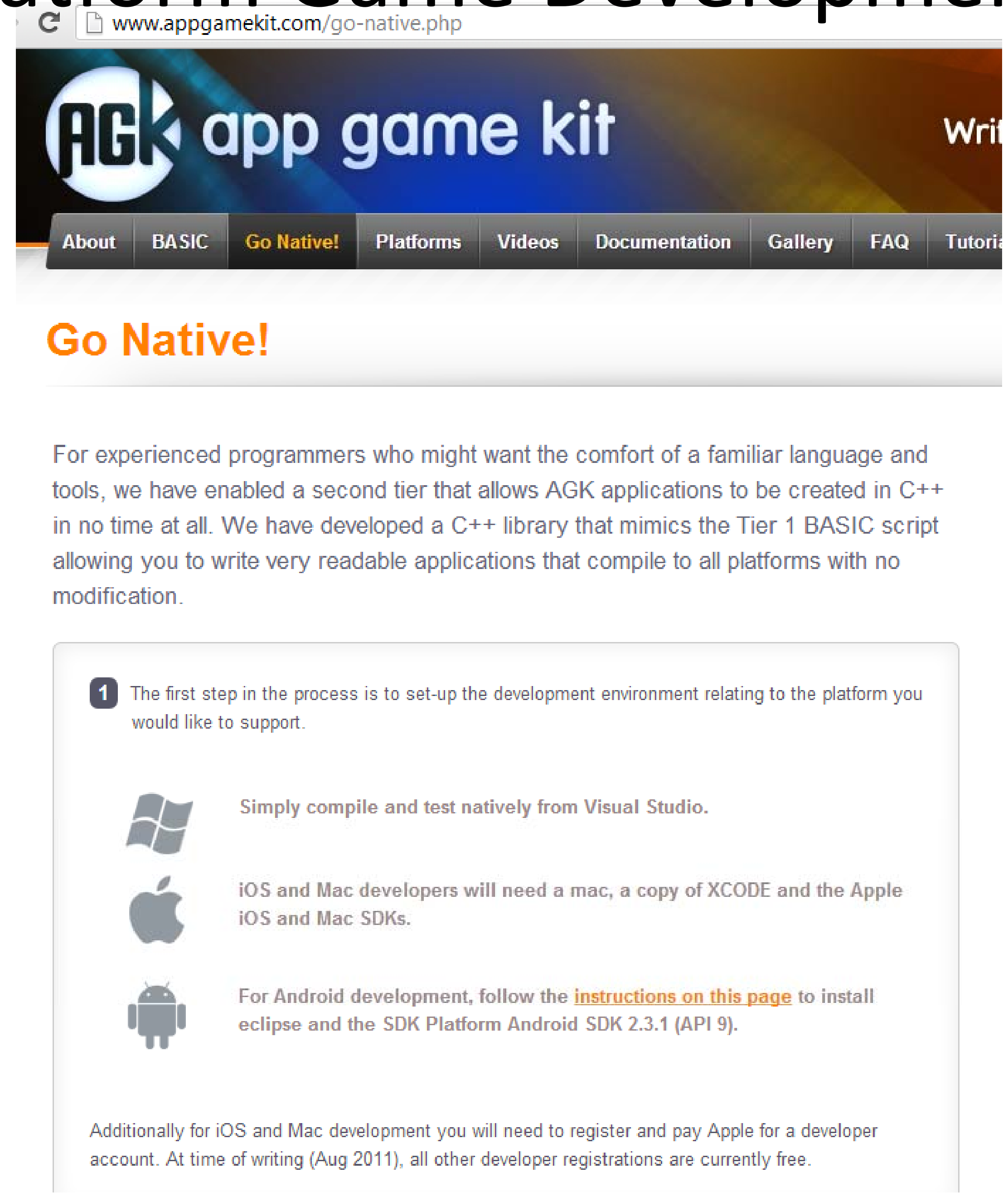
Example: STL map:

Sparse array example: (why hold space for thousands of elements when all we have is five)

```
#include <string.h>
#include <iostream>
#include <map>
#include <utility>

using namespace std;

int main()
{
   map<int, string> Employees;

   // 1) Assignment using array index notation
   Employees[5234] = "Mike C.";
   Employees[3374] = "Charlie M.";
   Employees[1923] = "David D.";
   Employees[7582] = "John A.";
   Employees[5328] = "Peter Q.";

   cout << "Employees[3374]=" << Employees[3374] << endl << endl;

   cout << "Map size: " << Employees.size() << endl;

   for( map<int,string>::iterator ii=Employees.begin(); ii!=Employees.end();
++ii)
   {
       cout << (*ii).first << ": " << (*ii).second << endl;
   }
}
```

Compile: `g++ testMap.cpp`
Run: `./a.out`

Cross-platform Game Development API's



Go Native!

For experienced programmers who might want the comfort of a familiar language and tools, we have enabled a second tier that allows AGK applications to be created in C++ in no time at all. We have developed a C++ library that mimics the Tier 1 BASIC script allowing you to write very readable applications that compile to all platforms with no modification.

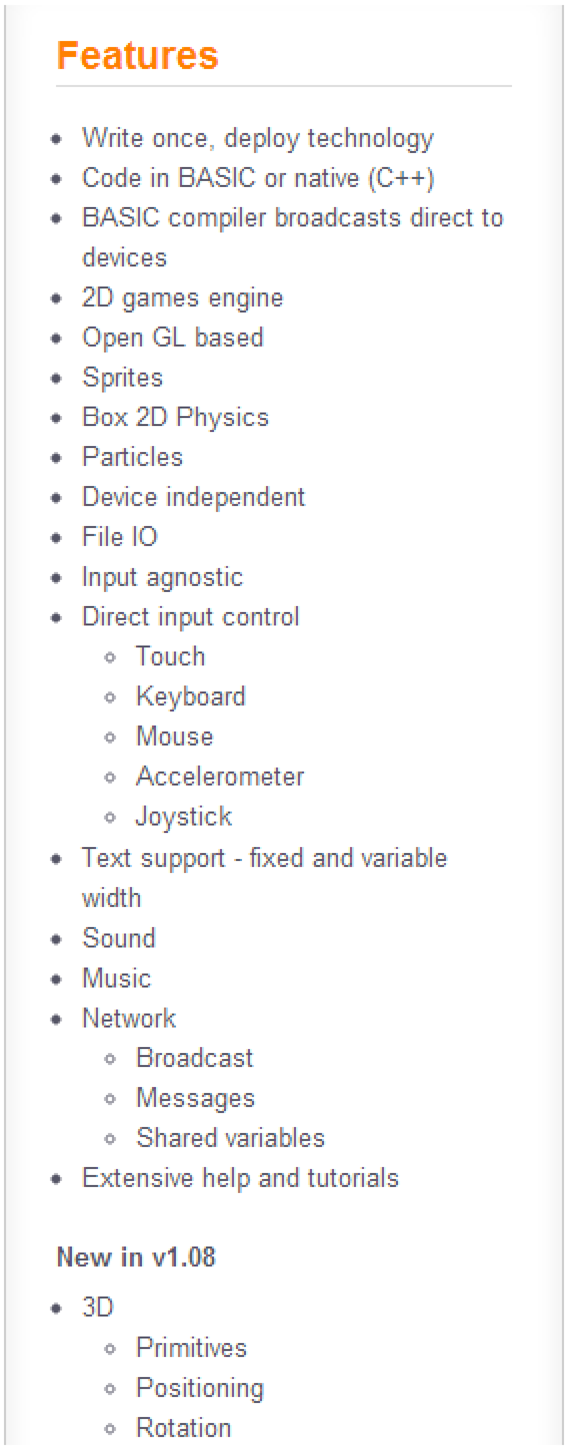
1 The first step in the process is to set-up the development environment relating to the platform you would like to support.

Simply compile and test natively from Visual Studio.

iOS and Mac developers will need a mac, a copy of XCODE and the Apple iOS and Mac SDKs.

For Android development, follow the instructions on this page to install eclipse and the SDK Platform Android SDK 2.3.1 (API 9).

Additionally for iOS and Mac development you will need to register and pay Apple for a developer account. At time of writing (Aug 2011), all other developer registrations are currently free.

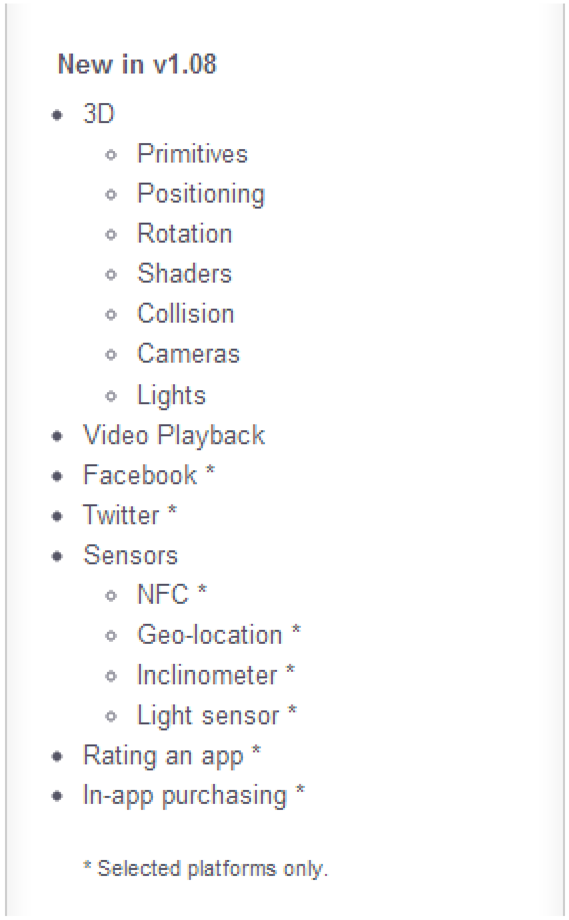


Features

- Write once, deploy technology
- Code in BASIC or native (C++)
- BASIC compiler broadcasts direct to devices
- 2D games engine
- Open GL based
- Sprites
- Box 2D Physics
- Particles
- Device independent
- File IO
- Input agnostic
- Direct input control
  - Touch
  - Keyboard
  - Mouse
  - Accelerometer
  - Joystick
- Text support - fixed and variable width
- Sound
- Music
- Network
  - Broadcast
  - Messages
  - Shared variables
- Extensive help and tutorials

New in v1.08

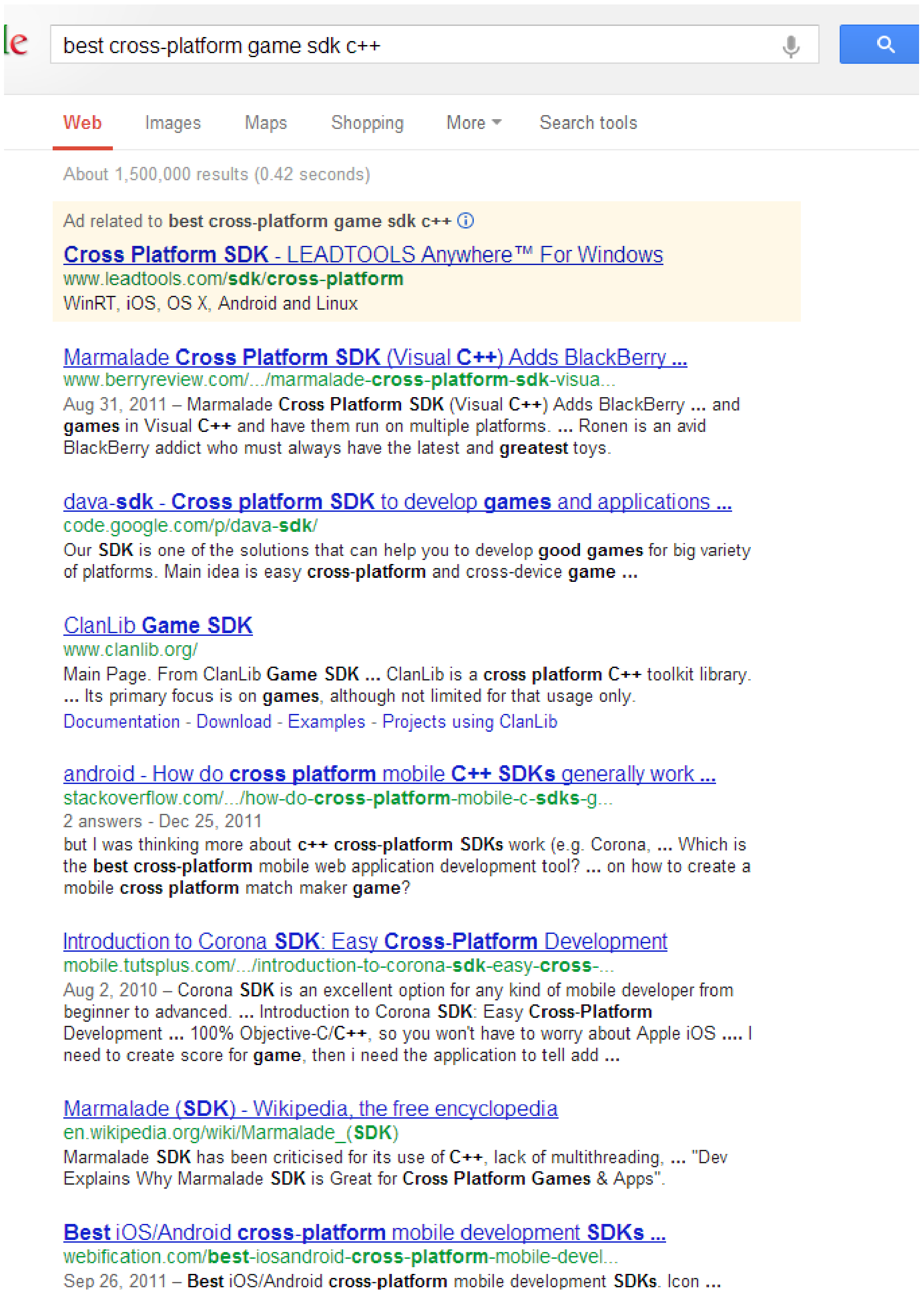
- 3D
  - Primitives
  - Positioning
  - Rotation



New in v1.08

- 3D
  - Primitives
  - Positioning
  - Rotation
  - Shaders
  - Collision
  - Cameras
  - Lights
- Video Playback
- Facebook *
- Twitter *
- Sensors
  - NFC *
  - Geo-location *
  - Inclinometer *
  - Light sensor *
- Rating an app *
- In-app purchasing *

* Selected platforms only.



best cross-platform game sdk c++

Web Images Maps Shopping More Search tools

About 1,500,000 results (0.42 seconds)

Ad related to **best cross-platform game sdk c++**

Cross Platform SDK - LEADTOOLS Anywhere™ For Windows
www.leadtools.com/**sdk**/**cross-platform**
WinRT, iOS, OS X, Android and Linux

Marmalade **Cross Platform SDK** (Visual **C++**) Adds BlackBerry **...**
www.berryreview.com/.../marmalade-**cross-platform-sdk**-visua...
Aug 31, 2011 – Marmalade **Cross Platform SDK** (Visual **C++**) Adds BlackBerry **...** and **games** in Visual **C++** and have them run on multiple platforms. **...** Ronen is an avid BlackBerry addict who must always have the latest and **greatest** toys.

dava-**sdk** - **Cross platform SDK** to develop **games** and applications **...**
code.google.com/p/dava-**sdk**/
Our **SDK** is one of the solutions that can help you to develop **good games** for big variety of platforms. Main idea is easy **cross-platform** and cross-device **game ...**

ClanLib **Game SDK**
www.clanlib.org/
Main Page. From ClanLib **Game SDK ...** ClanLib is a **cross platform C++** toolkit library. **...** Its primary focus is on **games**, although not limited for that usage only.
Documentation - Download - Examples - Projects using ClanLib

android - How do **cross platform** mobile **C++ SDKs** generally work **...**
stackoverflow.com/.../how-do-**cross-platform**-mobile-c-**sdks**-g...
2 answers - Dec 25, 2011
but I was thinking more about **c++ cross-platform SDKs** work (e.g. Corona, **...** Which is the **best cross-platform** mobile web application development tool? **...** on how to create a mobile **cross platform** match maker **game**?

Introduction to Corona **SDK**: Easy **Cross-Platform** Development
mobile.tutsplus.com/.../introduction-to-corona-**sdk**-easy-**cross**-...
Aug 2, 2010 – Corona **SDK** is an excellent option for any kind of mobile developer from beginner to advanced. **...** Introduction to Corona **SDK**: Easy **Cross-Platform** Development **...** 100% Objective-C/**C++**, so you won't have to worry about Apple iOS **....** I need to create score for **game**, then i need the application to tell add **...**

Marmalade (**SDK**) - Wikipedia, the free encyclopedia
en.wikipedia.org/wiki/Marmalade_(**SDK**)
Marmalade **SDK** has been criticised for its use of **C++**, lack of multithreading, **...** "Dev Explains Why Marmalade **SDK** is Great for **Cross Platform Games** & Apps".

Best iOS/Android **cross-platform** mobile development **SDKs ...**
webification.com/**best**-iosandroid-**cross-platform**-mobile-devel...
Sep 26, 2011 – **Best** iOS/Android **cross-platform** mobile development **SDKs**. Icon **...**

Homework for next Monday (pick 2, minimum)

1) Write a program that uses the modulus operator to determine if a number is odd or even. If the number is evenly divisible by 2, it is an even number. A remainder indicates it is odd. The number can be input by the user or read from a file.
2) Write an if statement that performs the following logic: if the variable *sales* is greater than 50,000, then assign 0.25 to the *commissionRate* variable, and assign 250 to the *bonus* variable.
3) Accept two strings as input at the prompt/command line, such as "Big" and "Apple." Join or concatenate the two words with a third word, such as "Ripe" and print the three words together with the third word the middle, "Big Ripe Apple".
4) Accept 5 integers on the command line, either all at once or separately. Save these to an array, vector, or list. Print the integers in the range 2 through 4, leaving off the first and the last. Bonus: Ask for the size of the array to be used, so it can be 5, 6, or 7 etc. Double Bonus: Allow a variable number of input numbers, stop input using a stop character or command the letter "s" say, then print all the input integers leaving off the first and the last.