CS101 Computer Programming and Utilization

Milind Sohoni

May 10, 2006

The story so far ...

- We have seen a basic programming language PCAL, for our calculator machine.
- This included commands like IF-ENDIF and REPEAT-UNTIL.
- We have seen how to write PCAL programs for some simple applications.

C++

We begin with C++, the programming language of our course.
www.cplusplus.com/doc/tutorial for reference.

The story so far ...

- We have seen a basic programming language PCAL, for our calculator machine.
- This included commands like IF-ENDIF and REPEAT-UNTIL.
- We have seen how to write PCAL programs for some simple applications.

C++

We begin with C++, the programming language of our course.
www.cplusplus.com/doc/tutorial for reference.

- download the file `CtoF.c` and view it.
- type `g++ CtoF.c` and get a new file a.out.
- type `./a.out`
- On being prompted, enter a number and observe the output.

Our First C++ Program CtoF.c

```
#include <iostream.h>
// this program takes a
// centigrade value as
// input and converts
// that into farenheit
int main()
{
  float C;
  float F;
  cout << "centigrades" << "\n";
  cin >> C;
  F=C*9/5+32;
  cout << F << "\n";
  return 0;
}
```

Recall the PCAL code:

```
M1=READIN 40
M2=M1*9/5+32
```

compare with

```
cin >> C;
F=C*9/5+32;
```

Our First C++ Program CtoF.c

```
#include <iostream.h>
// this program takes a
// centigrade value as
// input and converts
// that into farenheit
int main()
{
  float C;
  float F;
  cout << "centigrades" << "\n";
  cin >> C;
  F=C*9/5+32;
  cout << F << "\n";
  return 0;
}
```

Recall the PCAL code:

```
M1=READIN 40
M2=M1*9/5+32
```

compare with

```
cin >> C;
F=C*9/5+32;
```

Variables

- Memory registers can have names.

Our First C++ Program CtoF.c

```
#include <iostream.h>
// this program takes a
// centigrade value as
// input and converts
// that into farenheit
int main()
{
  float C;
  float F;
  cout << "centigrades" << "\n";
  cin >> C;
  F=C*9/5+32;
  cout << F << "\n";
  return 0;
}
```

Recall the PCAL code:

```
M1=READIN 40
M2=M1*9/5+32
```

compare with

```
cin >> C;
F=C*9/5+32;
```

Variables

- Memory registers can have names.

Largely true

- Every line ends with a ;.

Our First C++ Program CtoF.c

```
#include <iostream.h>
//
// comments about
// the program
//
int main()
{




  return 0;
}
```

Basic Structure

- The `include ..` tells us what family of commands the program will use.
- Lines beginning with \\ are ignored by the compiler. They just serve to make the code readable. Such lines are called comments and must be used extensively in your program.
- Every C++ program must have the `int main()` and the braces

```
{
    code
    return 0;
}
```

Our First C++ Program CtoF.c

```
#include <iostream.h>
//
// comments about
// the program
//
int main()
{
  float C;
  float F;




  return 0;
}
```

Declarations

- This tells us that there are two variables C and F. Both of them are floating point real numbers.
- Such statements are called the Declarations since they declare the type of the variables and their names.

Our First C++ Program CtoF.c

```
#include <iostream.h>
//
// comments about
// the program
//
int main()
{
  float C;
  float F;
  cout << "centigrades" << "\n";
  cin >> C;

  cout << F << "\n";
  return 0;
}
```

Input and Output

- Lets see the cout lines:

  ```
  cout << "centigrades" <<
  cout << F << \n;
  ```

  The first line tells the compiler to output (on the screen) the word centigrades and go to the next line.
  The second line throws out the value of F and goes to the next line.

- The cin line is:

  ```
  cin >> C;
  ```

  This makes the computer read your keyboard input and puts it into C.

Our First C++ Program CtoF.c

```
#include <iostream.h>
//
// comments about
// the program
//
int main()
{
  float C;
  float F;
  cout << "centigrades" << "\n";
  cin >> C;
  F=C*9/5+32;
  cout << F << "\n";
  return 0;
}
```

The Assignment Statement

- Causes the computation on the right to be assigned to the variable location named F.

A Variation intCtoF.c

```
#include <iostream.h>
//
// comments about
// the program
//
int main()
{
  int C;
  int F;
  cout << "centigrades" << "\n";
  cin >> C;
  F=C*9/5+32;
  cout << F << "\n";
  return 0;
}
```

A Variation intCtoF.c

```
#include <iostream.h>
//
// comments about
// the program
//
int main()
{
  int C;
  int F;
  cout << "centigrades" << "\n";
  cin >> C;
  F=C*9/5+32;
  cout << F << "\n";
  return 0;
}
```

We declare the variables F,C to be int (i.e., integers) and keep everything unchanged. We observe:

centigrades	4	14	27
float F,C	39.2	57.2	80.6
int F,C	39	57	80

A Variation intCtoF.c

```
#include <iostream.h>
//
// comments about
// the program
//
int main()
{
  int C;
  int F;
  cout << "centigrades" << "\n";
  cin >> C;
  F=C*9/5+32;
  cout << F << "\n";
  return 0;
}
```

We declare the variables F,C to be int (i.e., integers) and keep everything unchanged. We observe:

centigrades	4	14	27
float F,C	39.2	57.2	80.6
int F,C	39	57	80

What If

What is the output of the code when we declare F,C as int and change the assignment to:

$$F=C/5*9+32;$$

Can you explain? Does this matter when C,F were float?

Solving a Quadratic in C++ quad1.c

```
#include <iostream.h>
// solves a quadratic
int main()
{
  float A,B,C;
  float root1,root2,disc;
  cout << "A B C" << "\n";
  cin >> A >> B >> C;
  disc=B*B-4*A*C;
  disc=sqrt(disc);
  root1=(-B+disc)/(2*A);
  root2=(-B-disc)/(2*A);
  cout<<"root 1="<<root1<<"\n";
  cout<<"root 2="<<root2<<"\n";
  return 0;
}
```

Whats New?

- Intermediate variables `root1, root2, disc` are declared and used.
- See that C++ allows complicated expressions in the assignments for `root1, root2`.
- Note that the first `cout` prompts the user for A,B and C. The `cin` inputs three values separated by spaces and feeds them in the same order.

Solving a Quadratic in C++ quad1.c

```
#include <iostream.h>
// solves a quadratic
int main()
{
  float A,B,C;
  float root1,root2,disc;
  cout << "A B C" << "\n";
  cin >> A >> B >> C;
  disc=B*B-4*A*C;
  disc=sqrt(disc);
  root1=(-B+disc)/(2*A);
  root2=(-B-disc)/(2*A);
  cout<<"root 1="<<root1<<"\n";
  cout<<"root 2="<<root2<<"\n";
  return 0;
}
```

Whats New?

- `disc=sqrt(disc)` replaces the contents of `disc` with its square root.
- The second `cout` formats the output and writes it on the screen.

Solving a Quadratic in C++ quad1.c

```
#include <iostream.h>
// solves a quadratic
int main()
{
  float A,B,C;
  float root1,root2,disc;
  cout << "A B C" << "\n";
  cin >> A >> B >> C;
  disc=B*B-4*A*C;
  disc=sqrt(disc);
  root1=(-B+disc)/(2*A);
  root2=(-B-disc)/(2*A);
  cout<<"root 1="<<root1<<"\n";
  cout<<"root 2="<<root2<<"\n";
  return 0;
}
```

Whats New?

- `disc=sqrt(disc)` replaces the contents of `disc` with its square root.
- The second `cout` formats the output and writes it on the screen.

Try This

```
> ./a.out
A B C
1 1 1
root 1=nan
root 2=nan
```

Why has this happened?

This happened because $B^2 - 4AC = 1 - 3 = -2$. The calculator behind the computer konked while computing its square-root. What is to be done?:

```
#include <iostream.h>
// general square roots
int main()
{
  float r,x,y;
  cout << "real?" << "\n";
  cin >> r;
  if (r<0)  //new!!!
  {
    r=-r;
    x=sqrt(r);
    cout << x << "i" << "\n";
    return 0;
  };
  x=sqrt(r);
  cout << x << "\n";
  return 0;
}
```

Our Solution

mysqrt.c

This happened because $B^2 - 4AC = 1 - 3 = -2$. The calculator behind the computer konked while computing its square-root. What is to be done?:

```
#include <iostream.h>
// general square roots
int main()
{
  float r,x,y;
  cout << "real?" << "\n";
  cin >> r;
  if (r<0)  //new!!!
  {
    r=-r;
    x=sqrt(r);
    cout << x << "i" << "\n";
    return 0;
  };
  x=sqrt(r);
  cout << x << "\n";
  return 0;
}
```

Our Solution

mysqrt.c

The if Statement

This has the following form:

```
if (condition)
{
   code
};
```

The *condition* cab be any logical statement which returns true or false.

This happened because $B^2 - 4AC = 1 - 3 = -2$. The calculator behind the computer konked while computing its square-root.

```
#include <iostream.h>
// general square roots
int main()
{
  float r,x,y;
  cout << "real?" << "\n";
  cin >> r;
  if (r<0)  //new!!!
  {
    r=-r;
    x=sqrt(r);
    cout << x << "i" << "\n";
    return 0;
  };
  x=sqrt(r);
  cout << x << "\n";
  return 0;
}
```

In our case...

- if r is neagtive then we take the sqrt of its negative and print it with an i. The program then exits via return.
- if $r >= 0$ then execution proceeds the normal way.

If the return *within* the if were absent then control would go to:

```
x=sqrt(r);
cout << x << "\n";
return 0;
```

This would cause a double printing whenever $r < 0$.

The same program is better written as an if-else code as follows:

mysqrt2.c;

```
#include <iostream.h>
// general square roots
int main()
{
  float r,x,y;
  cout << "real?" << "\n";
  cin >> r;
  if (r<0)
  {
    r=-r;
    x=sqrt(r);
    cout << x << "i" << "\n";
  }   // NOTE no apostrophes
  else
  {
   x=sqrt(r);
   cout << x << "\n";
  };
  return 0;
}
```

The same program is better written as an if-else code as follows:

```
#include <iostream.h>
// general square roots
int main()
{
  float r,x,y;
  cout << "real?" << "\n";
  cin >> r;
  if (r<0)
  {
    r=-r;
    x=sqrt(r);
    cout << x << "i" << "\n";
  }   // NOTE no apostrophes
  else
  {
   x=sqrt(r);
   cout << x << "\n";
  };
  return 0;
}
```

mysqrt2.c;

In this case

- Note that there is no apostrophes after the if part. This indicates that there is an else part to the if.
- Note that there is a common return 0.
- if r is negative then we take the if part, and otherwise the else part. Thus only one of the code blocks is executed. This eliminates the need to have separate returns.

So far ...

- Program Structure: There are some include commands and then:

```
int main()
{
   code block
}
```

- Variables: Memory registers may have names. These must be words beginning with a non-numeral. Certain words are not allowed such as if, repeat, float.
- Declarations: Every variable must be declared to be of a certain type such as int, float. Operations must respect this type.
- Input and Output is enabled through cin, cout. Variable contents and strings may be manipulated in-order.
- Assignments are done by

```
var= expression;
```

- Note that every statement ends with a ;.

The If Statement Summary

The If statement is used as:

```
prevline;
if (condn)
{
   code block1
}
else
{
   code block2}
};
nextline;
```

The If Statement Summary

The If statement is used as:

```
prevline;
if (condn)
{
   code block1
}
else
{
   code block2}
};
nextline;
```

If the condn evaluates to true then the seqeunce is:

```
prevline;
code block1
nextline;
```

The If Statement Summary

The If statement is used as:

```
prevline;
if (condn)
{
   code block1
}
else
{
   code block2}
};
nextline;
```

If the condn evaluates to true then the seqeunce is:

```
prevline;
code block1
nextline;
```

otherwise it is:

```
prevline;
code block2
nextline;
```

Conditions are enclosed in brackets and must evaluate to either true or false.

```
if (A+B<C)       // clear enough
if (A+B==C)      // the double == separates it from assignments
if ((cond1) && (cond2)) // cond1 AND cond2
if ((cond1) || (cond2)) // cond1 OR cond2
if (!(condn))           // NOT condn
```

Assignment

- Write a C++ program to solve a quadratic. Consider all cases such as when $A = 0$, or when $A = B = 0$ but when $C \neq 0$.
- Consider the condition:

  ```
  if (a+b==c)
  {
  };
  ```

  What would be the PCAL expansion of such a condition?
- Consider the region $Ax + By + Cz + D \leq 0$. Given A, B, C, D and the point (x, y, z) check if the point is inside or outside the region. If outside, compute the distance of the point from the region.