CS101 Computer Programming and Utilization

Milind Sohoni

May 12, 2006
Control Flow in C++ programs
The story so far ...

We have seen elementary C++ programs and its functionalities such as:

- Variables, Assignments and Declarations.
- The basic program structure.
- Basic Input and Output.
- The If-else statement.

More Control Flows

Our objective is now to understand more intricate control-flows. Again [www.cplusplus.com/doc/tutorial](http://www.cplusplus.com/doc/tutorial) for reference.
The do-while

#include <iostream.h>
// estimate log
int main()
{
    int N,a,b;
    cout << "N?" << "\n";
    cin >> N;
    a=0; // this is the log
    b=1; // estimate to N
    do
    {
        a=a+1;
        b=b*10;
    }
    while (b<N);
    cout << a << "\n";
    return 0;
}

This program mylog.c computes largest \(a\) such that \(10^a < N\).
#include <iostream.h>
// estimate log
int main()
{
    int N,a,b;
    cout << "N?" << "\n";
    cin >> N;
    a=0; // this is the log
    b=1; // estimate to N
    do
    {
        a=a+1;
        b=b*10;
    }
    while (b<N);
    cout << a << "\n";
    return 0;
}
The do-while

#include <iostream.h>
// estimate log
int main()
{
    int N,a,b;
    cout << "N?" << "\n";
    cin >> N;
    a=0; // this is the log
    b=1; // estimate to N
    do
    {
        a=a+1;
        b=b*10;
    }
    while (b<N);
    cout << a << "\n";
    return 0;
}

This program mylog.c computes largest $a$ such that $10^a < N$.
The basic structure is:

prevline;
do
{
   codeblock
}
while (cond);
nextline;

The execution ...

- When the do statement is encountered, a note is made of the line number.
  Execution continues
The do-while

#include <iostream.h>
// estimate log
int main()
{
    int N,a,b;
    cout << "N?" << "\n";
    cin >> N;
    a=0; // this is the log
    b=1; // estimate to N
    do
    {
        a=a+1;
        b=b*10;
    } while (b<N);
    cout << a << "\n";
    return 0;
}

The basic structure is:
prevline;
do
{
    codeblock
}
while (cond);
nextline;

The execution ...

- After the execution of code block, the while and condition ($b < a$) is seen and evaluated. If this condition evaluates to true, the execution goes back to do.
- If false execution goes to the next line.
Estimating $\pi$: pi.c

```c
#include <iostream.h>
// estimate pi
int main()
{
    int N,count;
    float pi,r,delta,x,y;
    cout << "N?" << " \n";
    cin >> N;
    x=1; y=1; count=0;
    delta=1.0/N;
    
    count all grid points
    in unit circle
    
    pi=4.0*count/(N*N);
    cout << pi << " \n";
    return 0;
}
```

Milind Sohoni ()
CS101 Computer Programming and Utilization
May 12, 2006 9 / 3
#include <iostream.h>
// estimate pi
int main()
{
    int N,count;
    float pi,r, delta, x, y;
    cout << "N?" << "\n";
    cin >> N;
    x=1; y=1; count=0;
    delta=1.0/N;
    count all grid points
    in unit circle
    pi=4.0*count/(N*N);
    cout << pi << "\n";
    return 0;
}
Estimating \( \pi \): pi.c

```c
#include <iostream.h>
// estimate pi
int main()
{
    int N,count;
    float pi,r,delta,x,y;
    cout << "N?" << "\n";
    cin >> N;
    x=1; y=1; count=0;
    delta=1.0/N;
    count all grid points
    in unit circle
    pi=4.0*count/(N*N);
    cout << pi << "\n";
    return 0;
}
```

---

Milind Sohoni
CS101 Computer Programming and Utilization
May 12, 2006 11 / 3
The more common-while loop

Estimating log (again): mylog2.c

```c
#include <iostream.h>
// estimate log
int main()
{
    int N,a,b;
    cout << "N?" << "\n";
    cin >> N;
    a=0; // this is the log
    b=1; // estimate to N
    while (b<N)
    {
        a=a+1;
        b=b*10;
    }
    cout << a << "\n";
    return 0;
}
```

Usually, the `while` loop below is more prevalent. Basic structure:

```
prevline;
while (cond)
{
    code block
};
nextline;
```

The difference between the `do-while` and the `while` is that the condition is evaluated before the loop is entered. Thus, `mylog2.c` returns 0 for input 1 while `mylog.c` makes an error.
The for loop

Execution: fire.c

#include <iostream.h>
// countdown using a for loop
// from www.cplusplus.com
int main ()
{
    int n;
    for (n=10; n>0; n=n-1) {
        cout << n << " , ";
    }
    cout << "FIRE!";
    return 0;
}
The for loop

#include <iostream.h>
// countdown using a for loop
// from www.cplusplus.com
int main ()
{
    int n;
    for (n=10; n>0; n=n-1) {
        cout << n << "", ";
    }
    cout << "FIRE!";
    return 0;
}

The for loop has four parts:

for (init;condn;assignment)
{
    body
}
The for loop

Execution: fire.c

```c
#include <iostream.h>
// countdown using a for loop
// from www.cplusplus.com
int main ()
{
    int n;
    for (n=10; n>0; n=n-1) {
        cout << n << "", ";
    }
    cout << "FIRE!";
    return 0;
}
```

The for loop has four parts:

- **init** is the initialization of the looping variable. This is done only once when the statement is first encountered.
- **condn** is checked everytime, and if true, the **body** is entered.
- Everytime, but the first, the **assignment** is executed, which hopefully changes the looping variable.
- Note that the looping variable is also available inside the **body**.
#include <iostream.h>

// estimate pi
int main()
{
    float pi;
    int i,j,N,count;
    cout << "N?" << "\n";
    cin >> N;
    count=0;
    for (i=1;i<=N;i=i+1){
        for (j=1;j<=N;j=j+1){
            if (i*i+j*j<=N*N)
                count=count+1;
        }
    }
    pi=4.0*count/(N*N);
    cout << pi << "\n";
    return 0;
}

Main features:

- Two nested for loops.
- Note the two variables i,j must be declared, initialized, checked and updated.
- Structure, much much simpler.
\[ \pi \text{ again: pi2.c} \]

```c
#include <iostream.h>
// estimate pi
int main()
{
    float pi;
    int i,j,N,count;
    cout << "N?" << "\n";
    cin >> N;
    count=0;
    for (i=1;i<=N;i=i+1){
        for (j=1;j<=N;j=j+1){
            if (i*i+j*j<=N*N)
                count=count+1;
        }
    }
    pi=4.0*count/(N*N);
    cout << pi << "\n";
    return 0;
}
```

Main features:
- Two nested `for` loops.
- Note the two variables `i,j` must be declared, initialized, checked and updated.
- Structure, much much simpler.

Assignment
- Write a program which takes in \( N \) reals numbers and prints out the mean and variance.
- Use the expansion of \( e \) and compute an approximation to the value of \( e \).
Root Finding

desired root

lo

mid

hi

f(x)

Milind Sohoni
CS101 Computer Programming and Utilization
May 12, 2006
The Bisection Method

- Start with a lo and hi values such that \( f(lo) \times f(hi) < 0 \).
- Compute mid and \( f(mid) \).
- \textbf{while} \( |f(mid)| > tol \), locate the next interval to be either [low, mid] or [mid, hi].
Root Finding

The Bisection Method

- Start with a lo and hi values such that \( f(lo) \times f(hi) < 0 \).
- Compute mid and \( f(mid) \).
- while \( |f(mid)| > tol \), locate the next interval to be either [low,mid] or [mid,hi].

We write a program to solve cubics \( Ax^3 + Bx^2 + Cx + D \).

- Takes in inputs A,B,C,D.
- Takes in inputs lo,hi, tol. Returns if \( f(lo) \times f(hi) > 0 \).
- Sets up the while loop and returns mid so that \( |f(mid)| < tol \).
#include <iostream.h>
int main()
{
    float A,B,C,D,lo,hi,mid,flo,fhi,fmid;
    float tol;
    cout << "A B C D ?" << "\n";
    cin >> A >> B >> C >> D;
    cout << "low high tolerance" << "\n";
    cin >> lo >> hi >> tol;
    flo=A*lo*lo*lo +B*lo*lo +C*lo +D;
    fhi=A*hi*hi*hi +B*hi*hi +C*hi +D;
    if (flo*fhi>0)
    {
        cout << "error in hi-lo" << "\n";
        return 1;
    };

    THE MAIN PART
    cout << mid << "\n";
    return 0;
}
mid=(lo+hi)/2;
fmid=A*mid*mid*mid +B*mid*mid +C*mid +D;
while (fabs(fmid)>tol)
{
  if (flo*fmid >0)
  {
    lo=mid;
    flo=fmid;
  }
  else
  {
    hi=mid;
    fhi=fmid;
  }
  mid=(lo+hi)/2;
  fmid=A*mid*mid*mid +B*mid*mid +C*mid +D;
};  // end of while
The Secant Method

**assignment**

- The secant method computes mid in a different way. It takes mid as the point where the line joining the low and the high values, as shown in the figure above. Write a C++ program to implement the secant method of computing the roots of a degree 5 polynomial. Input the coefficients of the polynomial, hi and lo values, a tolerance and a bound N on the number of iterations.