Control Structures: From linear programs to decisions and repetitions

1. Programs we’ve seen are simple linear command sequences

2. Problem solving:
   (a) Decide what to do based on certain conditions
   (b) Repeat a sequence of actions, based on condition

3. C++ offers control structures:
   (a) Decide what to do: \textbf{if} \cdots \textbf{then} \cdots \textbf{else}
   (b) Iterate group of commands: \textbf{for}, \textbf{while}, \textbf{do while}, \textbf{for}, \textbf{while}
Conditional Execution

1. Condition specified
2. Actions taken in case condition is evaluated true
3. Actions taken in case condition is evaluated false
4. Sequences of decisions can be defined
Conditional execution: if ...else

if ... else: general structure:

```plaintext
if ( condition ){
    // executed if condition is true
    statement_block1;
}
else {
    // executed if condition is false
    statement_block2;
}
```
Boolean logic again: Conditional Expressions!

<table>
<thead>
<tr>
<th>Relational Operators</th>
<th>Logical Operators</th>
</tr>
</thead>
<tbody>
<tr>
<td>Operator</td>
<td>Interpretation</td>
</tr>
<tr>
<td>&lt;</td>
<td>is less than</td>
</tr>
<tr>
<td>&lt;=</td>
<td>is less than or equal to</td>
</tr>
<tr>
<td>&gt;</td>
<td>is greater than</td>
</tr>
<tr>
<td>&gt;=</td>
<td>is greater than or equal to</td>
</tr>
<tr>
<td>&lt;&gt;</td>
<td>is not equal to</td>
</tr>
</tbody>
</table>

Examples:
- if ( x<4 && !(y>6))
- if ( x*y > 5 || x%4 == 0)
- if ( !((x*y)%4 == 0))
### Precedence and associativity

<table>
<thead>
<tr>
<th>precedence</th>
<th>operator</th>
<th>Associativity</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>()</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>&lt;, &lt;=, &gt;, &gt;=</td>
<td>left to right</td>
</tr>
<tr>
<td>2</td>
<td>==, !=</td>
<td>left to right</td>
</tr>
<tr>
<td>3</td>
<td>&amp;&amp;</td>
<td>left to right</td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**General note:**

Always use brackets to make conditional statements more readable!
Example: if ... else

```cpp
// if_else.cpp: demonstration of if, else structure
#include <iostream.h>

int main (void){
  int a,b;

  cout << "Enter a and b: ";
  cin >> a >> b;
  cout << "You entered: " << a << ", " << b << endl;

  if ( a%b == 0 && b==4){
    cout << "Yes!" << endl;
  }
  else {
    cout << "No!" << endl;
  }

  return 0;
}
```
Example: if ... else

```cpp
// if_else2.cpp: demonstrates actually useful application of if else
#include <iostream.h>

int main (void){
    double a, b, c;

    cout << "Enter a and b (separate by whitespace): ";
    cin >> a >> b;

    if ( b != 0 ){
        c = a / b;
        cout << "a / b = " << c << endl;
    }
    else {
        cout << "Division by zero!" << endl;
    }

    return 0;
}
```
Example: if ... else: nested

```
// program demonstrates nested if ... else condition
#include <iostream.h>

int main (void){
    char answer;

    cout << ”Check your wallet. Do you have cash? (enter Y or N):”;
    cin >> answer;

    if ( answer == ’Y’ ){
        cout << ”Pay using cash
”;
    }
    else {
        cout << ”Oops! Oh well, do you have cheques? (enter Y or N):”;
        cin >> answer;
        if ( answer == ’Y’ ){
            cout << ”So write a check!
”; }
    } }else { 
```
cout << "Oops... Do you have a credit card? (enter Y or N):";
cin >> answer;
if (answer == 'Y') {
    cout << "Great! Use the credit card!\n";
}
else {
    cout << "Your turn to do the dishes, my friend!\n";
}
return 0;
Switch - case statements

1. Shorter version of if · · · else
2. switch statement identifies variable whose value will determine decision
3. case statement identifies in which case to do what:
   (a) case value:
   (b) followed by command block
   (c) use break to avoid subsequent case statements
switch...case example

1    // case_switch.cpp: test switch case statements
2    # include <iostream.h>
3
4    int main ( void) {
5        int color;
6
7        cout << "What’s your favorite color (red=1, blue=2)?\n";
8        cin >> color;
9
10       switch ( color) {
11           case 1: cout << "Bweurgh! I don’t like red!\n"; break;
12           case 2: cout << "I like blue! Excellent!\n"; break;
13           default: cout << "Not a color I know\n";
14       }
15
16       return 0;
17    }
switch...case nested example

```cpp
// switch_case_nested.cpp: test switch case statements
#include <iostream.h>

int main (void){
    int color, pet;

    cout << "Favorite color (red=1, blue=2) and pet (dog=1, cat=2)? \n";
    cin >> color >> pet;

    switch (color){
        case 1:
            switch (pet){
                case 1: cout << "red dogs?! \n"; break;
                case 2: cout << "red cats?! \n"; break;
            }
            break;
        case 2:
            switch (pet){
                case 1: cout << "blue dogs?! \n"; break;
            }
    }
}
```

January 18, 2004

case 2: cout << "blue cats?!
"; break;
}
break;
{
return 0;
}
Now, this is where it gets interesting: LOOPS!

1. Loops: conditional iteration of command block

2. Two types:
   (a) while, do · · · while
   (b) for

3. while:
   (a) conditional iteration
   (b) pre- and post-evaluation

4. for:
   (a) condition involves counter
   (b) only in pre-evaluation

5. Equivalence:
   (a) for can be implemented using while
   (b) Involves counter
while decription

1  pre:
2  while (condition) {
3    commands;
4  }
5
6  post:
7  do {
8    commands;
9  } while (condition);
while

1  // while_pre: description of while loop
2  #include <iostream.h>
3
4  int main ( void ){
5
6    char answer;
7
8    cout << "Y or N?: ";
9    cin >> answer;
10
11   while ( answer == 'Y' ) {
12       cout << "Great! Once more? (Y or N): ";
13       cin >> answer;
14   }
15
16   cout << "OK, we're done! " << endl;
17
18   return 0;
19 }
do ... while

1 // while_post: description of post evaluated while loop
2 #include<iostream.h>
3
4 int main ( void){
5    char a;
6
7    do {
8        cout << "Want more? (Y or N)" << endl;
9        cin >> a;
10    } while ( a == 'Y' );
11
12    cout << "OK OK!..." << endl;
13
14    return 0;
15  }

while counting

// while_count1.cpp: demonstration of while loop counting
#include <iostream.h>

int main (void)
{
    int i = 0;

    cout << "i = " << i << endl;

    while ( i < 100000000 ){
        i = i + 2;
        // cout << "i = " << i << endl;
    }

    cout << "Done."

    return 0;
}
do ... while counting

```c++
// while_count1.cpp: demonstration of while loop counting
#include <iostream.h>

int main(void)
{
    int i = 0;
    cout << "i = " << i << endl;
    do {
        i++;
        cout << "i = " << i << endl;
    } while (i < 10);
    return 0;
}
```
Let’s write a program!

Problem statement:
We need to write a C++ program that calculates the average ($\bar{x}$) of any given $n$ numbers.

$$\bar{x} = \frac{\sum_{i=1}^{n} x_i}{n}$$

**INPUT:** $n$ numbers are entered by the user, each on a new line. -1 indicates the user has entered the last number.

**OUTPUT:** After the user has entered the last number (i.e. -1), the program prints the average of all numbers (except -1) on a separate line.

**Requirement:** Use a while loop.
// mean.cpp: calculates average of n numbers
#include <iostream.h>

int main ( void ){
    int n=0;
    double s, v;
    s = 0;
    v = 0;
    do {
        s += v;
        cout << "s=" << s << endl;
        cin >> v;
        if ( v != -1 ){
            n++;
        }
    }
while ( v != -1 );

cout << "sum = " << s << endl;
cout << "n = " << n << endl;

return 0;

}
Let’s write a program!

Problem statement:

We want to write a guessing game. The computer determines a random number in the range 1 to 10. The user can guess, and the computer will tell whether the guess is either too high or too low. When you guess the number, a congratulatory message appears.

**INPUT:** User enters guesses.

**OUTPUT:** The program prints messages. One message says the guess is too high, the other message says the number is too low. After each guess, the program asks the user whether they want to continue, Y or N. In case the user answer ’Y’, he or she can enter another guess. If not, the program terminates by revealing the random number.

**Requirement:** Use a while loop, and the `stdlib.h`, `math.h` and `time.h` libraries (random numbers)
Random Numbers

// guess_game/rand.cpp: random numbers
#include <iostream.h>
#include <stdlib.h>
#include <time.h>
#include <math.h>

int main (void) {
    int r;

    // random number generator must be initialized
    srand ( time(NULL) );
    r = (int) ceil ( ((double) rand () / RAND_MAX) * 10 );
    cout << "r=" << r << endl;

    return 0;
}

while description

1 pre:
2 while (condition) {
3 commands;
4 }
5
6 post:
7 do {
8 commands;
9 } while (condition);
```cpp
#include <iostream.h>

int main ( void )
{
    int i = 0;

    cout << "i = " << i << endl;

    while ( i < 100000000 )
    {
        i = i + 2;
        // cout << "i = " << i << endl;
    }

    cout << "Done."
        << endl;

    return 0;
}
```
do ... while counting

1 // while_count1.cpp: demonstration of while loop counting
2 #include <iostream.h>
3
4 int main ( void ){
5     int i=0;
6
7     cout "i=" << i " endl; 
8     do {
9         i ++;
10        cout "i=" << i " endl;  
11    } while ( i < 10 );
12
13    return 0;
14 }


for loop

for (initial value; condition; increment) {
    commands;
}

1. Three components:
   (a) Set counter variable to initial
       value (can be declaration)
   (b) Termination condition: evaluated
       at each iteration
   (c) Increment statement: increases

2. Absent or inadequate condition: endless loop

3. Increment: does not have to be 1
Example for loop

```cpp
// for_counter.cpp: counts from 0 to 9 and prints squares
#include <iostream.h>

int main (void) {
    int x, y;
    cout << "x" << endl;

    for (x = 0; x < 5; x++) {
        for (y = 0; y < 5; y++) {
            cout << "(" << x << "," << y << ")" << endl;
        }
    }
    return 0;
}
```
Equivalence of for and while loop

```
// for_while_equivalent.cpp: demonstrates how for loop can be replaced
#include <iostream.h>

int main ( void ){
  int i;
  // note declaration in for loop!
  for ( i=0; i<10; i++){
    cout << "for i: " << i << endl;
  }

  // while loop:
  i =0;
  while ( i<10){
    cout << "while i: " << i << endl;
    i++;
  }
  return 0;
}
```
Some notes on for loops

1. For counter variable problems:
   (a) Modified anywhere loop: affects for loop termination!
   (b) If set to zero or any other value: may cause endless loop!

2. Mostly appropriate when number of iterations is know or determined in advance
   (a) Always think about three issues:

3. Always make sure counter variable is declared

4. Be careful with termination condition:
   \[ t=0, \ t<10 = (0, \ldots, 9) \]