Until now, statements were all simple statements. Here we will introduce structured statements. First of all, a *compound statement* is a sequence of 2 or more consecutive statements (simple, compound, and/or structured) enclosed within { and } .

I. Selection Constructs

- if
- if/else
- nested if
- switch/case

if statement

In pseudocode (structured English) we might express a decision as:

If student's grade is greater than or equal to 60
Print "Passed"

In C++ this would be coded as:

```
if (grade >= 60)
     cout << "Passed";
```

Syntax for **if** statement:
if <*boolean expression* > <*statement1*> ;

A *boolean expression* represents a condition that is either true or false. It is formed using operands (constants, variables) and operators (arithmetic operators, relational operators, logical operators).

Arithmetic	**Relational**	**Logical**
+ -	< less than > greater than	|| or
* / %	== equal to != not equal to	&& and
arithmetic functions	<= less than or equal to >= greater than or equal to	! not

Ex. `if ((x > 0) && (a == b)) …`

Operator Precedence

Operator	**Associativity**	**Type**
( )	L → R	parenthesis
++ – – + – !	R → L	unary operators
* ? %	L → R	multiplicative
+ –	L → R	additive
<< >>	L → R	insertion
< <= > >=	L → R	relational
= = !=	L → R	equality
&&	L → R	and
||	L → R	or
?:	R → L	conditional
= += –= *= /= %=	R → L	assignment

if/else statement

In pseudocode:
If student's grade is greater than or equal to 60
Print "Passed"
else
Print "Failed"

In C++

```
if (grade >= 60)
     cout << "Passed";
else
     cout << "Failed";
```

syntax for if/else statement:
if <*boolean expression* > <*statement1*> [**else** <*statement2*>];

The C++ conditional operator - ?:

Same as the if/else structure but more compact. This is the only ternary operator in C++. Example:

```
grade>=60? cout << "Passed" : cout << "Failed";
```

This is the same as:

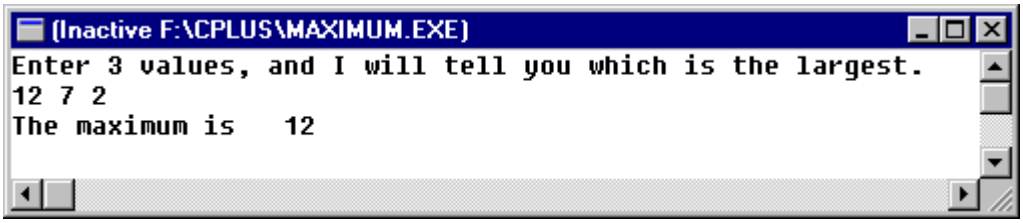
```
cout << (grade >=60? "Passed" : "Failed");
```

In this case, the conditional expression evaluates to "Passed" if the condition is true and to "Failed " if the condition is false.

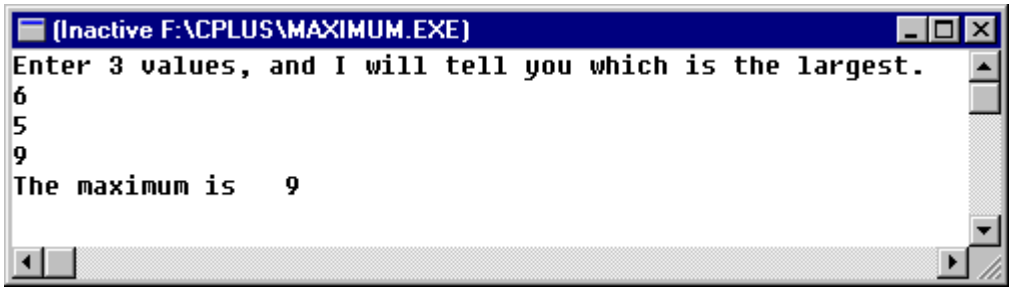
==FLOWCHARTING==

```
//maximum.cpp
// Finding the largest of 3 values

#include <iostream>

int main()
{
   int A, B, C, max;

   cout << "Enter 3 values, and I will tell you which is largest.\n";
   cin >> A >> B >> C; //really should have user prompts here
   if (A > B)
      max = A;
   else
      max = B;
   if (C > max)
      max = C;
   cout << "The maximum is   " << max << endl;
   return 0;      //successful termination
} //end main
```

RUN #1:

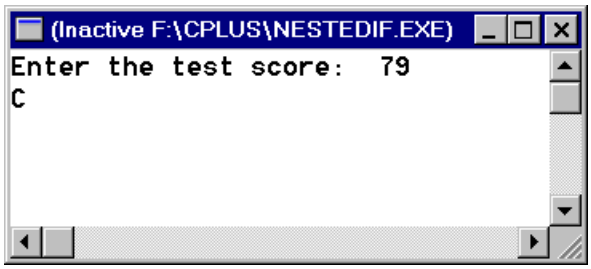


RUN#2:



Nested-if

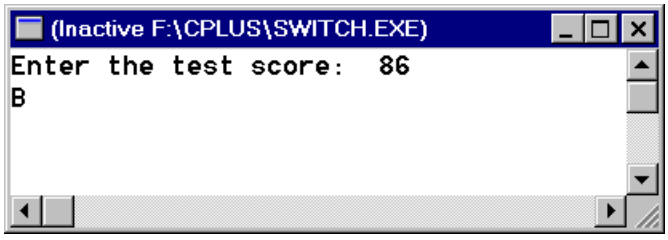
```
//nestedIf.cpp
// modified from Hubbard Ex. 2.14
// This program converts a test score into
// its equivalent letter grade.

#include <iostream>
int main()
{
   int score;
   cout << "Enter the test score:  ";
   cin >> score;
   if (score > 100) cout << "Error: score is out of range." ;
   else if (score >= 90) cout << 'A';
   else if (score >= 80) cout << 'B';
   else if (score >= 70) cout << 'C';
   else if (score >= 60) cout << 'D';
   else if (score >=  0) cout << 'F';
   else cout << "Error: score is out of range.";

   return 0;   //successful termination
} //end main
```



Why does this work? EXPLAIN.

Switch

```
//switch.cpp
// modified from Hubbard Ex. 2.15
// This program converts a test score into
// its equivalent letter grade, using a switch statement.

#include <iostream>
int main()
{
   int score;
   cout << "Enter the test score:  ";
   cin >> score;
   switch (score/10) {
      case 10:
       case  9: cout << 'A' << endl; break;
       case  8: cout << 'B' << endl; break;
       case  7: cout << 'C' << endl; break;
       case  6: cout << 'D' << endl; break;
       case  5:
       case  4:
       case  3:
       case  2:
       case  1:
       case  0: cout << 'F' << endl; break;
       default: cout << "Error: score is out of range.\n";
      }
   return 0;   //successful termination
} //end main
```



How does this work? The computer "drops down" through the "empty" conditions. EXPLAIN.

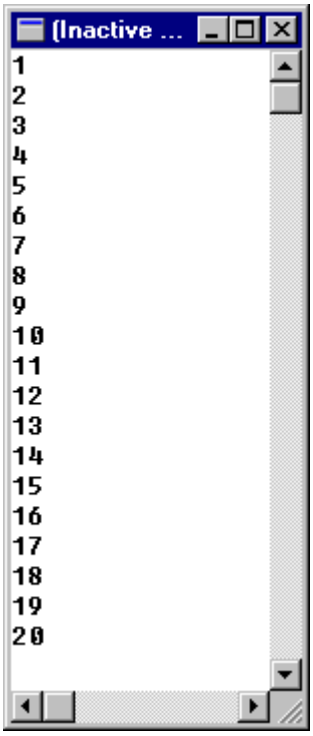
In the following example, the function `rand()` is defined in the header file `<stdlib>`. `rand()` generates pseudo-random unsigned integers in the range of 0 to `RAND_MAX`. `RAND_MAX` is a constant which is also defined in `<stdlib>`.

II. Iteration (Repetition) Constructs

In order to control iteration, we use one of three structured control statements.

- for
- while
- do/while

A Counting Program: **We would like to write a program that will print the integers between 1 and 20, say, like the following output:**



Here's one way:

```
//counting2.cpp
// A Counting Program
// Using a FOR/DO Loop

#include <iostream>
using namespace std;
int main()
{
   int count ;

   for (count=1; count<=20; count++)
     cout << count << endl;
   return 0;   //successful termination
} //end main
```

Here's another way:

```
//counting.cpp
// A Counting Program
// Using a WHILE/DO Loop

#include <iostream>
using namespace std;
int main()
{
   int count;

   count = 1;
   while (count <=20){
      cout << count << endl;
      count = count + 1;
      }
   return 0;   //successful termination
} //end main
```

And yet another way:

```
//counting3.cpp
// A Counting Program
// Using a DO/WHILE Loop

#include <iostream>
using namespace std;
int main()
{
   int count;

   count = 1;
   do {
     cout << count << endl;
     count = count + 1;
   }  while (count <=20);

   return 0;   //successful termination
} //end main
```

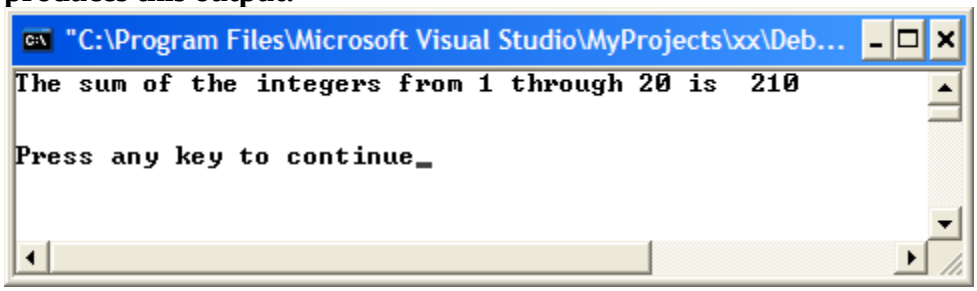
A Summing Program

We wish to write a program that will calculate the sum of the integers between 1 and 20.

The program:

```
//sum1.cpp
// A Summing Program
// Using a WHILE/DO Loop

#include <iostream>
using namespace std;
int main()
{
   int sum, count ;
   sum = 0;
   count = 1;
   while (count <=20){
     sum = sum + count;
     count = count + 1;
   }
   cout << "The sum of the integers from 1 through 20 is  ";
   cout << sum << endl;
   return 0;   //successful termination
} //end main
```

produces this output:



and so does this program:

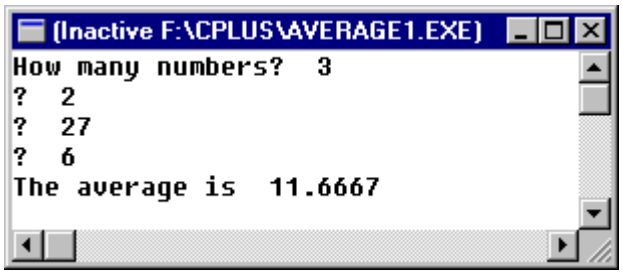
```
//sum2.cpp
// A Summing Program
// Using a FOR Loop

#include <iostream>
using namespace std;
int main()
{
   int sum, count ;
   sum = 0;

   for (count=1; count<=20; count++)
     sum = sum + count;
   cout << "The sum of the integers from 1 through 20 is  ";
   cout << sum << endl;
   return 0;   //successful termination
```

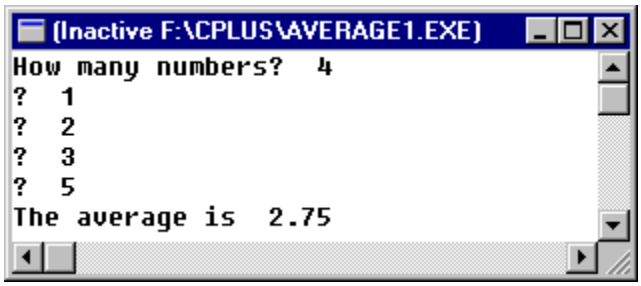
```
} //end main
```

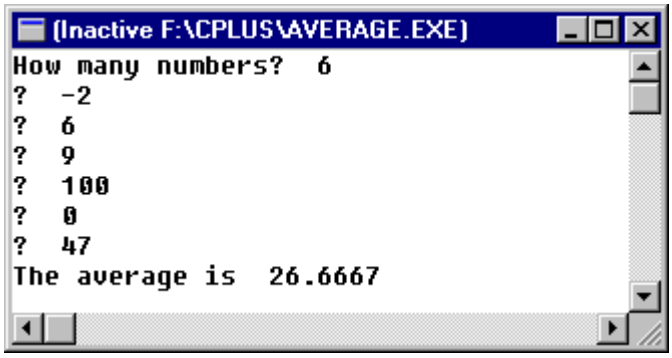
Just an Average Program ;-)

```
//average1.cpp
// This program calculates the average of any number of numbers.
// Using the while/do structure

#include <iostream>

int main()
{
   int n, count;
   float x, sum, avg;

   count = 1;
   sum = 0;
   cout << "How many numbers?  ";
   cin >> n;
   while (count <=n) {
      cout << "?  ";
      cin >> x;
      sum = sum + x;
      count++;
   }  //end while
   avg = sum / n;
   cout << "The average is  " << avg << endl;
   return 0;   //successful termination
} //end main
```

```
//average2.cpp
// This program calculates the average of any number of numbers.
// Using the for structure

#include <iostream>

int main()
{
   int n, count;
   float x, sum, avg;

   sum = 0;
   cout << "How many numbers?  ";
   cin >> n;
   for (count=1; count<=n; count++){
      cout << "?  ";
      cin >> x;
      sum = sum + x;
   }  //end for
   avg = sum / n;
   cout << "The average is  " << avg << endl;
   return 0;   //successful termination
} //end main
```







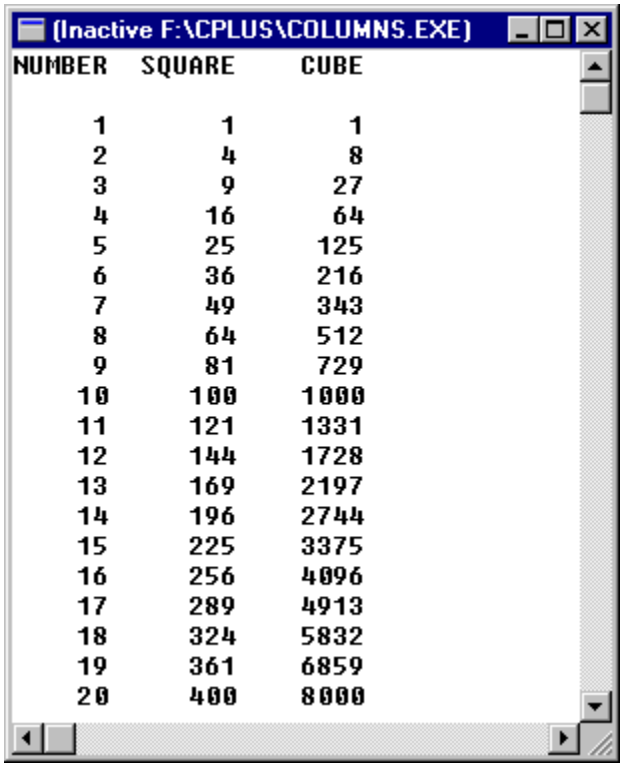
When to use the while construct? When to use for?

Improving our Output

Setw is a stream, insertion manipulator that sets the field width of the output. It requires the iomanip.h header file, i.e.,: #include <iomanip>

The following program illustrates how it works.

```
//columns.cpp
// Some Calculations
// Prettier output using stream manipulators

#include <iostream>
#include <iomanip>

int main()
{
   int number, square, cube;

   cout << setw(6) << "NUMBER" << setw(8) << "SQUARE";
   cout << setw(8) << "CUBE" << endl << endl;
   for (number=1; number <=20; number++){
      square = number * number;
      cube = square * number;
      cout << setw(6) << number << setw(8) << square;
      cout << setw(8) << cube << endl;
   } //end for
   return 0; //successful termination of program
} //end main
```



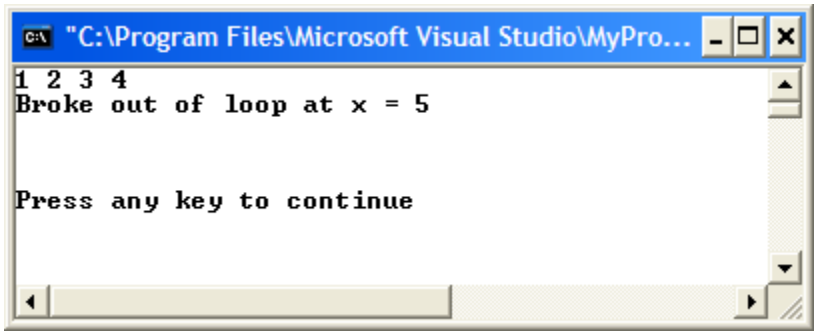
Branching -
Break and Continue Statements

Conditional vs. unconditional branching. goto
These statements should be used only when absolutely necessary.

Break - to break out of a loop. Transfers control to the first statement after the end of the loop.

Example. What is output?

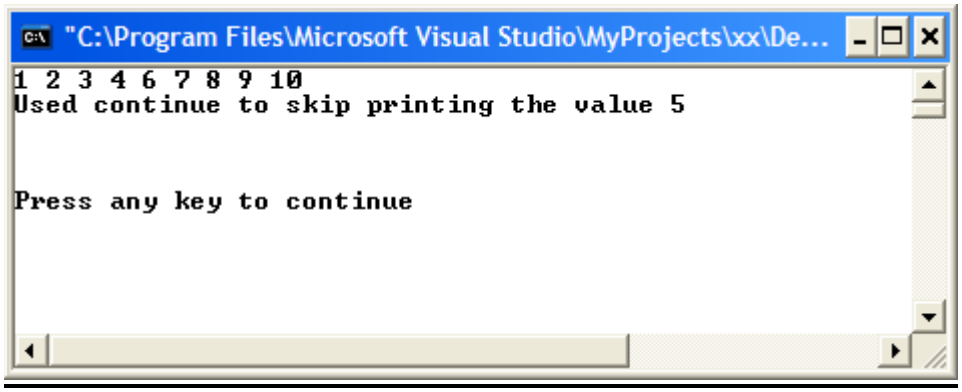
```
// Deitel2-26.cpp
// This program is modified from the Deitel book, Figure 2.26, p.104.
// Using the break statement in a for structure
#include <iostream>
using namespace std;
int main()
{
   int x;
   for (x=1; x<=10; x++) {
      if (x==5)
         break;    //break loop only if x is 5
      cout << x << " ";
   }  //end for
   cout << "\nBroke out of loop at x = " << x << endl;
   cout<< endl << endl << endl;
   return 0;  //successful termination of program
}  //end main
```



Continue - skips the remainder of the body of the loop, and continues the loop. Transfers control to "loop again."

Example. What is output?

```
// Deitel2-26.cpp
// This program is modified from the Deitel book, Figure 2.27, p.105.
// Using the Continue statement in a for structure

#include <iostream>
using namespace std;
int main()
{
   int x;
   for (x=1; x<=10; x++) {
      if (x==5)
         continue;      // skip remaining code in loop
                              // only if x is 5
      cout << x << " ";
   }  //end for
   cout << "\nUsed continue to skip printing the value 5 " << endl;
   cout << endl << endl << endl;
   return 0;  //successful termination of program
} //end main
```



Random Numbers

Using switch, break, for, mod

```
//TestRandNums.cpp
// This program gets 1000 random numbers and tests them
#include <iostream>
#include <stdlib>

int main(){
   int num0, num1, num2, num3, num4, num5, num6, num7, num8, num9;
   num0=num1=num2=num3=num4=num5=num6=num7=num8=num9=0;
   for (int i=0; i<1000; i++) {
      int r = rand()%10;      //remainder gives a one digit RN
                              //between 0 and 9
   switch (r) {
      case 0: num0++; break;     //Note: This program will be more
      case 1: num1++; break;    //efficient when we learn arrays.
      case 2: num2++; break;
      case 3: num3++; break;
      case 4: num4++; break;
      case 5: num5++; break;
      case 6: num6++; break;
      case 7: num7++; break;
      case 8: num8++; break;
      case 9: num9++; break;
      default: cout << "Error: unexpected random number" << r <<endl;
      } //end switch
   } //end for
   cout << "number of 0 : " << num0 << endl;
   cout << "number of 1 : " << num1 << endl;
   cout << "number of 2 : " << num2 << endl;
   cout << "number of 3 : " << num3 << endl;
   cout << "number of 4 : " << num4 << endl;
   cout << "number of 5 : " << num5 << endl;
   cout << "number of 6 : " << num6 << endl;
   cout << "number of 7 : " << num7 << endl;
   cout << "number of 8 : " << num8 << endl;
   cout << "number of 9 : " << num9 << endl;
   return 0;   //successful termination
} //end main
```