SSCM 1313

C++ COMPUTER PROGRAMMING

Chapter 3:

Loop and Branching

Authors:

Farhana Johar

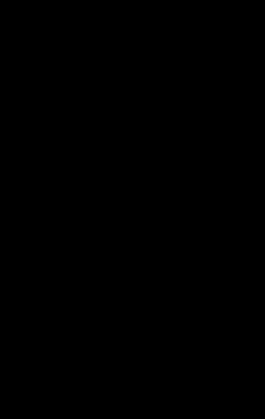
Professor Dr. Shaharuddin Salleh

Loop

*Something that repeats.*

|  |  |
| --- | --- |
| Start | 1.0 |
| End | 2.0 |
| Increment | 0.25 |

|  |  |  |  |
| --- | --- | --- | --- |
| *Iteration* 0 | 1.0 | Start |  |
|  |  |
| *Iteration* 1 |  |  |  |
| 1.25 |  |  |
|  |  |  |
| *Iteration* 2 |  |  |  |
| 1.5 |  |  |
|  |  |  |
| *Iteration* 3 |  |  |  |
| 1.75 |  |  |
|  |  |  |
| *Iteration* 4 |  | End |  |
| 2.0 |  |
|  |  |
|  |  |  |  |



**Figure 3.2.** Iterations from 1.0 to 2.0 with an increment of 0.25.

Three types of loops in C++:

for …

while …

do … while

To end a loop:

|  |  |  |
| --- | --- | --- |
| break | *Terminates the loop and places the* |  |
| *control at the statement immediately* |  |
|  |  |
|  | *after the loop* |  |
|  |  |  |
| return | *Terminates the loop and ends the* |  |
| *program immediately* |  |
|  |  |
|  |  |  |

|  |  |  |
| --- | --- | --- |
| for loop |  |  |
| for (*Start; End; Increment*) |  |  |
| { |  |  |
| <*Set of Statements* > | *Body* |  |
| } |  |
|  |  |

**Code 3A.cpp: for loop.**

#include <iostream>

#include <conio.h>

#define n 3

using namespace std;

void main()

{

// this is not a loop int i=1;

cout << "statement " << i++ << endl;

cout << "statement " << i++ << endl;

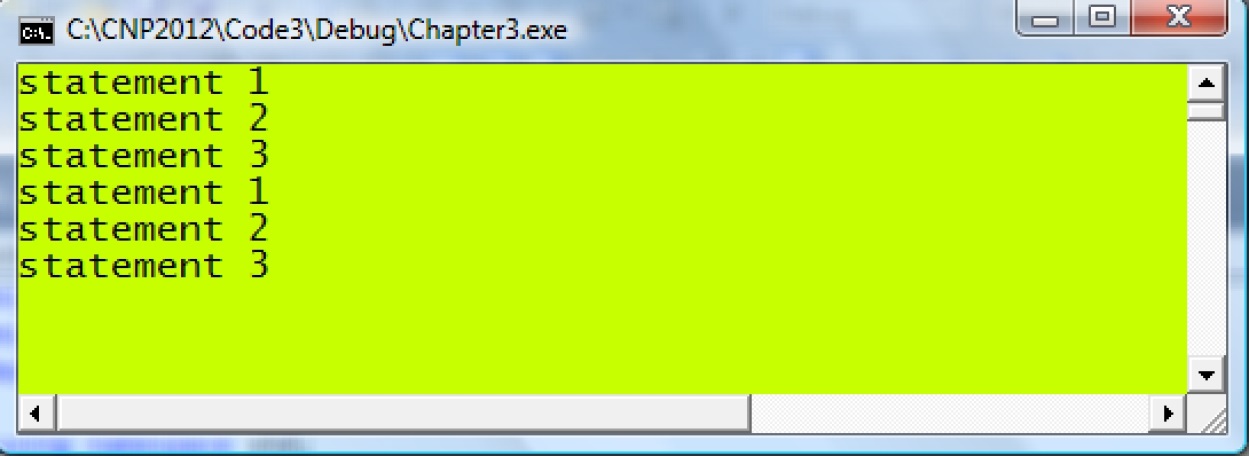
cout << "statement " << i << endl;

// for loop

For (i=1; i<=n; i++)

cout << "statement " << i << endl; getch();

}



Increment

i++ is i=i+1, or *the new value is the old one adds 1*,

i‐‐ is i=i‐1, or *the new value is the old one subtracts 1*,

i+=3 is i=i+3, or *the new value is the old one adds 3*,

i‐=3 is i=i‐3, or *the new value is the old one subtracts 3*,

i+=0.5 is i=i+0.5, or *the new value is the old one adds 0.5*.

i\*=0.5 is i=i\*0.5, or *the new value is the product of old one with 0.5.*

i/=0.5 is i=i/0.5, or *the new value is the old one divided by 0.5.*

for (i=1;i<=10;i+=2)

has five repeats, at i=1,3,5,7 and 9.

for (i=10;i>=1;i‐=2)

has five repeats, at i=10,8,6,4 and 2.

For (i=1;i<=5;++i)

has four repeats, at i=2,3,4 and 5.

for (i=0;i<=1;i+=0.5)

has three repeats, at i=0, 0.5 and 1.

**Code3B.cpp: for Solution to Figure 3.2**

#include <iostream>

#include <iostream>

#include <conio.h> #define f(x) (2\*sin(x))

using namespace std;

void main()

{

double x,y;

for (x=1.0; x<=2.0; x+=.25)

{

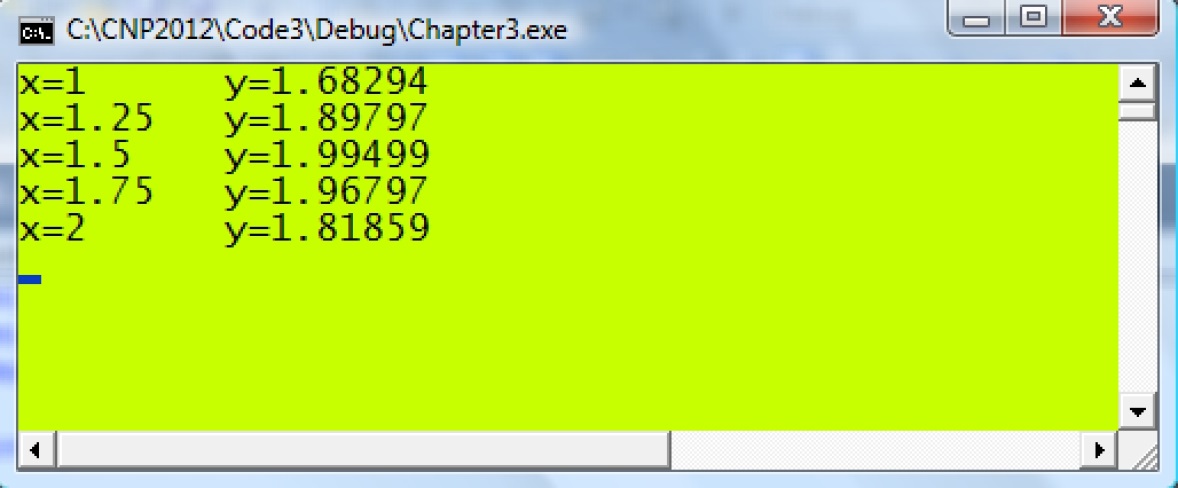
y=f(x);

cout << "x=" << x << "\t y=" << y << endl;

}

getch();

}



|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | *i* |  | *x* |  |  | *y* |  |  | *z* |  |
|  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |
|  | *Iteration* 0 | 0 |  | 1.0 |  |  | 8.0 |  |  | -5.0 |  |
|  |  |  |  |  |  |  |  |  |  |  |
|  | *Iteration* 1 |  |  |  |  |  |  |  |  |  |  |
|  | 1 |  | 1.25 |  |  | 6.0 |  |  | -2.25 |  |
|  |  |  |  |  |  |  |  |  |  |  |
|  | *Iteration* 2 |  |  |  |  |  |  |  |  |  |  |
|  | 2 |  | 1.5 |  |  | 4.0 |  |  | 0.5 |  |
|  |  |  |  |  |  |  |  |  |  |  |
|  | *Iteration* 3 |  |  |  |  |  |  |  |  |  |  |
|  | 3 |  | 1.75 |  |  | 2.0 |  |  | 3.25 |  |
|  |  |  |  |  |  |  |  |  |  |  |
|  | *Iteration* 4 |  |  |  |  |  |  |  |  |  |  |
|  | 4 |  | 2.0 |  |  | 0.0 |  |  | 6.0 |  |
|  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |
|  | **Figure 3.3**. Five iterations in | | | | *z*  *f* (*x* , *y* )3 *x* − *y* . | | | | |  |  |

**Code3C.cpp: Looping on a function.**

#include <iostream>

#include <conio.h>

#define f(a,b) ((double)(3\*a‐b))

using namespace std;

void main()

{

int i;

double x=1, y=8, z; for (i=0; i<=4; i++)

{

z=f(x,y);

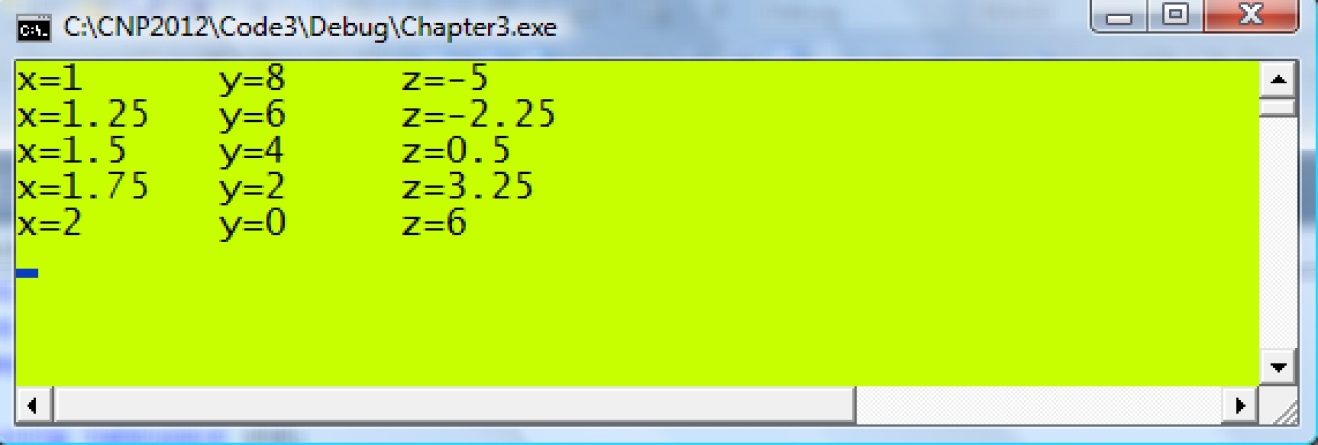
cout << "x=" << x << "\t y=" << y << "\t z=" << z << endl; x += 0.25;

y ‐= 2;

}

getch();

}



Looping with while

while (*End*)

{

<*Set of Statements* >

}

**Code3D.cpp: while loop.**

#include <iostream>

#include <conio.h> #define n 3

using namespace std;

void main()

{

* this is not a loop int i=1;

cout << "statement " << i++ << endl; cout << "statement " << i++ << endl; cout << "statement " << i << endl;

* while loop

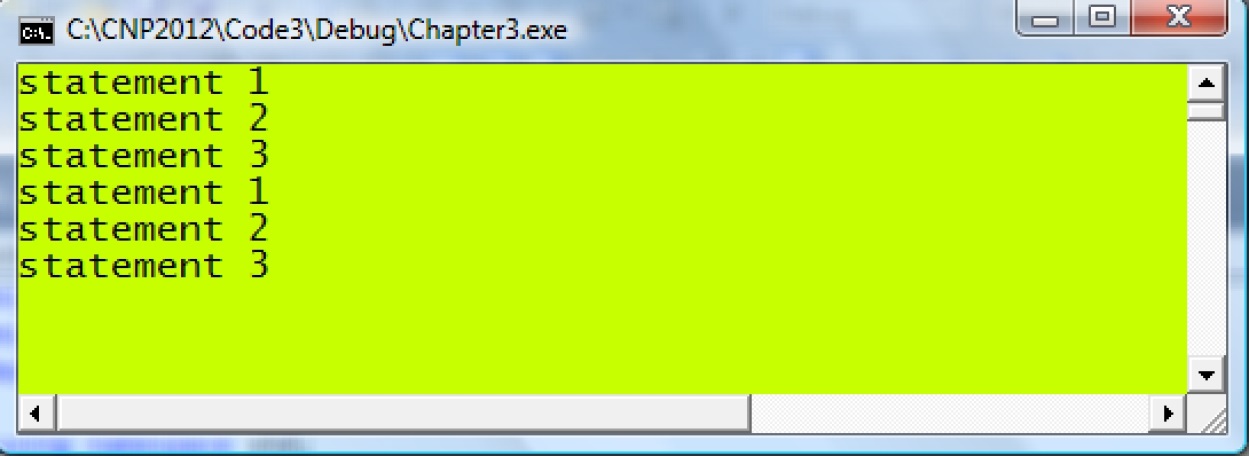
i=1;

while (i<=n)

cout << "statement " << i++ << endl;

getch();

}



*Body*

Infinite loop

while(1)

An infinite loop is practical in applications such as in controlling the user’s input in a menu. An infinite loop will only stop if it encounters break or return which is placed through a conditional branching. An infinite loop is like driving in a roundabout where the driver can go on driving forever. An infinite loop will need to stop after some criteria has been fulfilled through break or return.

Code3E.cpp illustrates an infinite loop for asking the user to key in characters in the keyboard. The program will not stop, and will continue asking the user to type the characters until the user presses ‘x’, which terminates the program through break. In this program, a *conditional branching* is used to break the loop through

if (w=='x’) break;

**Code3E.cpp: Infinite loop.**

#include <iostream>

#include <conio.h>

using namespace std;

void main()

{

int count=0; char w;

cout << "Enter a character or X to exit: " << endl;

while (1)

{

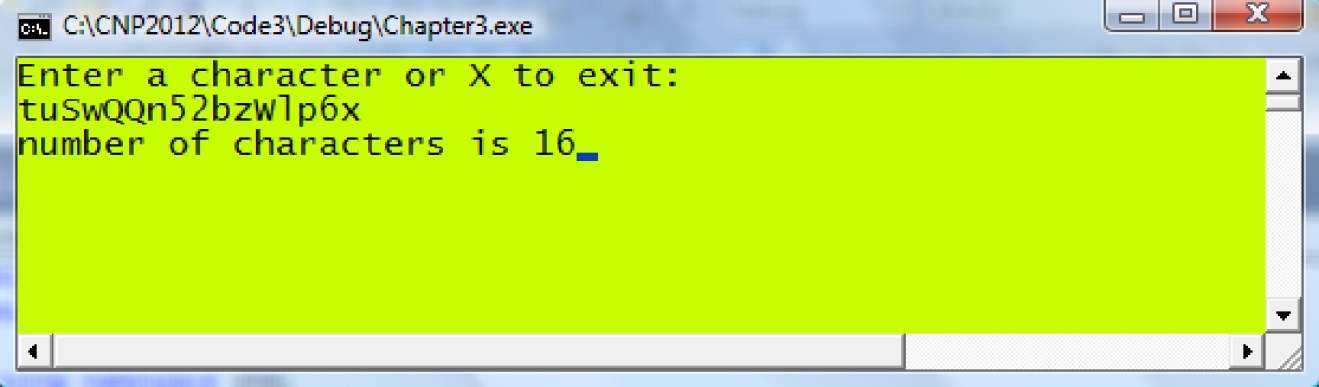
w=getche(); count++;

if (w=='x') break;

}

cout << endl << "number of characters is " << count; getch();

}



|  |  |  |
| --- | --- | --- |
| **getch()** | Read input from the keyboard. |  |
|  |  |
|  |  |  |
| return type | int |  |
| arguments | *void* |  |
| Prototype | conio.h |  |
| example | char g; |  |
| g=getch(); |  |
|  | //assigns g with the character from the keyboard. |  |
|  |  |  |
|  |  |  |
|  |  |  |
| **getche()** | Read input from the keyboard. |  |
|  |  |
|  |  |  |
| return type | int |  |
| arguments | *void* |  |
| prototype | conio.h |  |
| example | char g; |  |
| g=getch(); |  |
|  | //assigns g with the character from the keyboard and |  |
|  | displays it on the console. |  |
|  |  |  |

Looping with do…while

do

{

<*Set of Statements* > } while (*End*);

**Code 3F.cpp: do…while loop.**

#include <iostream> #include <conio.h>

using namespace std;

void main()

{

int i=1;

cout << "statement " << i++ << endl;

cout << "statement " << i++ << endl;

cout << "statement " << i << endl;

i=1;

do

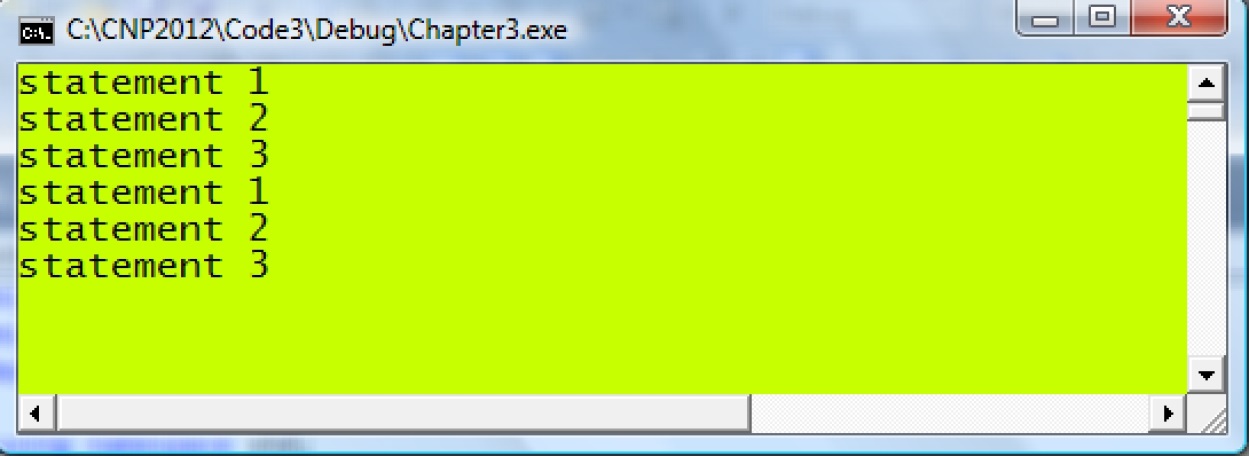
{

cout << "statement " << i++ << endl;

} while (i<=3);

getch();

}



*Body*

**Code3G.cpp: Computing the mean of numbers.**

#include<iostream>

#include <conio.h>

using namespace std;

void main()

{

int i,n;

double x, sum=0.0, mean;

cout << "Enter number of input: "; cin >> n;

for (i=1;i<=n;i++)

{

cout << "Input " << i << ", enter a number: ";

cin >> x;

sum += x;

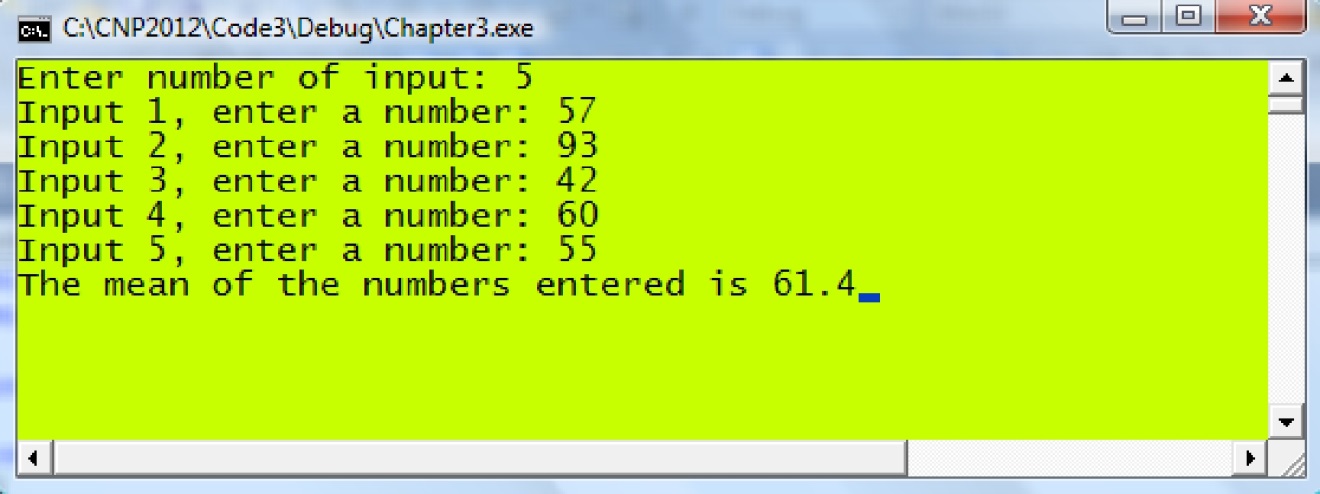
}

mean=(double)sum/n;

cout << "The mean of the numbers entered is " << mean;

getch();

}



**Code3H.cpp: Evaluating functions.**

#include <iostream>

#include <iomanip>

#define f(t) (1+2\*t\*exp(‐t)) #define g(u,v) (1+u\*v\*sin(u\*v))

using namespace std;

void main()

{

double x,y,z;

// iterations using for

cout << "x" << setw(15) << "y" << setw(15) << "z" << endl; for (x=‐1; x<=1; x+=0.5)

{

y=f(x); z=y/g(x,y);

cout << x << setw(15) << y << setw(15) << z << endl;

}

Cout << endl;

// iterations using while

cout << "x" << setw(15) << "y" << setw(15) << "z" << endl; x=‐1;

while (x<=1)

{

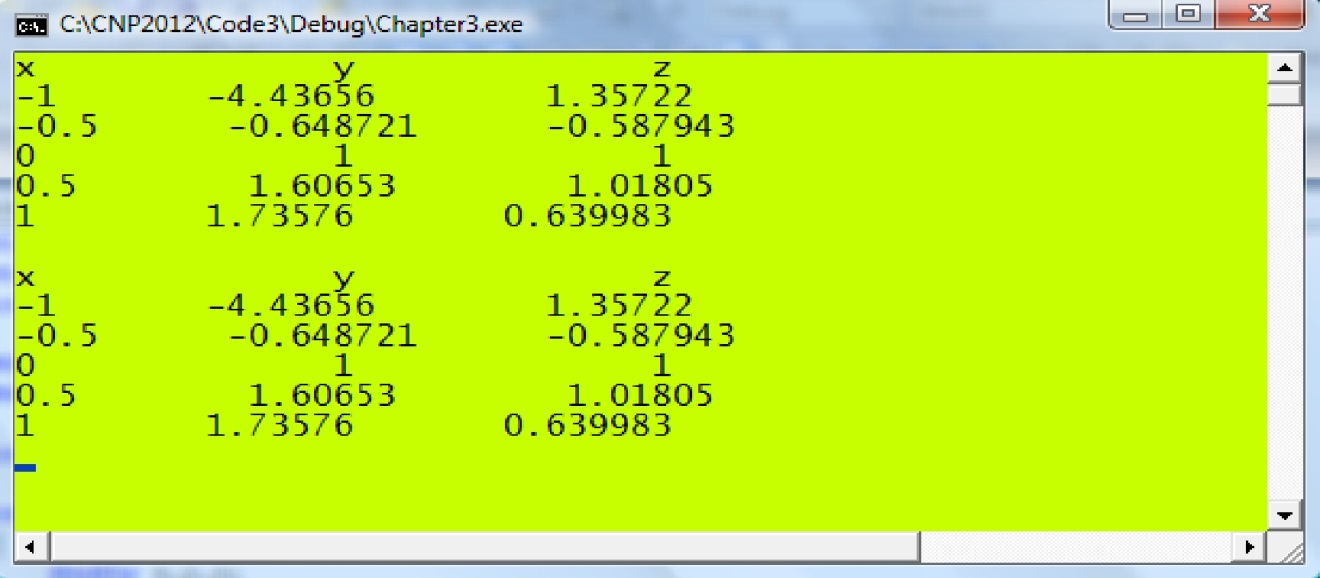
y=f(x); z=y/g(x,y);

cout << x << setw(15) << y << setw(15) << z << endl; x += 0.5;

}

cin.get();

}



**Nested Loop**

**-** a loop within another loop

**Code3I.cpp: Nested loop.**

#include <iostream>

#include <conio.h>

using namespace std;

void main()

{

for (int i=1;i<=3;i++)

{

for (int j=1;j<=2;j++)

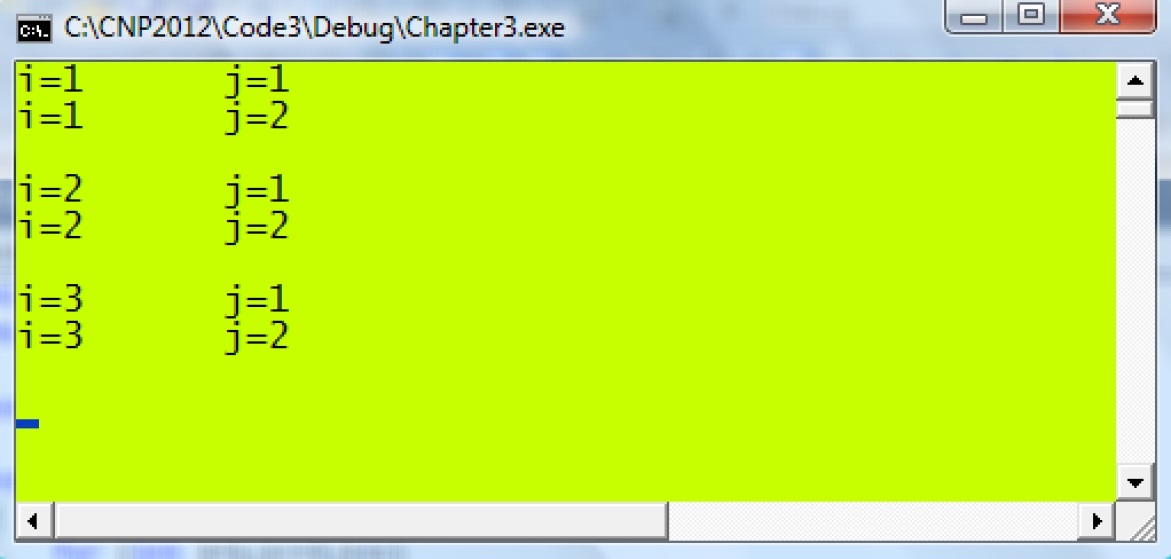
cout << "i=" << i << "\t j=" << j << endl;

cout << endl;

}

getch();

}



|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | for (i=0;i<=2;i++) | | | | | | | |  |  |  |
|  |  |  | | for (j=1;j<=2;j++) | | | | | | |  |  |
|  |  |  | | { | | |  | | |  |  |  |
|  |  |  | | x=5\*i-2\*j; | | | | | |  |  |  |
|  |  |  | | y=i\*j-3; | | | | | |  |  |  |
|  |  |  | | z=x+2\*y; | | | | | |  |  |  |
|  |  |  | | } | | |  | | |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |
|  | *i* |  | *j* |  |  | *x* |  | *y* |  | *z* |  |  |
|  |  |  |  |  |  |  |  |  |  |  | Start |  |
|  |  |  |  |  |  |  |  |  |  |  |  |
| *Iteration* 0 | 0 |  | 1 |  |  | -2 |  | -3 |  | -8 |  |
|  |  |  |  |  |  |  |  |  |  |  |
| *Iteration* 1 |  |  |  |  |  |  |  |  |  |  |  |  |
| 0 |  | 2 |  |  | -4 |  | -3 |  | -10 |  |  |
| *Iteration* 2 |  |  |  |  |  |  |  |  |  |  |  |  |
| 1 |  | 1 |  |  | 3 |  | -2 |  | -1 |  |  |
| *Iteration* 3 |  |  |  |  |  |  |  |  |  |  |  |  |
| 1 |  | 2 |  |  | 1 |  | -1 |  | -1 |  |  |
| *Iteration* 4 |  |  |  |  |  |  |  |  |  |  |  |  |
| 2 |  | 1 |  |  | 8 |  | -1 |  | 6 |  |  |
| *Iteration* 5 |  |  |  |  |  |  |  |  |  |  | End |  |
| 2 |  | 2 |  |  | 6 |  | 1 |  | 8 |  |
|  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |



**Figure 3.4.** Six iterations in the nested loop.

**Code3J.cpp: Iterations on a nested loop.**

#include <iostream>

#include <conio.h>

using namespace std;

void main()

{

int i,j,x,y,z;

for (i=0; i<=2; i++)

for (j=1; j<=2; j++)

{

x=5\*i‐2\*j; y=i\*j‐3; z=x+2\*y;

cout <<"i="<<i<<"\t j="<<j;

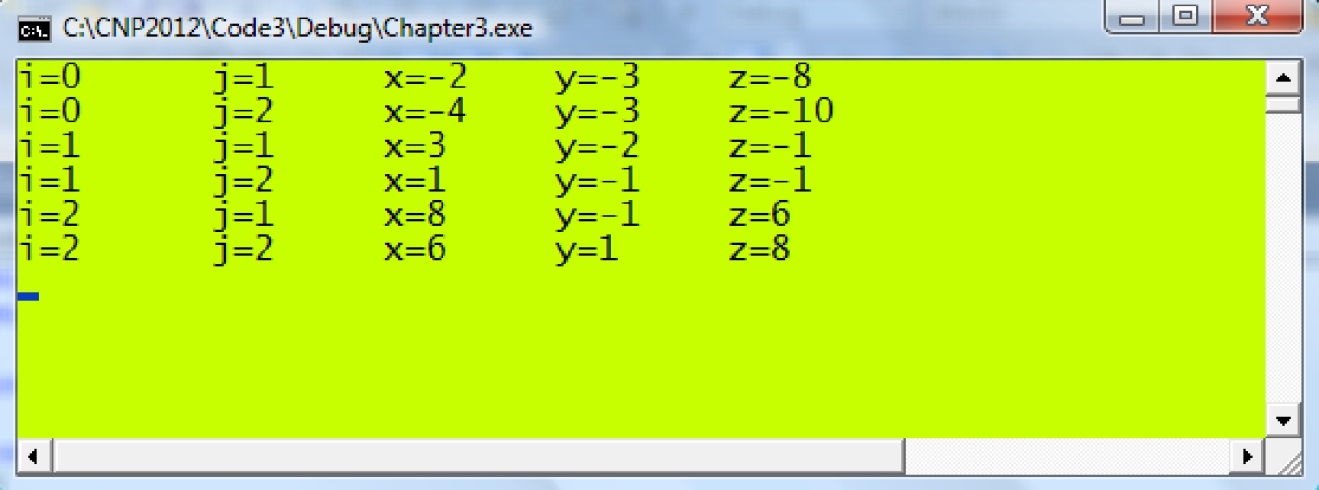
cout <<"\t x="<< x << "\t y="<< y <<"\t z="

<< z << endl;

}

getch();

}



**Code3K.cpp: Displaying data in columns.**

#include <iostream>

#include <conio.h>

using namespace std;

void main()

{

int i,j,m,k=1;

cout << "Enter number of rows:"; cin >> m;

for (i=1;i<=m;i++)

{

for (j=i;j<=m;j++)

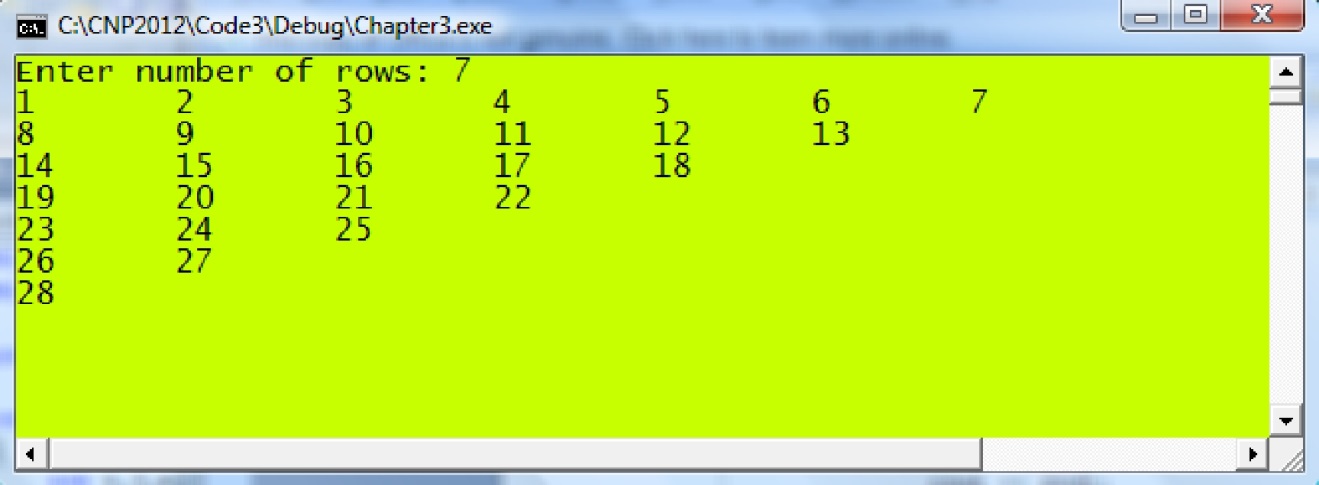
cout << k++ << "\t";

cout << endl;

}

getch();

}

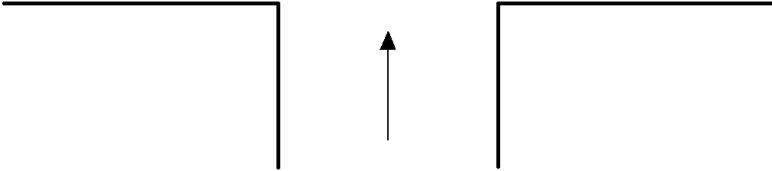


Conditional Branching

if TRUE if FALSE



*Condition*



**Figure 3.5.** Theifroadmap.

The if directive has a body of statements enclosed in the { } brackets according to the following format:

|  |  |  |  |
| --- | --- | --- | --- |
| if (*Condition*) |  |  |  |
| { |  |  |  |
|  |  |  |
| <*Set of Statements* > |  | *if Condition==*TRUE |  |
| } |  |  |  |
|  |  |  |

*Condition* in the above directives refers to a logical expression which returns eitherTRUE (1) or FALSE (0). For example, if x=5, then

if (x>5) returns FALSE,

if (x>=5) returns TRUE,

if (x==5) returns TRUE,

if (x<5) returns FALSE,

if (x<=5) returns TRUE.

Please note

if (x=5) is not a conditional test,

if (x==5) is the right way for testing if x=5.

An expression can also include two or more conditions. This is possible through the Boolean relationships given by AND and OR. Their corresponding representations in C++ are given as follows:

AND is represented as &&,

OR is represented as ||

Both AND and OR require two tokens or expressions, one on the left and another on the right of the operator. The AND expression returns true if only both tokens on the left and right return TRUE. In contrast, the OR expression returns TRUE if one or both tokens agree. For example, the following relationships checks if *x* is a number either greater than 5 or less than -5:

if (x<‐5 || x>5)

While, the following relationship performs a check to see if *x* is a number inside

the interval given by −5 ≤ *x* ≤5 :

if (x>=‐5 && x<=5)

|  |  |  |  |
| --- | --- | --- | --- |
| if (*Condition*) |  |  |  |
| { |  |  |  |
|  |  |  |
| <*Set of Statements* > |  | *if Condition==*TRUE |  |
| } |  |  |  |
|  |  |  |
| else |  |  |  |
| { |  |  |  |
|  |  |  |
| <*Set of Statements* > |  | *if Condition==*FALSE |  |
| } |  |  |
|  |  |  |
| ((*Condition*)? *Statement* 1: *Statement* 2); | | |  |



TRUE FALSE



In the above format, the control flows to *Statement* 1 if *Condition* returns TRUE, and to *Statement* 2 if *Condition* returns FALSE. For example,

y=((x<5)?x\*x:2x‐1);

assigns y=x\*x if x<5, otherwise y=2\*x‐1. This statement is similar to

if (x<5) y=x\*x;

else

y=2x‐1;

**Code3L.cpp: Testing a number.**

#include <iostream>

#include <conio.h>

using namespace std;

void main()

{

int x,y;

cout << "Enter a number:";

cin >> x;

cout << "the entered number is x=" << x << endl; if (x>0)

cout << "x is positive" << endl;

else

cout << "x is negative" << endl;

if (x==0)

cout << "x is zero" <<endl; if (x!=3)

cout << "x is not 3" << endl; if (x>0 && x<=5)

cout << "x is greater than 0 but less than or equal to 5" << endl; if (x<‐2 || x>2)

{

cout << "x is greater than 2, or " << endl; cout << "x is smaller than ‐2" << endl;

}

y=((x>0)?1:‐1);

cout << "Value of y is " << y << endl;

bool a; a=((y>0)?1:0); if (a)

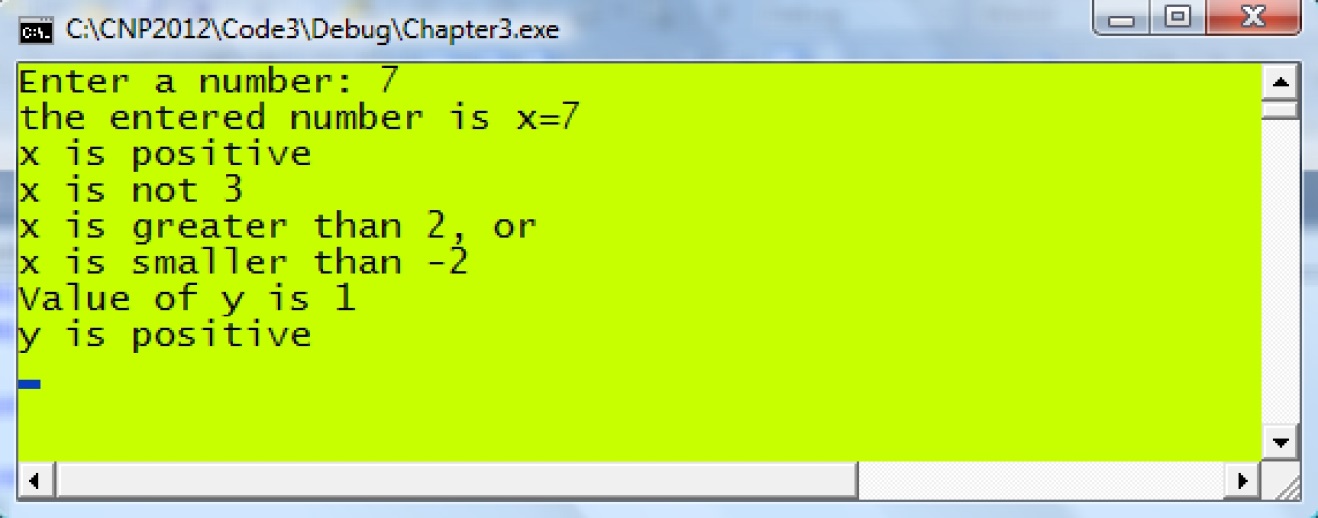
cout << "y is positive" << endl;

else

cout << "y is negative" <<endl;

getch();

}



**Code3M.cpp: Guessing a number.**

#include <iostream>

#include <conio.h> #define N 59

using namespace std;

void main()

{

int i,n; while (1)

{

cout << "Enter a number from 0 to 99 (or 999 to quit): "; cin >> n;

if (n==999) break;

if (n==N)

{

cout << "Congratulation, you got it!" << endl;

break;

}

if (n>N)

cout << "Your guess is too large..." << endl;

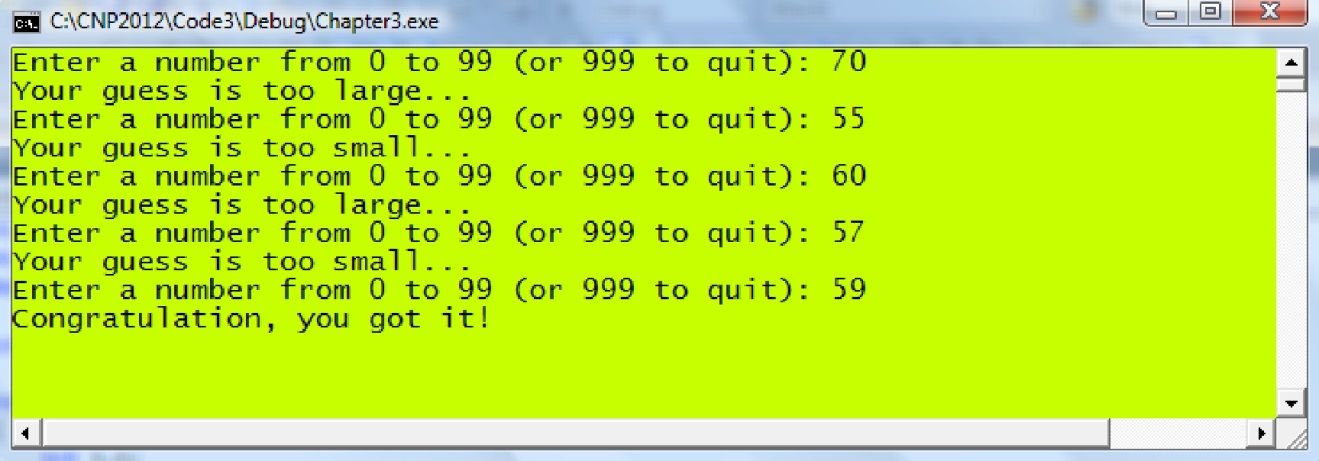
if (n<N)

cout << "Your guess is too small..." << endl;

}

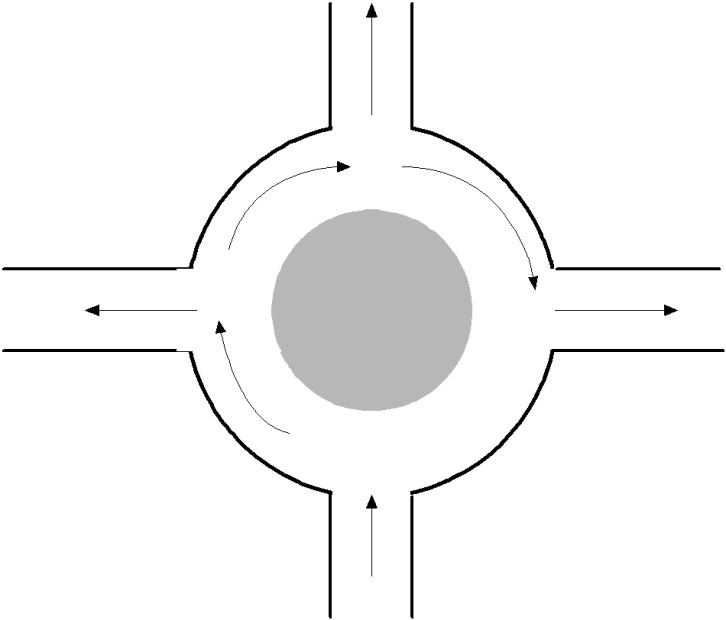
getch();

}



Multiple Branching with switch

case 2



|  |  |
| --- | --- |
| case 1 | case 3 |

*Condition*

**Figure 3.6.** Theswitchroadmap.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| switch (*variable*) |  |  |  |  |
| { |  |  |  |  |
| case *var*1: |  |  |  |  |
|  |  |  |  |
| <*Body*1> |  | if | *variable*=*var*1 |  |
| break; |  |  |  |  |
| case *var*2: |  |  |  |  |
|  | if *variable*=*var*1 | |  |
| <*Body*2> |  |  |
|  |  |  |  |
| break; |  |  |  |  |
|  |  |  |  |
| . |  |  |  |  |
| . |  |  |  |  |
| . |  |  |  |  |
| default: |  | if | *variable* is not |  |
|  |  |
| <*BodyDefault>* |  |  |
|  | any of the above | |  |
| break; |  |  |
|  |  |  |  |
| } |  |  |  |  |

char g=’b’; int x=5,y; switch (g)

{

case ‘a’:

y = 3\*x+1; break;

case ‘b’:

y = 3\*x‐1;

break;

}

case ‘x’:

case ‘X’:

y = 3\*x‐1;

break;

**Code3N.cpp: Multiple branching with switch.**

#include <iostream>

#include <conio.h>

using namespace std;

void main()

{

int qa=0,qb=0,qd=0; char q;

cout << "Type characters, 'x' or 'X' to exit: "; while (1)

{

q=getche();

if (q=='x' || q=='X') break;

switch (q)

{

case 'a': qa++; break;

case 'b': qb++; break;

default: qd++; break;

}

}

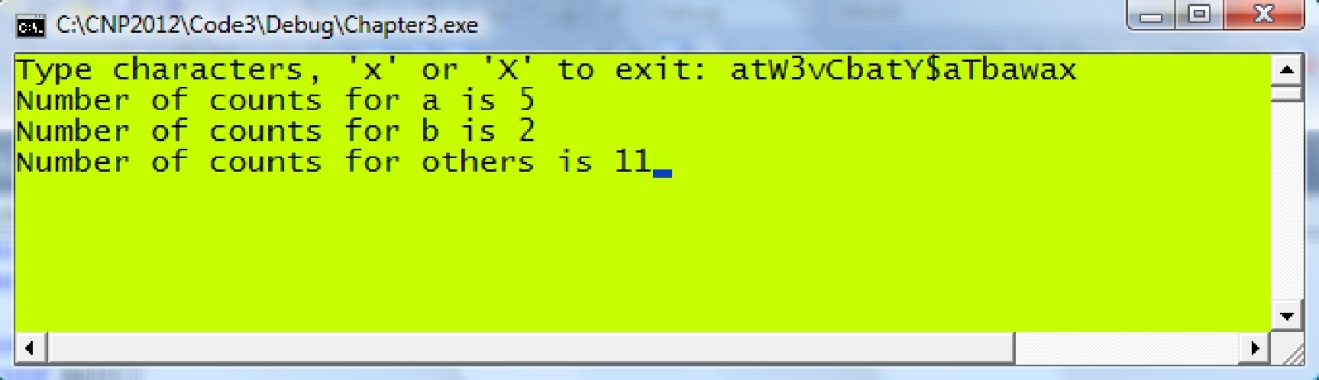
cout << endl << "Number of counts for a is " << qa << endl;

cout << "Number of counts for b is " << qb << endl;

cout << "Number of counts for others is " << qd+1;

getch();

}



MAIN REFERENCE:

Shaharuddin Salleh (2012), C++ Numerical Programming.