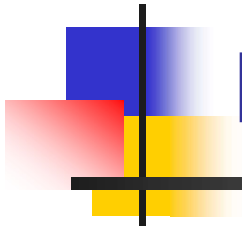
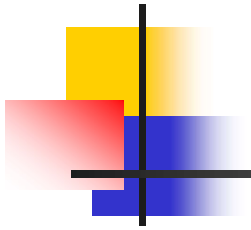


Introduction to Computer Programing C++ Ch2 Flow Control and Logical Operation

National Taiwan University
Dept. of Chemical Engineering
Prof. Chengche Hsu

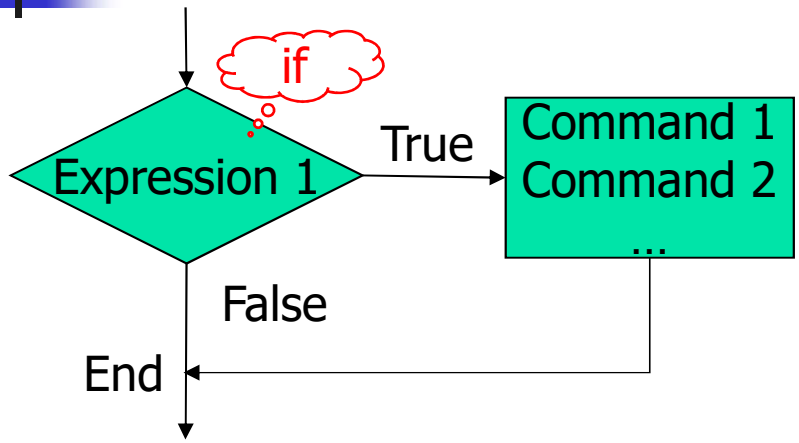
謹記：「ERROR都是暫時的，沒有解不掉的BUG，莫忘心靈祥和。」



What Will We Learn?

- *if….*
- *if….else….*
- *if….else if … else if….else….*
- *switch* (Skip….)
- *Do while* (Skip….)
- *for* loop
- *while* loop

if statement and Rational Operator



Syntax for single action:
if (*condition*) *action statement*

Syntax for multiple actions:
if (*condition*) {
action statement 1
action statement 2
}

Operator	Meaning
<	Less than
<=	Less than or equal to
>	Greater than
>=	Greater than or equal to
==	Equal to
!=	Not equal to

```
....
score=75;
if (score<=60) cout<<"You fail!\n";
....
```

```
....
score=75;
if (score<=60) {
cout<<"You fail!\n";
cout <<"Work harder!\n";
}
....
```

Test Your Understanding

- Ask the user to input two integers between 0 and 100. Display the following if the average is greater than 60:

  You pass!!

- Ask the user to input an integer aa between 0 and 100. Display the following if aa=100:

  Good Job!

  Congratulations!!

- Do the above problem, but replace *if (….)* with *if (bb=100):*

  Congratulations!!

- Check the following

  if (10==3) cout <<"test for 10==3" <<endl;

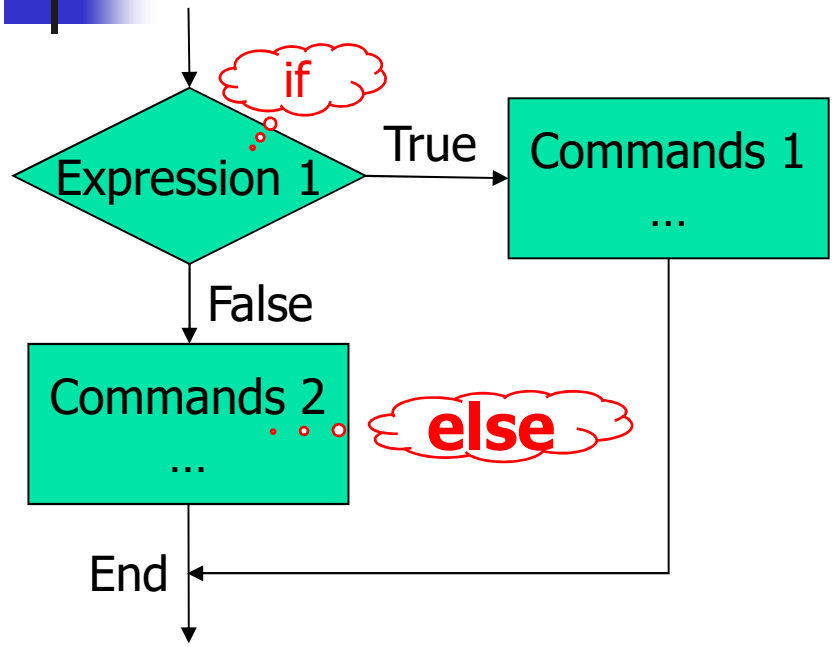
  if (2) cout <<"test for 2 \n";

  if (0) cout <<"test for 0 \n";

  if (10=3) cout <<"test for 10=3 \n";

  int a=3.5; if (a==3) cout <<"test for int a=3.5 \n";

if…else statement



Syntax for single action

If (condition) action statement
Else action statement

Syntax for single or multiple actions:

if (condition) {
action statements
} else {
Action statements
}

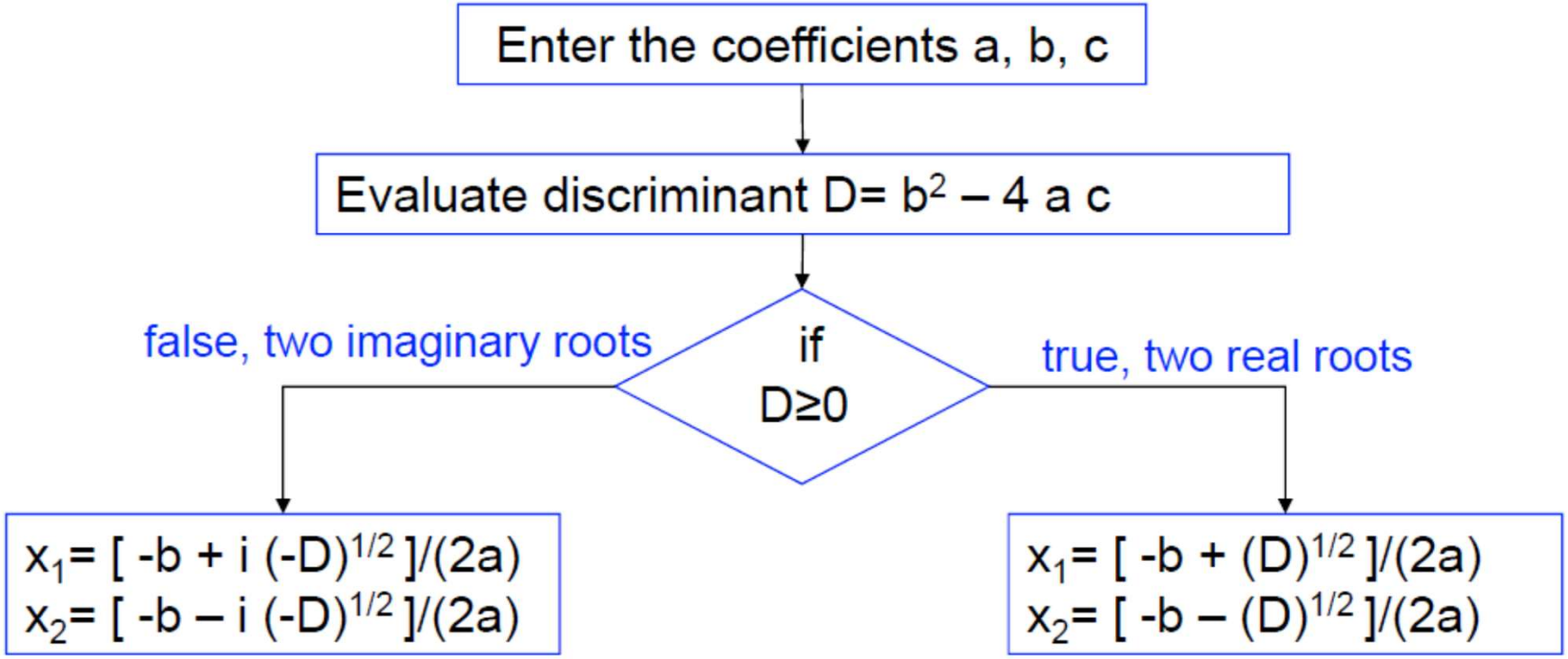
```
...
if (a<60) cout << "You failed!!" << endl;
else cout << "Congratulations"<<endl;
...
```

```
a=75;
if (a<60){
    cout << "You failed!!" << endl;
}else{
    cout << "Congratulations"<<endl;
}
```

Flow for Finding the Roots

Finding the roots of a quadratic equation : $ax^2+bx+c=0$

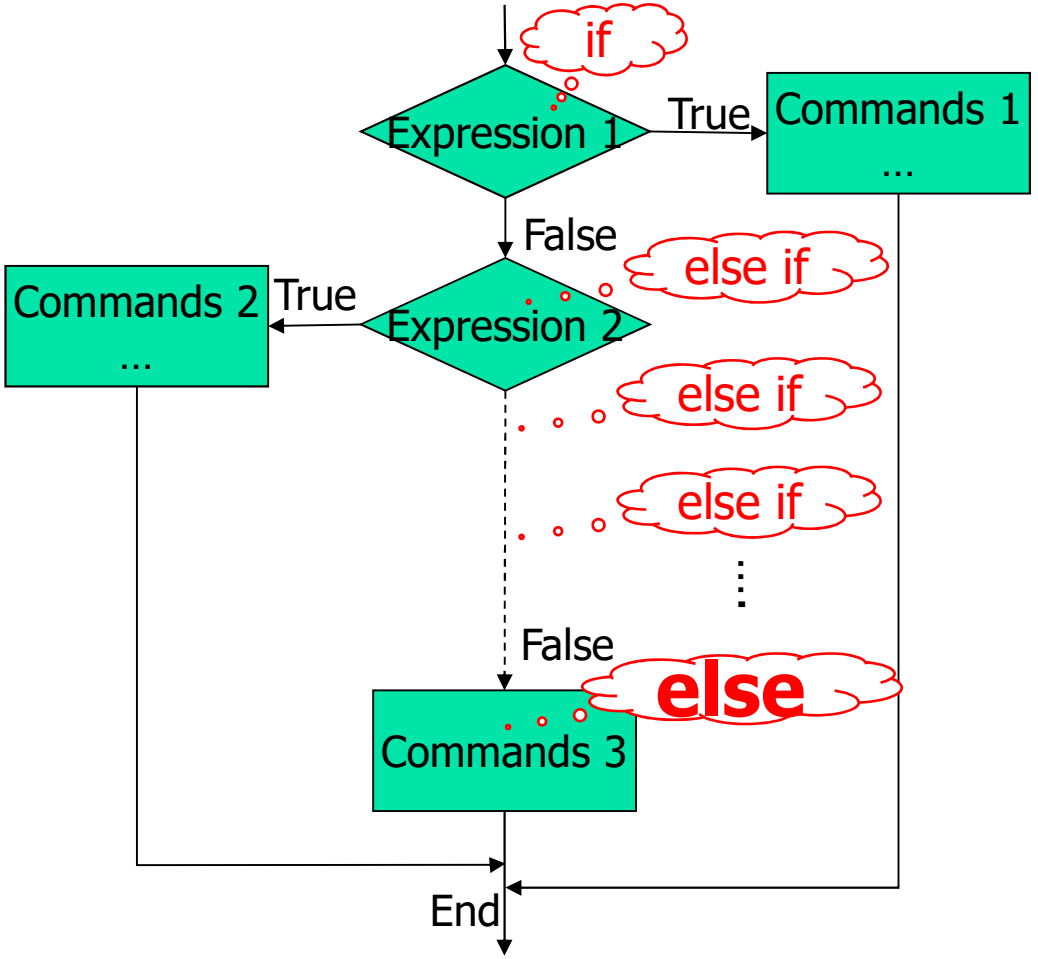
(i) Formulate the flow chart



Acknowledgement: Prof. S. T. Lin

(ii) Coding based on this flow chart

If…else if…else if…else



Syntax:

if (expression 1) {
action statements
} else if (expression 2){
action statements
} else if (expression 3){
action statements
}
….
else {
action statements
}

Logical Operators

Logical Operator	Meaning
&&	and
||	or
!	not

```
int a=3, b=5;
if (a>4 && b>6) cout << "true" << endl;
else cout << "false";
```

```
int a=3, b=5;
if (a>4 || b>6) cout << "true" << endl;
else cout << "false";
```

```
int a=3, b=5;
if (!b>6) cout << "true" << endl;
else cout << "false";
```

Any problem???

Boolean Logic

a	b	expression	value
true	true	a && b	true
true	false		false
false	true		false
false	false		false

a	expression	value
true	! a	false
false		true

a	b	expression	value
true	true	a || b	true
true	false		true
false	true		true
false	false		false

! (a || b) is the same as (!a && !b)
! (a && b) is the same as (!a || !b)

Acknowledgement: Prof. S. T. Lin

Test Your Understanding

- Ask the user to input a real number, then output the square root of this number. (Hint: check the use of the function *sqrt*.)
- (Take home) Ask the user to input the grade (g) of the exam (0-100). If grade<60, the new grade (n_g) is sqrt(grade)*10. If grade >=60, n_g is grade+20. But the maximum grade has to be no more than 100. Properly calculate and display the grade and new grade.
- (Take home) Ask the user to input three real numbers: a, b, and c. Find and display the roots of $ax^2+bx+c=0$.
- (Take home) Ask the user to input two numbers a and b, compare the values and properly display the results. Ex. a is greater (or less) than (or equal to) b.
- Ask the user to input a number x, find and properly display the output of the following function:

  x<0, y=abs(x);

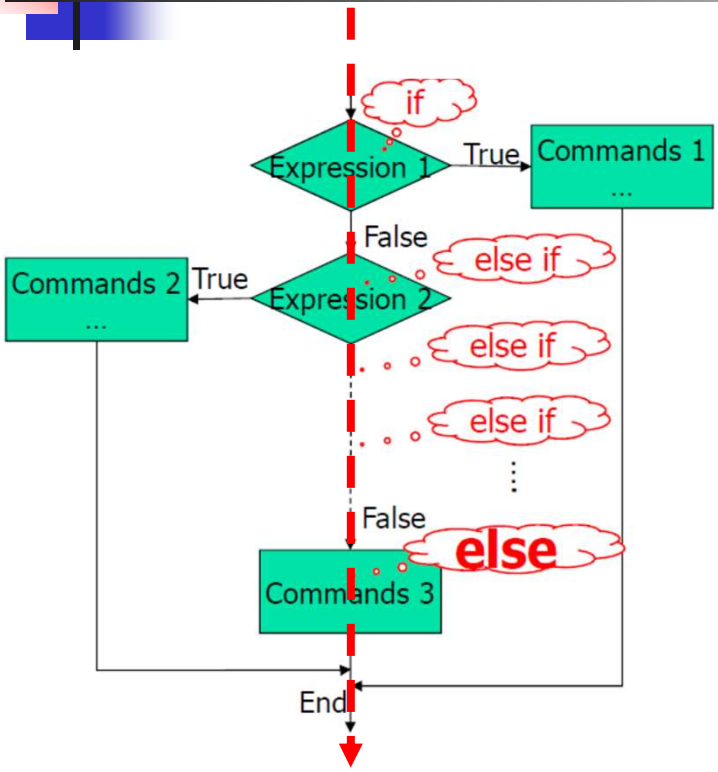
  0<=x<3, y=2*x;

  x>=3, y=sin(x)

  - Repeat the above problem, for x>=3, y=sqrt(sin(x))

Hint: Try not to finish all at once. Code a portion, test, then move on….

Loops



What if you need to repeat actions multiple times?

What if you need to:

1+2+3+….+n (with known or unknown n)

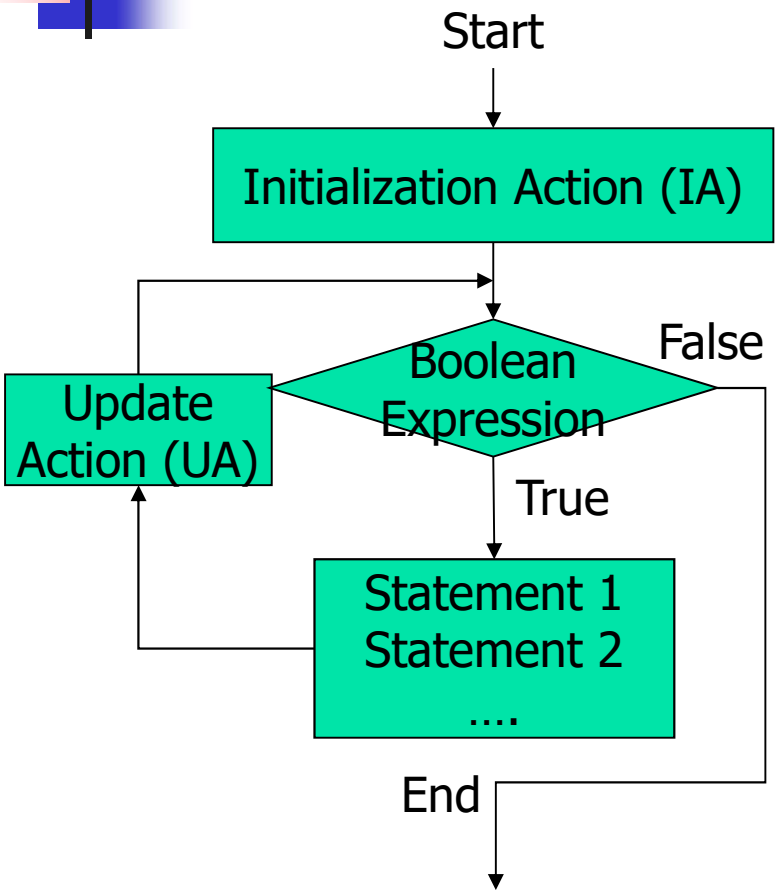
Sin(1)+sin(2)+….

Add/read multiple inputs (with known or unknown number of inputs)

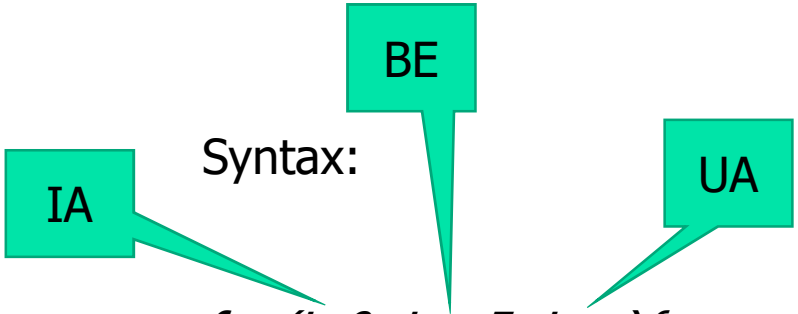
Check and see if a number is a prime number.

….

for loop



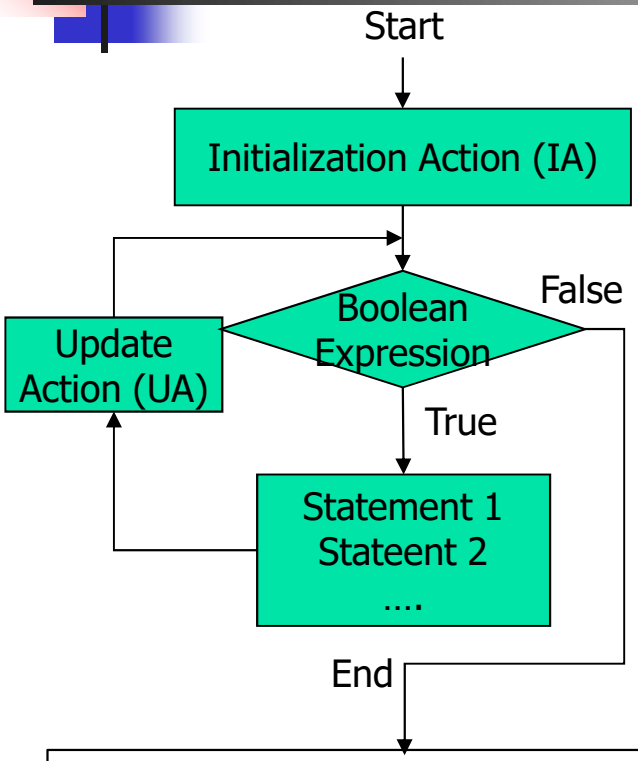
Go through the loop for given (known) times.

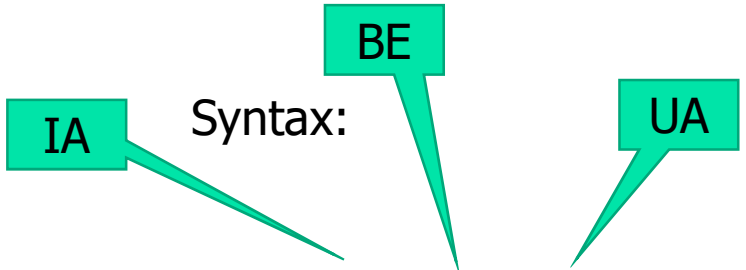


What does this do?

```
int i;
for (i=0 ; i<3 ; i++){
    cout << "i=" << i <<endl;
}
```

for loop



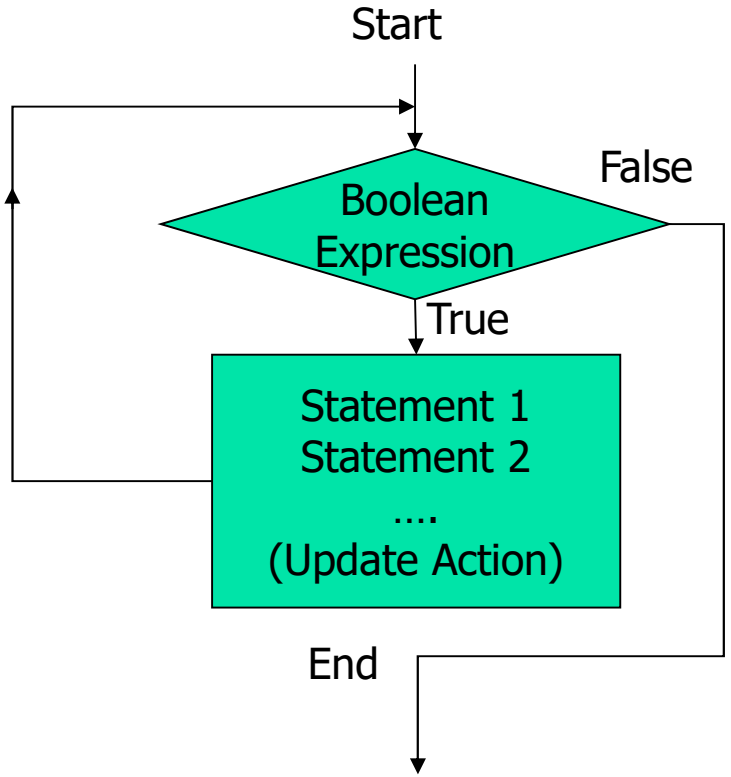


What do these do?

```
int i, sum;
sum = 0;
for (i=0 ; i<5 ; i++){
    sum=sum+i;
    cout << "sum=" << sum <<endl;
}
```

```
int i, sum;
sum = 0;
for (i=0 ; i=5 ; i++){
    sum=sum+i;
    cout << "sum=" << sum <<endl;
}
```

While loop



Go through the loop for uncertain times.

Syntax: BE

```
while (i<=5){
Statement 1;
Statement 2;
....
}
```

UA

```
int sum = 0, i=0;
while (i<5){
  sum=sum+i;
  cout << "sum=" << sum <<endl;
  i++;
}
```

UA

```
int i=0, sum=0;
while (sum<=100){
  sum=sum+i;
  cout << "sum=" << sum <<endl;
  i++;
}
  cout << i-1;
```

UA

Test Your Understanding

For loop:

- Calculate: 1+2+… +100
- Ask the user to input a positive integer, then calculate and display n!
- Calculate 1*3*5+2*4*6+3*5*7+...+n*(n+2)*(n+4) for n=10.
- Find the approximated value of $\int_0^2 x^2 \, dx$ with 100 increments and compare with the actual number.

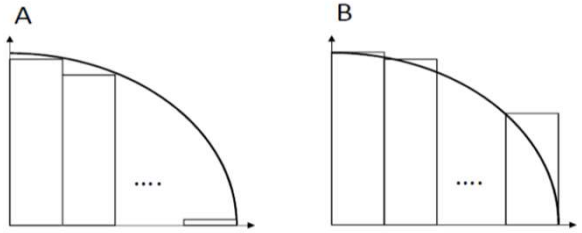
While loop:

- Calculate 1+2+….+n, until the summation is greater than 1000. Display n and the summation.
- Check the above calculation using a for loop, compare the results of the two, and properly display.
- Calculate 1*3*5+2*4*6+3*5*7+...+n*(n+2)*(n+4) when the summation is greater than 20000. Properly display sum and n.

Test Your Understanding

Challenging Problems

- Following are curves for $y = \sqrt{1 - x^2}$ (part of $x^2 + y^2 = 1$), the area underneath is pi/4 ($\int_0^1 \sqrt{1 - x^2}\, dx$). Use a given n (increments) and integration, find the approximated pi.



- Pi/4 = 1 - 1/3 + 1/5 - 1/7 + ...
  - Find the approximated pi using 100 term approximation.
  - Find the number of terms needed when the error is below 0.0001.
- Ask the user to input two positive integers a and b. Properly display if b is a factor(因數) of a.
- Ask the user to input a positive integer, then find the number of positive factors (正因數數量)
- Ask the user to input a positive integer, then check and display whether it is a prime number.