Variable Declaration

```
type variable_name;
```

Common data types

- `int` - integer numbers
- `double` - floating point numbers
- `bool` – logical type (true, false)
- `char` – characters

Variable's name

- Series of characters (letters, digits, underscores "_")
- Cannot begin with digit.
- Not include "__" (double underscore).
- C++ is case sensitive i.e., a1 and A1 represent different variables.
- Must NOT be a keyword.

```
alignas, alignof, and, and_eq, asm, auto, bitand, bitor, bool, break, case, catch, char, char16_t,
char32_t, class, compl, const, constexpr, const_cast, continue, decltype, default, delete, do,
double, dynamic_cast, else, enum, explicit, export, extern, false, float, for, friend, goto, if,
inline, int, long, mutable, namespace, new, noexcept, not, not_eq, nullptr, operator, or, or_eq,
private, protected, public, register, reinterpret_cast, return, short, signed, sizeof, static,
static_assert, static_cast, struct, switch, template, this, thread_local, throw, true, try, typedef,
typeid, typename, union, unsigned, using, virtual, void, volatile, wchar_t, while, xor, xor_eq
```

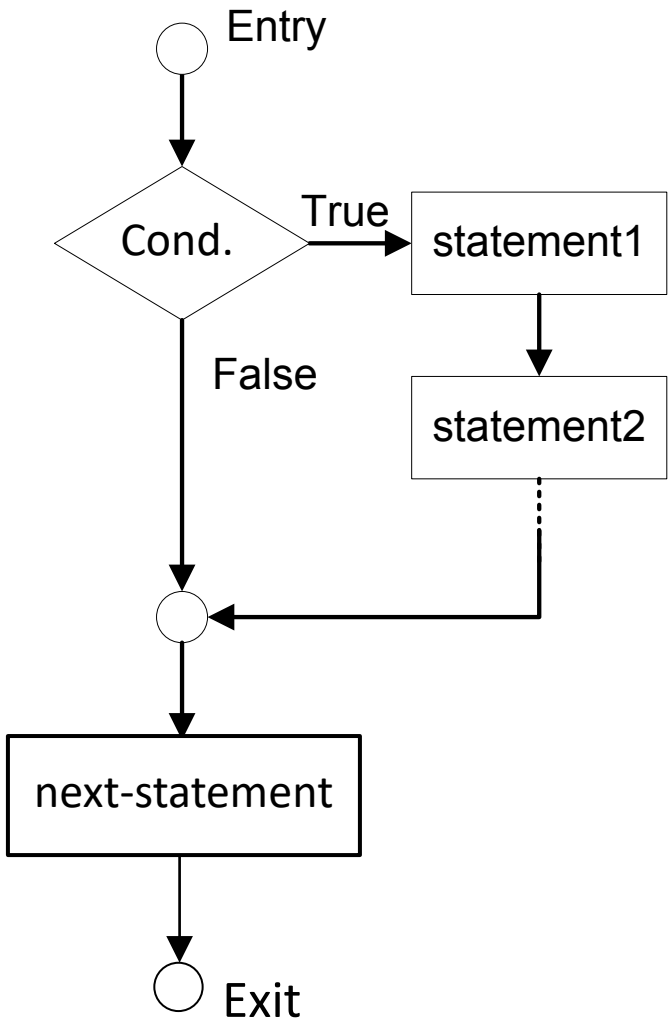
Example 2-C: Arithmetic Operators

```
1   #include <iostream>
2
3   int main()
4   {
5     float x, y, z;
6
7     std::cout << "Enter first number: ";
8     std::cin >> x;
9
10    std::cout << "Enter second number: ";
11    std::cin >> y;
12
13    z = 2*x*y/(x+y);
14
15    std::cout << "Hamonic mean of " << x << " and " << y << " is " << z;
16
17    return 0;
18  }
```

```
Enter first number: 1.045454
Enter second number: 0.522727
Hamonic mean of 1.04545 and 0.522727 is 0.696969
```

if Statement

```
if (condition) {
   statement1;
   statement2;
   ...
}
next-statement;
```



- If the `condition` is `true`
  - `statement1, statement2, …` are executed in order then program continues to **`next-statement`**
- If the `condition` is `false`
  - `statement1, statement2, …` are ignored and program continues to **`next-statement`**
- Use { … } to define the block of compound statements

Example 3-A: Chat Bot v0.1

```
1  #include <iostream>
2  #include <string>
3  using namespace std;
4
5  int main()
6  {
7      string name;
8      cout << "Who are you?\n";
9      cout << ">>";
10     cin >> name;
11
12     if(name == "Karn"){
13         cout << "Oh, you look so good. (>w<) \n";
14     }
15
16     cout << "Nice to meet you.";
17
18     return 0;
19 }
```

```
Who are you?
>>Karn
Oh, you look so good. (>w<)
Nice to meet you.
```

```
Who are you?
>>Kasemsit
Nice to meet you.
```

Note1 : `using` statements

- Eliminate use of `std::` prefix
- Write `cout` instead of `std::cout`

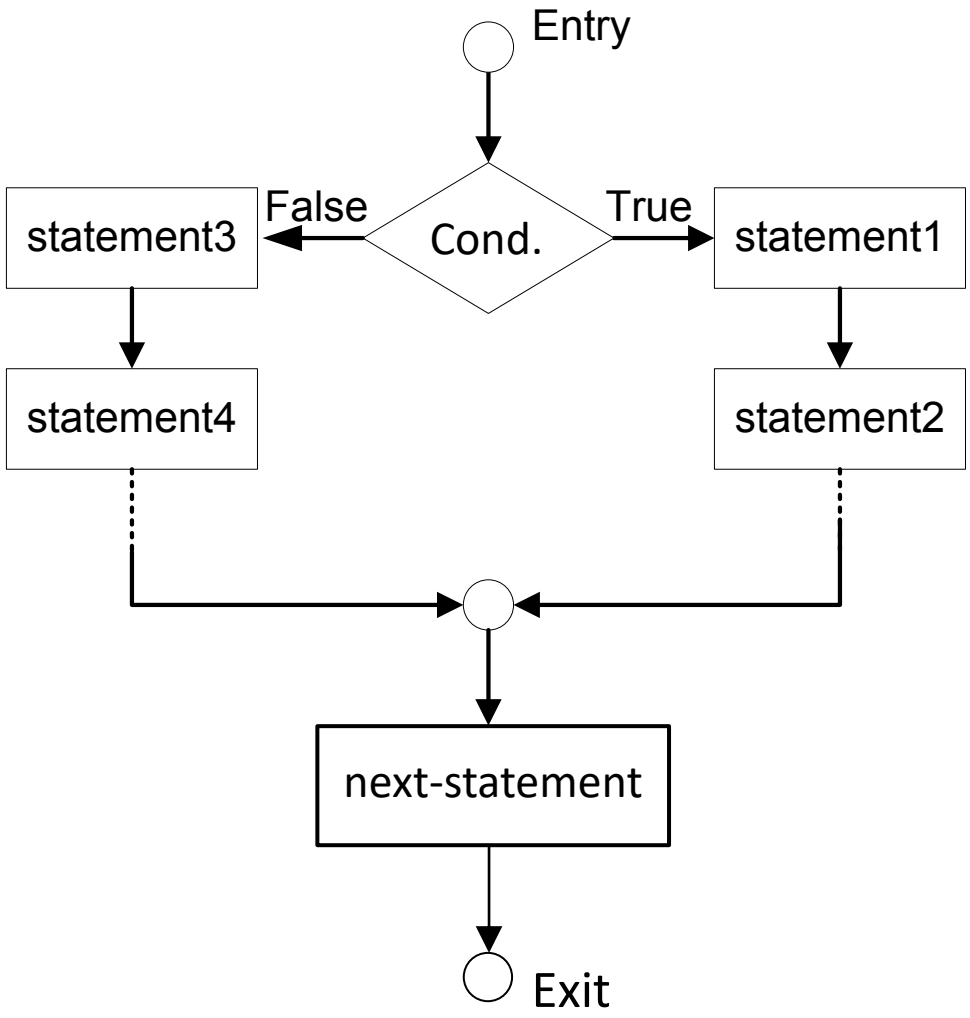
Note2 : `string` type of variable

- Used for storing text
- Require `#include<string>`

Note3: Indenting makes programs easier to read

if-else Statement

```
if (condition) {
   statement1;
   statement2;
   …
} else {
   statement3;
   statement4;
   …
}
next-statement;
```



Example 3-B: Find Absolute Value

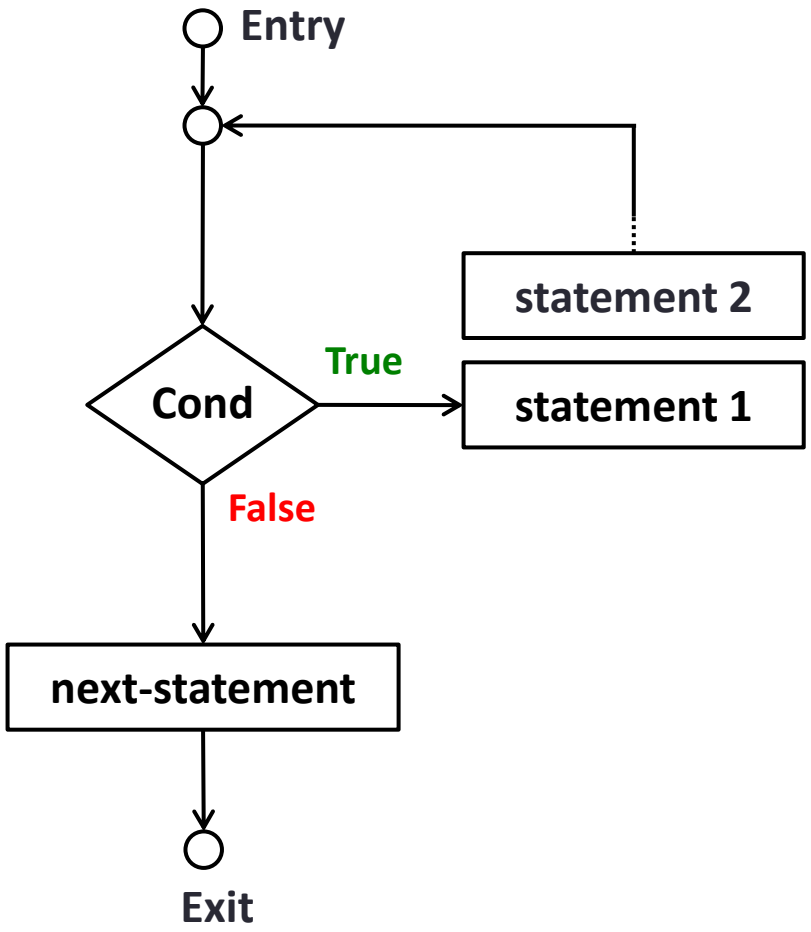
```
1  #include <iostream>
2  using namespace std;
3
4  int main()
5  {
6      double x;
7      cout << "Enter x: ";
8      cin >> x;
9
10     if(x >= 0){
11         cout << "|x| = " << x;
12     }else{
13         cout << "|x| = " << -x;
14     }
15
16     return 0;
17 }
```

Enter x: -2.15
|x| = 2.15

Enter x: 0.69
|x| = 0.69

while Statement

```
while (condition) {
   statement1;
   statement2;
   ...
}
next-statement;
```



- While the `condition` is `true`
  - `statement1, statement2, …` are repeated again and again
- Until the `condition` is `false`
  - `statement1, statement2, …` are ignored and program continues to `next-statement`

while Statement

```
1  #include <iostream>
2  using namespace std;
3
4  int main()
5  {
6      int x = 1;
7      while(x < 5){
8          cout << "Print " << x << endl;
9          x = x+1;
10     }
11     return 0;
12 }
```

```
Print 1
Print 2
Print 3
Print 4
```

Note1: We can define value of the variable during declaration

Note2 : We can use `endl` for new line. (It is similar to `"\n"`);

- `cout << endl;` is equivalent to `cout << "\n";`
- `cout << "A" << endl;` is equivalent to `cout << "A\n";`

Function Definition

```
return-value-type  function-name ( parameter-list )
{
    declarations and statements
}
```

- Parameter list
  - Comma separated list of arguments
    - Data type needed for each argument
  - If no arguments, use **`void`** or leave blank
- Return-value-type
  - Data type of result returned (use **`void`** if nothing returned)

Example 4-A: Area of a Circle

```
1  #include<iostream>
2  using namespace std;
3
4  double findCircleArea(double r){
5      double A = 3.1416*r*r;
6      return A;
7  }
8
9  int main(){
10     double A1 = findCircleArea(1);
11     cout << "A1 = " << A1 << "\n";
12
13     double r2 = 2;
14     double A2 = findCircleArea(r2);
15     cout << "A2 = " << A2 << "\n";
16
17     cout << "A3 = " << findCircleArea(3) << "\n";
18
19     return 0;
20 }
```

return value from findCircleArea(1)

Output

A1 = 3.1416
A2 = 12.5664
A3 = 28.2744

return value from findCircleArea(r2)

return value from findCircleArea(3)

Example 4-C: Find Absolute

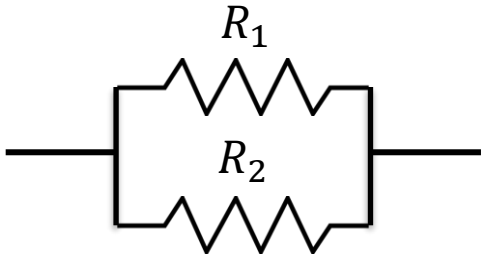
```
1  #include<iostream>
2  using namespace std;
3
4  double findAbsolute(double x){
5      if(x >= 0){
6          return x;
7      }else{
8          return -x;
9      }
10 }
11
12 int main(){
13     cout << "|8| = " << findAbsolute(8) << "\n";
14     cout << "|-2| = " << findAbsolute(-2) << "\n";
15
16     return 0;
17 }
```

Output

```
|8| = 8
|-2| = 2
```

Example 4-D: Parallel Resistors

$$R = R_1 + R_2$$



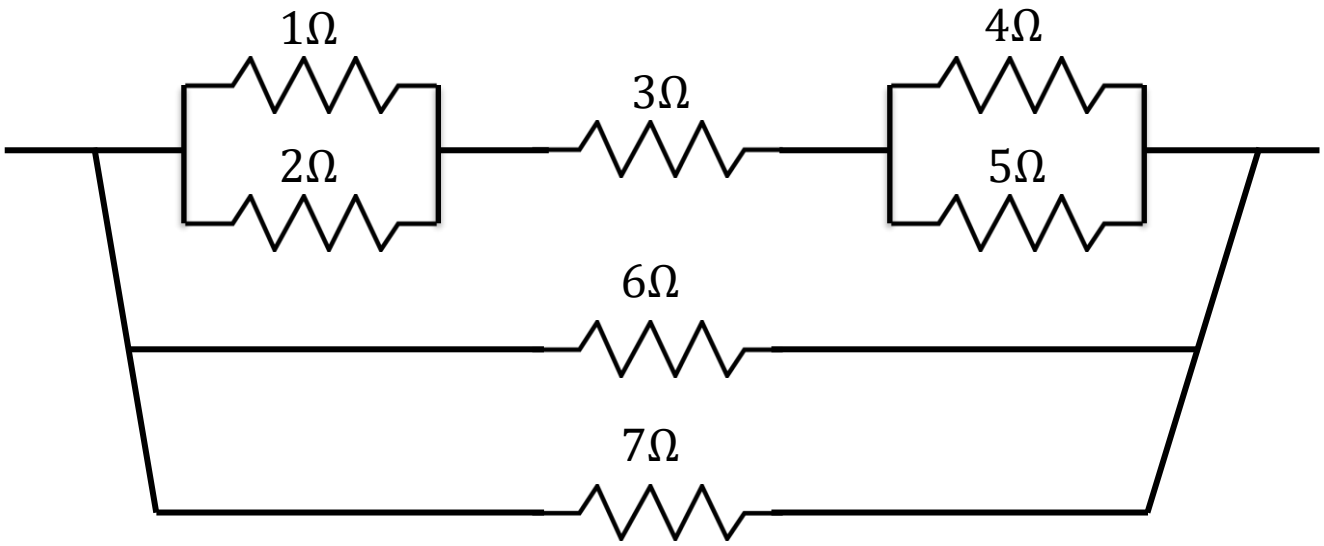
$$R = \frac{R_1 R_2}{R_1 + R_2}$$

```
1  #include<iostream>
2  using namespace std;
3
4  double paraR(double r1,double r2){
5      return r1*r2/(r1+r2);
6  }
7
8  int main(){
9      cout << "2.4 ohm || 2.4 ohm = " << paraR(2.4,2.4) << " ohm\n";
10     cout << "2 ohm || 3 ohm = " << paraR(2,3) << " ohm\n";
11     cout << "2 ohm || 3 ohm || 5 ohm = " << paraR( paraR(2,3) , 5 )<< " ohm\n";
12     return 0;
13 }
```

Output

```
2.4 ohm || 2.4 ohm = 1.2 ohm
2 ohm || 3 ohm = 1.2 ohm
2 ohm || 3 ohm || 5 ohm = 0.967742 ohm
```

Example 4-D: Parallel Resistors



paraR(paraR(paraR(1,2) + 3 + paraR(4,5) , 6) , 7)

Example 4-E: Find GCD (brute force)

```
1   #include<iostream>
2   using namespace std;
3
4   int findGCD(int x, int y){
5       int min;
6       if(x <= y){
7           min = x;
8       }else{
9           min = y;
10      }
11
12      int i = min;
13      while(i > 0){
14          if(x%i == 0 and y%i == 0){
15              return i;
16          }else{
17              i--;
18          }
19      }
20  }
21
22  int main()
23  {
24      cout << "GCD(12,8) = " << findGCD(12,8) << "\n";
25      cout << "GCD(240,750) = " << findGCD(240,750) << "\n";
26
27      return 0;
28  }
```

Output

GCD(12,8) = 4
GCD(240,750) = 30

return the divisor i and immediately stop the loop when the divisor is found.

Function that Returns No Value

```
1   #include <iostream>
2   using namespace std;
3
4   void printCPP(int N){
5       int i = 1;
6       while(i<=N){
7           cout << "C++\n";
8           i++;
9       }
10  }
11
12  int main()
13  {
14      cout << "I LOVE\n";
15      printCPP(1);
16      cout << "YOU LOVE\n";
17      printCPP(1);
18      cout << "ALL WE NEED IS\n";
19      printCPP(4);
20      return 0;
21  }
```

Use `void` as a return-value type

void function-name (*parameter-list*)
{
    declarations and statements
}

Output

```
I LOVE
C++
YOU LOVE
C++
ALL WE NEED IS
C++
C++
C++
C++
```

Function with Empty Parameter Lists

`void` or leave parameter list empty indicates function takes no arguments

```
#include <iostream>
using namespace std;

void printSmile(){
    int i = 1;
    while(i<=69){
        cout << ":)";
        if(i%23==0) cout << '\n';
        i++;
    }
}

int main()
{
    printSmile();
    return 0;
}
```

return-value-type function-name (*void*)
{
declarations and statements
}

return-value-type function-name ()
{
declarations and statements
}

Output

```
:):):):):):):):):):):):):):):):):):):):):):):)
:):):):):):):):):):):):):):):):):):):):):):):)
:):):):):):):):):):):):):):):):):):):):):):):)
```

Mathematical Functions with C++

```cpp
#include <iostream>
#include <cmath>
using namespace std;

int main()
{
    cout << "3^4 = " << pow(3,4) << "\n";
    cout << "sqrt(10) = " << pow(10,0.5)  << "\n";
    cout << "sqrt(10) = " << sqrt(10) << "\n";
    cout << "e^(-1.5) = " << exp(-1.5) << "\n";
    cout << "ln(7) = " << log(7) << "\n";
    cout << "log10(7) = " << log10(7) << "\n";
    cout << "log2(7) = " << log2(7) << "\n";
    cout << "sin(30 deg) = " << sin(30*M_PI/180) << "\n";
    cout << "cos(pi/2) = " << cos(M_PI/2) << "\n";
    return 0;
}
```

Output

```
3^4 = 81
sqrt(10) = 3.16228
sqrt(10) = 3.16228
e^(-1.5) = 0.22313
ln(7) = 1.94591
log10(7) = 0.845098
log2(7) = 2.80735
sin(30 deg) = 0.5
cos(pi/2) = 1
```

Declaring Arrays

type arrayName [*arraySize*] ;

- When declaring arrays, specify
  - Name
  - Type of array - any data type
  - Number of elements

```
int c[10];      // array of 10 integers
float d[3284]; // array of 3284 floats
```

- Declaring multiple arrays of same type
  - Use comma separated list, like regular variables

```
int b[100], x[27];
```

Example 5-D: Find Max

```
1   #include<iostream>
2   using namespace std;
3   int main(){
4
5       int N = 10;
6       double x[N] = {1.1,2.5,-3.1,4.0,6.9}, max = x[0];
7
8       int i = 1;
9       while(i < N){
10          if(x[i] > max){
11              max = x[i];
12          }
13          i++;
14      }
15
16      cout << "max = " << max;
17
18      return 0;
19  }
```

Output

```
max = 6.9
```

Example 5-E: Collect Input Data in Array

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

    int N;
    cout << "Enter the number of students: ";
    cin >> N;

    double x[N];
    int i = 0;
    while(i < N){
        cout << "Enter score of student [" << i+1 << "]: ";
        cin >> x[i];
        i++;
    }

    int j = 0;
    double sum = 0;
    while(j < N){
        sum += x[j];
        j++;
    }
    cout << "Average score is " << sum/N;

    return 0;
}
```

Output

```
Enter the number of students: 6
Enter score of student [1]: 10
Enter score of student [2]: 20
Enter score of student [3]: 30
Enter score of student [4]: 40
Enter score of student [5]: 50
Enter score of student [6]: 63
Average score is 35.5
```

String (std::string)

- Series of characters treated as single unit
- Can include letters, digits, special characters **+**, **−**, ***** ...
- String literal enclosed in " " (double quotes) , for example:

```
"I like C++"
```

- #include <string> in preprocessor

```
1  #include <iostream>
2  #include <string>
3
4  int main()
5  {
6    std::string sub, verb, comp, full_sentense;
7    sub = "Luffy";
8    verb = "will become";
9    comp = "the prirate king";
10   full_sentense = sub+' '+verb+" "+comp+"!!!";
11   std::cout << full_sentense;
12 }
```

```
Luffy will become the prirate king!!!
```

String Concatenation

```cpp
#include<iostream>
#include<string>
using namespace std;

int main(){
    string x = "Hello";
    string y = "World";
    string z = x+y;
    cout << x << "\n";
    cout << y << "\n";
    cout << z << "\n";
    return 0;
}
```

Output

```
Hello
World
HelloWorld
```

Access Character in String

```cpp
#include<iostream>
#include<string>
using namespace std;

int main(){

    string x = "SMILE";

    int i = 0;
    while(i < 5){
        cout << x[i] << "\n";
        i++;
    }

    return 0;
}
```

Output

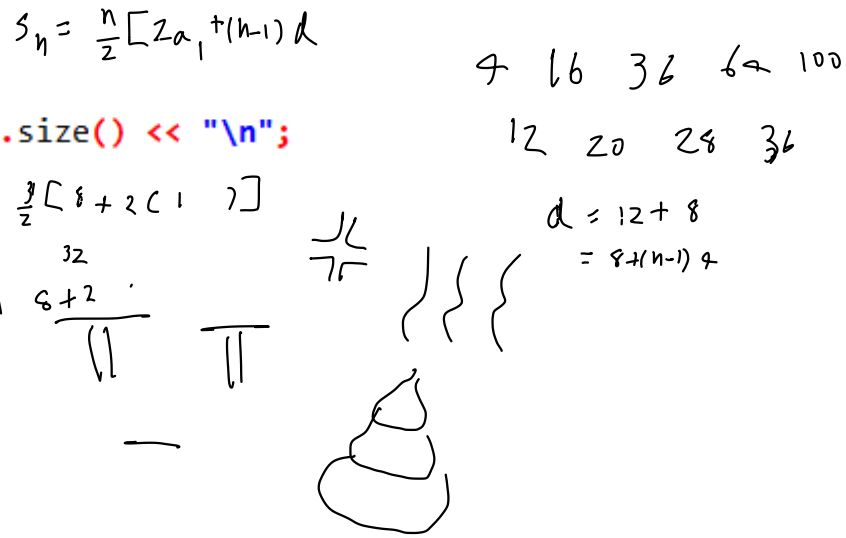
```
S
M
I
L
E
```

Example 5-F: Reverse String

```
1  #include<iostream>
2  #include<string>
3  using namespace std;
4
5  int main(){
6
7      string s;
8      cout << "Enter string: ";
9      cin >> s;
10
11     int L = s.size();
12     cout << "String Length: " << s.size() << "\n";
13
14     cout << "Reverse String: ";
15     int i = 0;
16     while(i < L){
17         cout << s[L-i-1];
18         i++;
19     }
20
21     return 0;
22 }
```

Output

```
Enter string: HelloWorld
String Length: 10
Reverse String: dlroWolleH
```



Example 5-F: Reverse String

```cpp
#include<iostream>
#include<string>
using namespace std;

string reverseString(string x){
    string y = x;
    int L = x.size();

    int i = 0;
    while(i < L){
        y[i] = x[L-i-1];
        i++;
    }

    return y;
}

int main(){
    string s;
    cout << "Enter string: ";
    cin >> s;

    string r = reverseString(s);
    cout << "Reverse String: " << r;

    return 0;
}
```

Output

```
Enter string: HelloWorld
Reverse String: dlroWolleH
```

Example 7-Q: Logical Error (bool operators)

```
#include <iostream>

using namespace std;

int main()
{
     int x;
     cin >> x;
     if(x == 4 or 5)
     {
          cout << "OK";
     }
}
```

```
3
OK
```

```
#include <iostream>

using namespace std;

int main()
{
     int x;
     cin >> x;
     if(x == 4 or x == 5)
     {
          cout << "OK";
     }
}
```

```
3
```

Example 7-R: Logical Error (bool operators)

```
#include <iostream>

using namespace std;

int main()
{
     int x;
     cin >> x;
     if(x = 55)
     {
          cout << "OK";
     }
}
```

```
69
OK
```

```
#include <iostream>

using namespace std;

int main()
{
     int x;
     cin >> x;
     if(x == 55)
     {
          cout << "OK";
     }
}
```

```
69
```

Example 7-S: Logical Error (integer division)

```
#include <iostream>

using namespace std;

int main()
{
     float x;
     cin >> x;
     cout << 1/2*(x*x);
}
```

```
10
0
```

```
#include <iostream>

using namespace std;

int main()
{
     float x;
     cin >> x;
     cout << 1.0/2*(x*x);
}
```

```
10
50
```

Value Assignment

- **Input stream object**

```
1  #include <iostream>
2  using namespace std;
3
4  int main()
5  {
6      int a,b,c,x;
7      cout << "Input 3 numbers: ";
8      cin >> a >> b >> c;
9
10     cout << "Average = " << (a+b+c)/3.0;
11
12     cout << "\nInput another number: ";
13     cin >> x;
14     cout << "Another number is " << x;
15
16     return 0;
17 }
```

```
Input 3 numbers: 15 20 30 40 50 60 70
Average = 21.6667
Input another number: Another number is 40
```

อ้าว..ไม่ได้พิมพ์ ข้ามไปเลย

Value Assignment

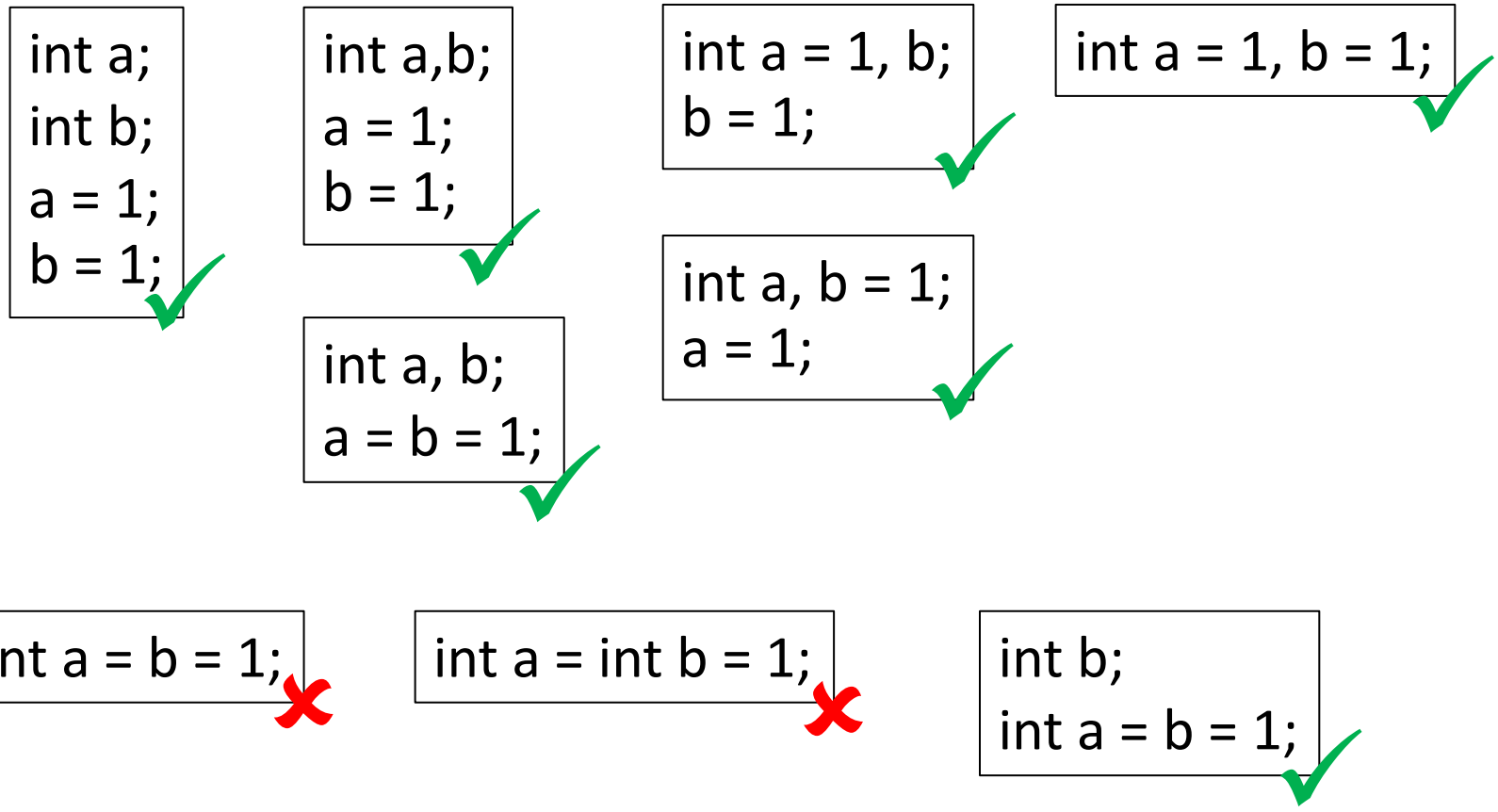
- **= (Assignment operator)**
  - Assigns value to variable
  - Copy value on the right side to the variable on the left side
  - Example:

```
PI = 3.1416;
sum = variable1 + variable2;
a = b = 5;

int x = a + 1;
double x = 5.5;
char char5 = '5', at = '@';
string mylove = "I love you.";
```

Initialization

Value Assignment

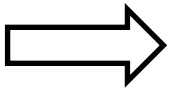
- = (Assignment operator)



Fundamental Data Types

Character types

```
1  #include <iostream>
2  using namespace std;
3
4  int main() {
5      char A = '+';
6      char x = A;
7      char y = 'A';
8
9      cout << "x = " << x << endl;
10     cout << "y = " << y << endl;
11
12     return 0;
13 }
```



```
x = +
y = A
```

Fundamental Data Types

Boolean type

- Representing logical data (true, false).

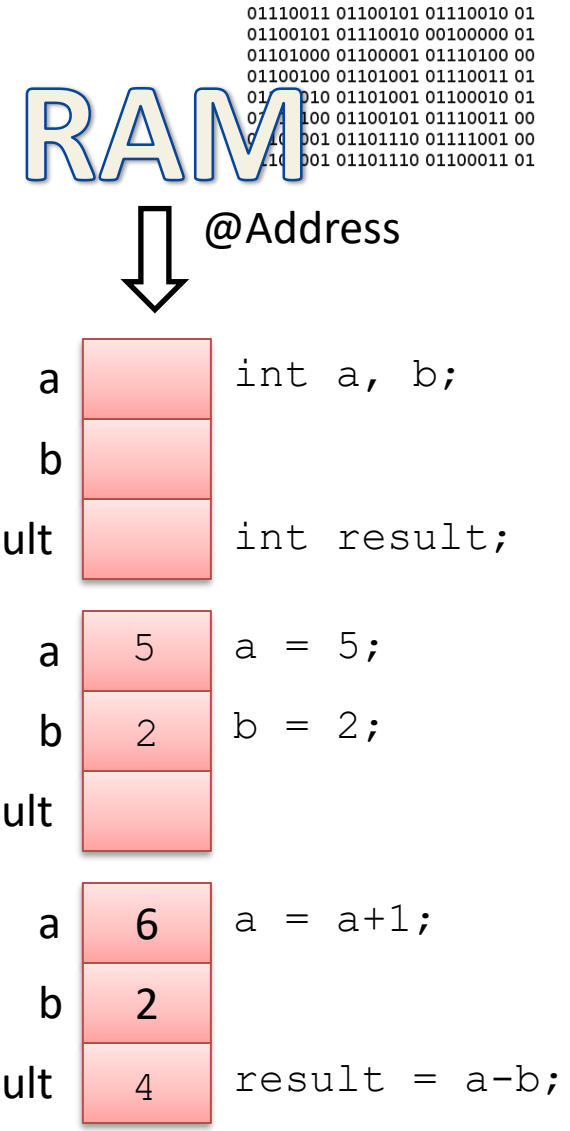
```
false (0)
true  (any values other than 0)
```

- Example of variable declaration, assignment and usage

```
bool isKak;
isKak = score < 55;
bool narak = true;
isKak&&(!narak)
```

Memory Concepts

```
1  // operating with variables
2
3  #include <iostream>
4  using namespace std;
5
6  int main ()
7  {
8    // declaring variables:
9    int a, b;
10   int result;
11
12   // process:
13   a = 5;
14   b = 2;
15   a = a + 1;
16   result = a - b;
17
18   // print out the result:
19   cout << result;
20
21   // terminate the program:
22   return 0;
23 }
```



Type Casting

(type_name) expression	C-like cast notation
type_name (expression)	Functional cast notation

```
#include <iostream>

int main() {

   int sum = 17, count = 5;
   double mean;

   mean = sum / count;
   std::cout << "Value of mean : " << mean ;

   return 0;
}
```

⇩

Value of mean : 3

```
#include <iostream>

int main() {

   int sum = 17, count = 5;
   double mean;

   mean = (double) sum / count;
   std::cout << "Value of mean : " << mean ;

   return 0;
}
```

⇩

Value of mean : 3.4

Constant Expression

```
const type_name variable_name = value;
```

– Use const type qualifier **const** to defines that the data is constant (is not modifiable).

```
1  #include <iostream>
2  #include <iomanip>
3  using namespace std;
4
5  int main() {
6      const float PI = 3.1416;
7      const char nl = '\n';
8
9      cout << setw(16) << "r = 1: Area = " << PI*1*1 << nl;
10     cout << setw(16) << "r = 1.5: Area = " << PI*1.5*1.5 << nl;
11     cout << setw(16) << "r = 2: Area = " << PI*2*2 << nl;
12     cout << setw(16) << "r = 2.5: Area = " << PI*2.5*2.5 << nl;
13
14     return 0;
15 }
```

```
  r = 1: Area = 3.1416
r = 1.5: Area = 7.0686
  r = 2: Area = 12.5664
r = 2.5: Area = 19.635
```

Arithmetic & Compound Assignment Operators

- **Arithmetic Operators**

  (+, -, *, /, %)

operator	description
+	addition
-	subtraction
*	multiplication
/	division
%	modulo

- **Compound Assignment**

  (+=, -=, *=, /=, %=, >>=, <<=, &=, ^=, |=)

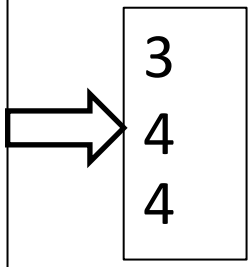
expression	equivalent to...
`y += x;`	`y = y + x;`
`x -= 5;`	`x = x - 5;`
`x /= y;`	`x = x / y;`
`price *= units + 1;`	`price = price * (units+1);`

Increment & Decrement Operators

- Increment operator (++) increases the value stored in a variable by one (equivalent to +=1)
- Decrement operator (--) decreases the value stored in a variable by one (equivalent to -=1)
- ++x; and x+=1; and x=x+1; are equivalent expressions
- Can be used both as a prefix (++x) and as a suffix (x++)
  - prefix (++x) = the expression evaluates to the final value of x (already increased)
  - suffix (x++) = the value is also increased, but the expression evaluates to the value that x had before being increased.

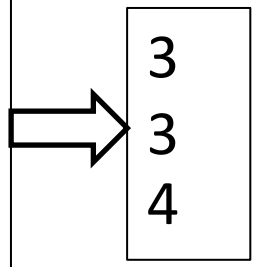
Increment & Decrement Operators

```
int x = 3;
cout << x <<'\n';
cout << ++x <<'\n';
cout << x;
```



(x increases before cout)

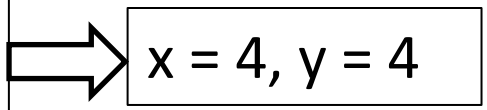
```
int x = 3;
cout << x <<'\n';
cout << x++ <<'\n';
cout << x;
```



(x increases after cout)

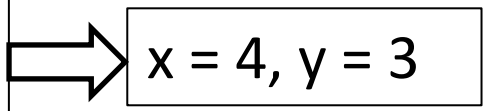
```
int y,x = 3;
y = ++x;
cout << "x = " << x << ", y = " << y;
```

(x increases before assign value to y)




```
int y,x = 3;
y = x++;
cout << "x = " << x << ", y = " << y;
```

(x increases after assign original value to y)



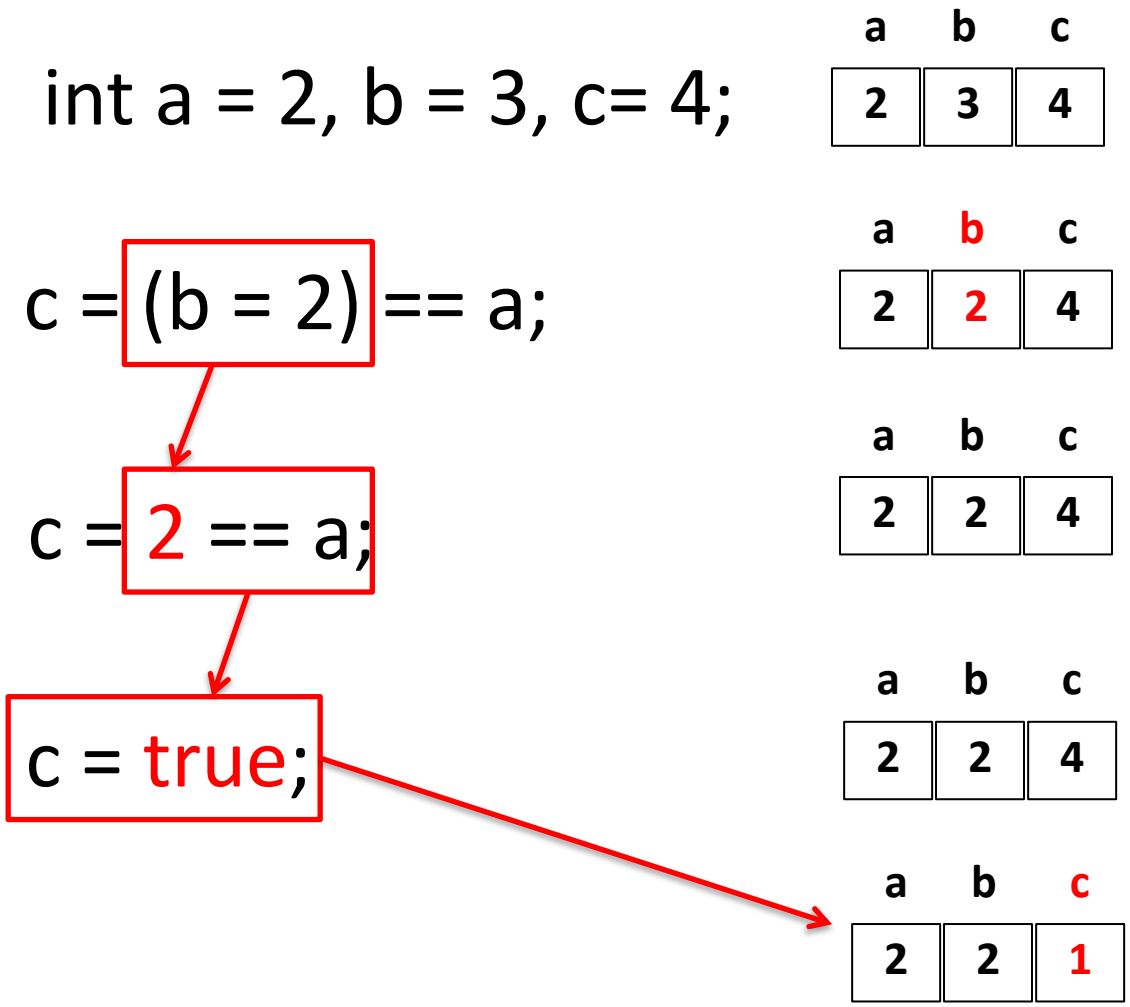
Logical Operators

- The result of rational and equality operation is either true or false
  - Value of 0 is consider as false
  - Any values other than 0 is considered as true
- The operator ! is the Boolean operation NOT
- The operator && corresponds to the operation AND
- The operator || corresponds to the operation OR

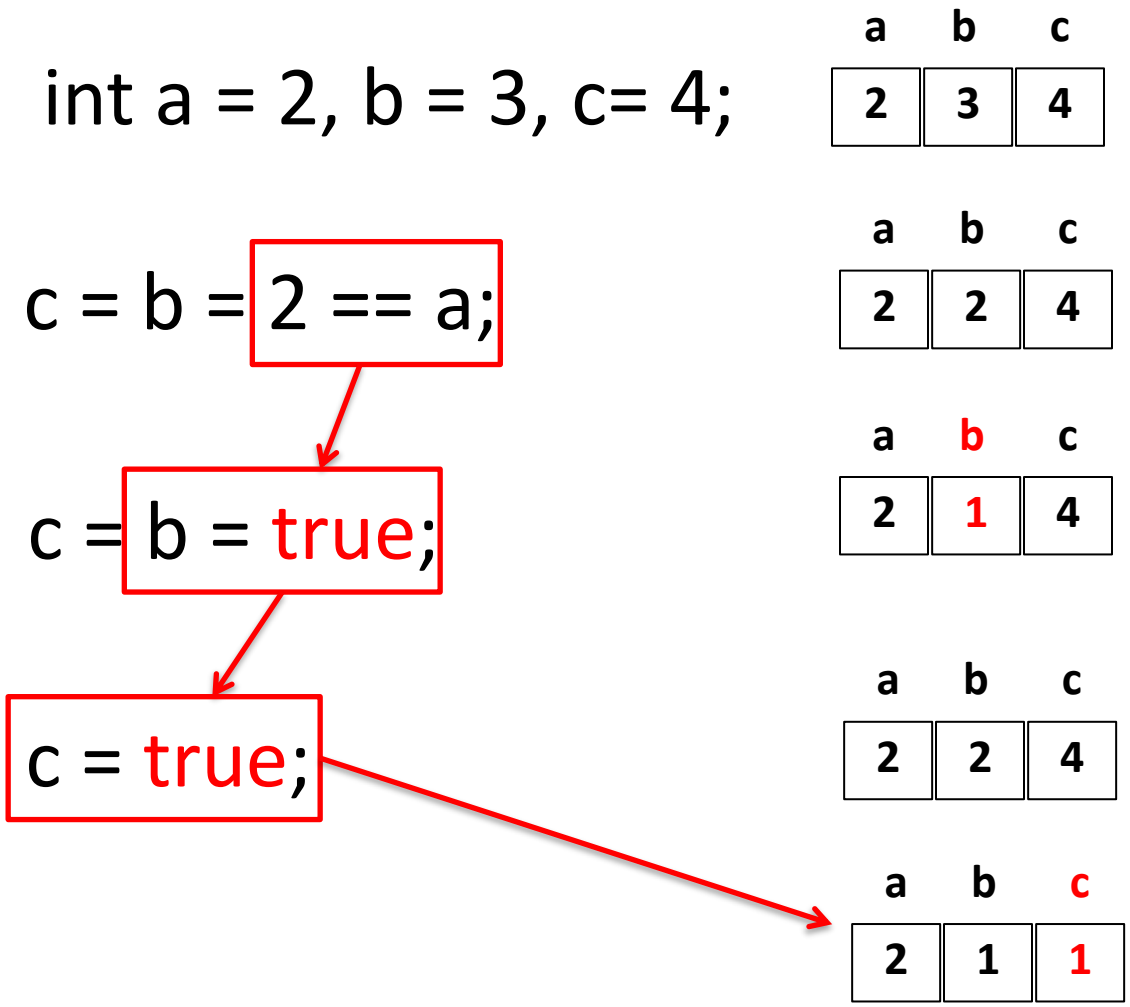
&& OPERATOR (and)		
a	b	a && b
true	true	true
true	false	false
false	true	false
false	false	false

|| OPERATOR (or)		
a	b	a || b
true	true	true
true	false	true
false	true	true
false	false	false

Precedence of Operators



Precedence of Operators



String (std::string)

- Series of characters treated as single unit
- Can include letters, digits, special characters **+**, **−**, ***** ...
- String literal enclosed in **" "** (double quotes) , for example:

```
"I like C++"
```

- #include <string> in preprocessor

```
#include <iostream>
#include <string>
using namespace std;

int main ()
{
  string mystring;
  mystring = "This is the initial string content";
  cout << mystring << endl;
  mystring = "Now... It's different string\n";
  cout << mystring;
  mystring = "How to display '\\' and '\"' ";
  cout << mystring;
  return 0;
}
```

```
This is the initial string content
Now... It's different string
How to display '\' and '"'
```

String (std::string)

```
cin >> string_variable;
```

```
getline(cin, string_variable);
```

```
#include<iostream>
#include<string>

using namespace std;

int main(){
    string name;
    cout << "Enter your name: ";
    cin >> name;
    cout << "Your name is " << name;

    return 0;
}
```

Output

```
Enter your name: Shinnosuke Nohara
Your name is Shinnosuke
```

```
#include<iostream>
#include<string>

using namespace std;

int main(){
    string name;
    cout << "Enter your name: ";
    getline(cin,name);
    cout << "Your name is " << name;

    return 0;
}
```

Output

```
Enter your name: Shinnosuke Nohara
Your name is Shinnosuke Nohara
```

Example 8-A: Chat with Fahsai

```
getline(cin, string_variable);
```

```
cin.ignore();
```

```
#include <iostream>
#include <string>
using namespace std;

int main()
{
    int year;
    cout << "Fahsai: Sawadee ka...\n";
    cout << "Fahsai: I want to know what year were you born in?\n";
    cout << "Me: ";
    cin >> year;
    cin.ignore();
    cout << "Fahsai: Wow!!! You're so young.\n";
    cout << "Fahsai: You're just only " << 2021-year << " years old.\n" ;

    string ans;
    cout << "Fahsai: Do you like to watch movies?\n";
    cout << "Me: ";
    getline(cin,ans);
    cout << "Fahsai: What is your favorite movie?\n";
    cout << "Me: ";
    getline(cin,ans);
    cout << "Fahsai: I think so. " << ans << " was a really good movie!!!" ;

    return 0;
}
```

cin >> leaves the '\n' character in the input stream
When switch from >> operator to getline() use cin.ignore() to discard '\n'

Extracts characters (including space and tab) from the stream until '\n' found

Example 8-A: Chat with Fahsai

Fahsai: Sawadee ka...
Fahsai: I want to know what year were you born in?
Me: 1967
Fahsai: Wow!!! You're so young.
Fahsai: You're just only 55 years old.
Fahsai: Do you like to watch movies?
Me: Very very very much
Fahsai: What is your favorite movie?
Me: Porn Jak Fah
Fahsai: I think so. Porn Jak Fah was a really good movie!!!

รอด!!!

Comments

- Document programs
- Improve program readability
- Ignored by compiler
- Single-line comment
  - Begin with //
- Multiple-line comment
  - Begin with /*
  - End with */

```
1  #include<iostream>
2  using namespace std;
3
4  /* This is a simple program for adding two numbers.
5  This example shows how to write comments in your code.
6  These lines will be ignored by complier.
7  */
8
9  int main(){
10     int a, b; //Declare two variables.
11
12     //Get inputs from user.
13     cout << "Enter two numbers: ";
14     cin >> a >> b;
15
16     //Show result.
17     cout << "Result = " << a+b;
18
19     return 0;
20 }
21 //Finished....
22 //Good bye!!!
```

output

```
Enter two numbers: 56 13
Result = 69
```

if Compound Statement

```cpp
#include <iostream>
using namespace std;

int main ()
{
    if(true)
    cout << "Print 1\n";
    cout << "Print 2\n";
    cout << "Print 3\n";
    cout << "Print 4\n";
    cout << "Print 5\n";

    return 0;
}
```

```
Print 1
Print 2
Print 3
Print 4
Print 5
```

```cpp
#include <iostream>
using namespace std;

int main ()
{
    if(false)
    cout << "Print 1\n";
    cout << "Print 2\n";
    cout << "Print 3\n";
    cout << "Print 4\n";
    cout << "Print 5\n";

    return 0;
}
```

```
Print 2
Print 3
Print 4
Print 5
```

```cpp
#include <iostream>
using namespace std;

int main ()
{
    if(true){
    cout << "Print 1\n";
    cout << "Print 2\n";
    cout << "Print 3\n";
    }
    cout << "Print 4\n";
    cout << "Print 5\n";

    return 0;
}
```

```
Print 1
Print 2
Print 3
Print 4
Print 5
```

```cpp
#include <iostream>
using namespace std;

int main ()
{
    if(false){
    cout << "Print 1\n";
    cout << "Print 2\n";
    cout << "Print 3\n";
    }
    cout << "Print 4\n";
    cout << "Print 5\n";

    return 0;
}
```

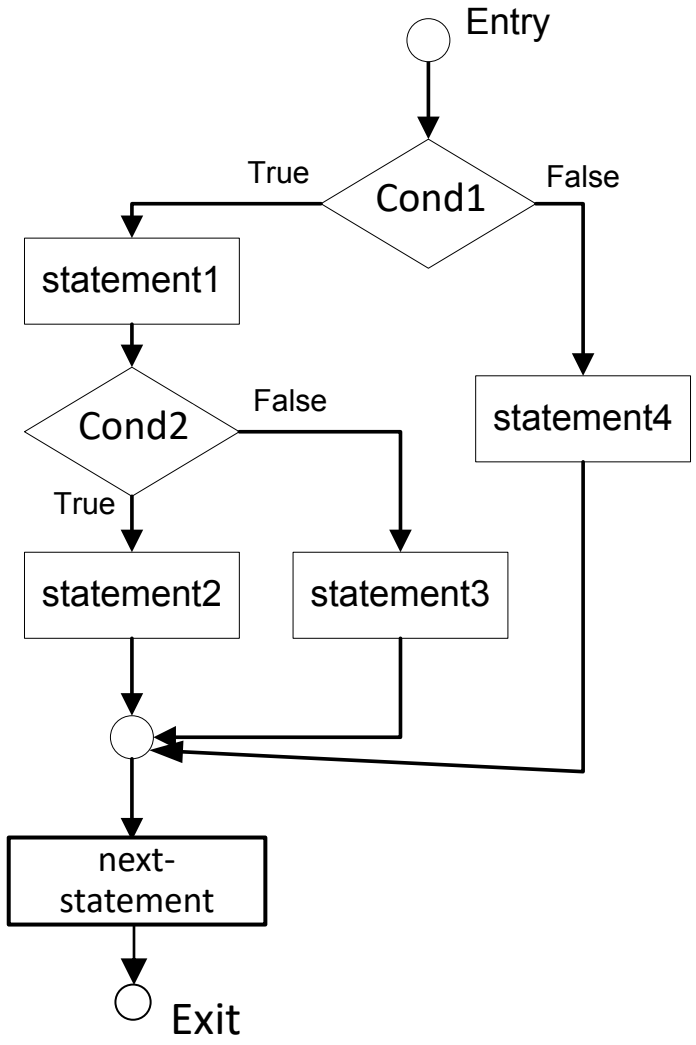
```
Print 4
Print 5
```

Nested if-else Structure

- Use one if-else statement inside another if-else statement.

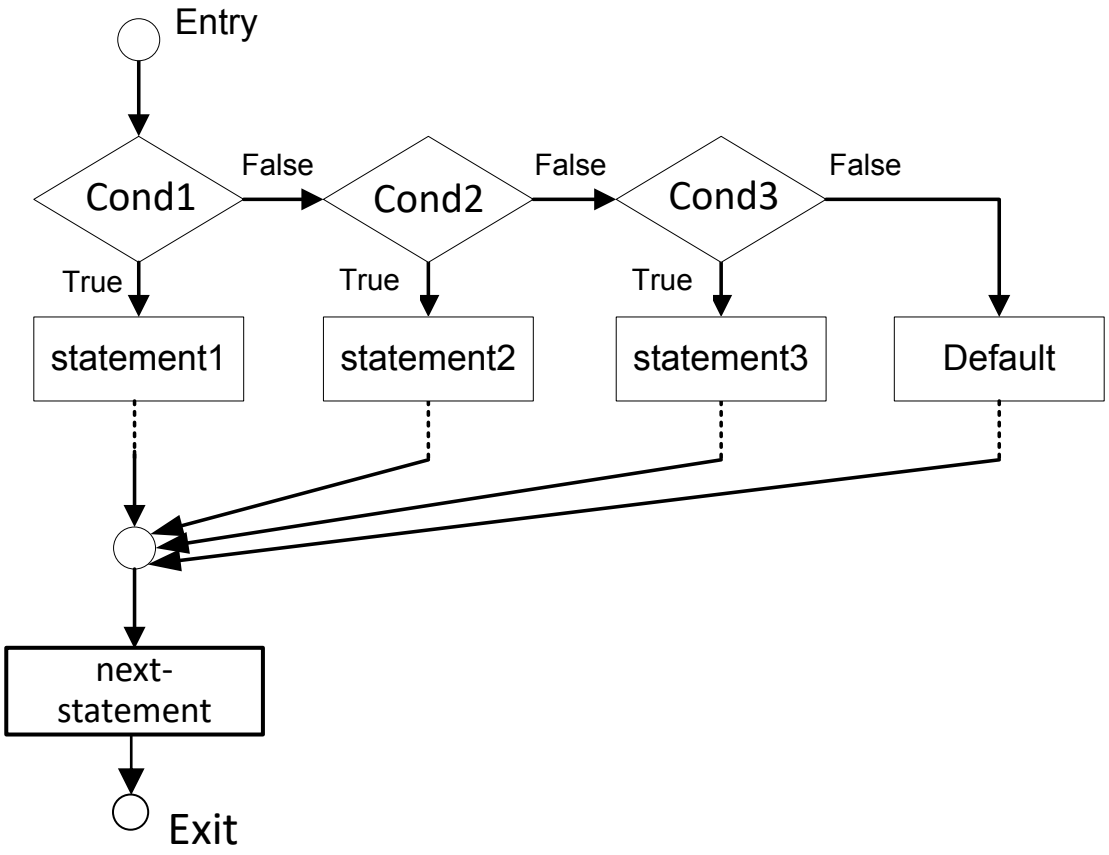
```
if (condition1){
   statement1;
   if (condition 2)
      statement2;
   else
      statement3;
}
else
   statement4;
next-statement;
```

Nested if-else Structure



if-else if Statement

```
if (condition1){
   statement1;
   ……
}
else if(condition2){
   statement2;
   ……
}
else if (condition3){
   statement3;
   ……
}
else{
   default;
   ……
}
next-statement;
```



Example 9-C: Personal Income Tax

```
1  #include <iostream>
2  using namespace std;
3
4  int main() {
5      int income,tax;
6
7      cout << "Please input your taxable income: ";
8      cin >> income;
9
10     if(income <= 150000)        tax = 0;
11     else if(income <= 300000)   tax = (income-150000)*0.05;
12     else if(income <= 500000)   tax = (income-300000)*0.1+7500;
13     else if(income <= 750000)   tax = (income-500000)*0.15+27500;
14     else if(income <= 1000000)  tax = (income-750000)*0.2+65000;
15     else if(income <= 2000000)  tax = (income-1000000)*0.25+115000;
16     else if(income <= 4000000)  tax = (income-2000000)*0.30+365000;
17     else                        tax = (income-4000000)*0.35+965000;
18     cout << "You have to pay " << tax << " baht.";
19
20     return 0;
21 }
```

switch Statement

- Test variable for multiple values
- Series of `case` labels and optional `default` case

```
switch ( variable ) {
 case value1:
   statement1;
   statement2;
   ……
   break;

 case value2:
 case value3:
   statement3;
   statement4;
   ……
   break;

 default:
  statement5;
  statement6;
  ……
   break;
}
```

Executed if **`variable`** matches no other cases (no **`break`** is executed before reaching here)

Actually, **`break`** is not necessary to put it in the last case

switch Statement

```
1  #include <iostream>
2  using namespace std;
3
4  int main() {
5      int x;
6      cout << "Input Number: ";
7      cin >> x;
8
9      switch(x){
10         case 1:
11             cout << "Print 1\n";
12             cout << "Print 2\n";
13             cout << "Print 3\n";
14             break;
15         case 2:
16         case 3:
17             cout << "Print 4\n";
18             cout << "Print 5\n";
19             break;
20         case 4:
21             cout << "Print 6\n";
22             break;
23         default:
24             cout << "Print 7\n";
25             cout << "Print 8\n";
26     }
27     return 0;
28 }
```

Input Number: 1
Print 1
Print 2
Print 3

Input Number: 2
Print 4
Print 5

Input Number: 3
Print 4
Print 5

Input Number: 4
Print 6

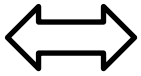
Input Number: 5
Print 7
Print 8

Input Number: 0
Print 7
Print 8

switch Statement

```cpp
#include <iostream>
using namespace std;

int main() {
    int x;
    cout << "Input Number: ";
    cin >> x;

    switch(x){
        case 1:
            cout << "Print 1\n";
            cout << "Print 2\n";
            cout << "Print 3\n";
            break;
        case 2:
        case 3:
            cout << "Print 4\n";
            cout << "Print 5\n";
            break;
        case 4:
            cout << "Print 6\n";
            break;
        default:
            cout << "Print 7\n";
            cout << "Print 8\n";
    }
    return 0;
}
```

Equivalent



```cpp
#include <iostream>
using namespace std;

int main() {
    int x;
    cout << "Input Number: ";
    cin >> x;

    if(x == 1){
        cout << "Print 1\n";
        cout << "Print 2\n";
        cout << "Print 3\n";
    }else if(x == 2 || x == 3){
       cout << "Print 4\n";
       cout << "Print 5\n";
    }else if(x == 4){
        cout << "Print 6\n";
    }else{
        cout << "Print 7\n";
        cout << "Print 8\n";
    }

    return 0;
}
```

switch Statement

```
1  #include <iostream>
2  using namespace std;
3
4  int main() {
5      int x;
6      cout << "Input Number: ";
7      cin >> x;
8
9      switch(x){
10         case 1:
11             cout << "Print 1\n";
12             cout << "Print 2\n";
13             cout << "Print 3\n";
14
15         case 2:
16         case 3:
17             cout << "Print 4\n";
18             cout << "Print 5\n";
19             break;
20         case 4:
21             cout << "Print 6\n";
22             break;
23         default:
24             cout << "Print 7\n";
25             cout << "Print 8\n";
26     }
27     return 0;
28 }
```

Input Number: 1
Print 1
Print 2
Print 3
Print 4
Print 5

Input Number: 2
Print 4
Print 5

Input Number: 3
Print 4
Print 5

Input Number: 4
Print 6

Input Number: 5
Print 7
Print 8

Input Number: 0
Print 7
Print 8

All statements under matched case are executed until **break** is executed or it is end of **switch** statement

Conditional Ternary Operator (?)

```
condition ? result1 : result2
```

If `condition` is `true`, the entire expression evaluates to `result1`,
If `condition` is `false`, the entire expression evaluates to `result2`.

```
7==5 ? 4 : 3      // evaluates to 3, since 7 is not equal to 5.
7==5+2 ? 4 : 3    // evaluates to 4, since 7 is equal to 5+2.
5>3 ? a : b       // evaluates to the value of a, since 5 is greater than 3.
a>b ? a : b       // evaluates to whichever is greater, a or b.
```

http://www.cplusplus.com/doc/tutorial/operators/

Conditional Ternary Operator (?)

```cpp
#include <iostream>
using namespace std;

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

  a=2;
  b=7;
  c = (a>5) ? a : b;

  cout << "c = " << c << '\n';
}
```

```cpp
#include <iostream>
using namespace std;

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

  a=2;
  b=7;
  if(a > 5) c = a;
  else c = b;

  cout << "c = " << c << '\n';
}
```

```
c = 7
```

http://www.cplusplus.com/doc/tutorial/operators/

Conditional Ternary Operator (?)

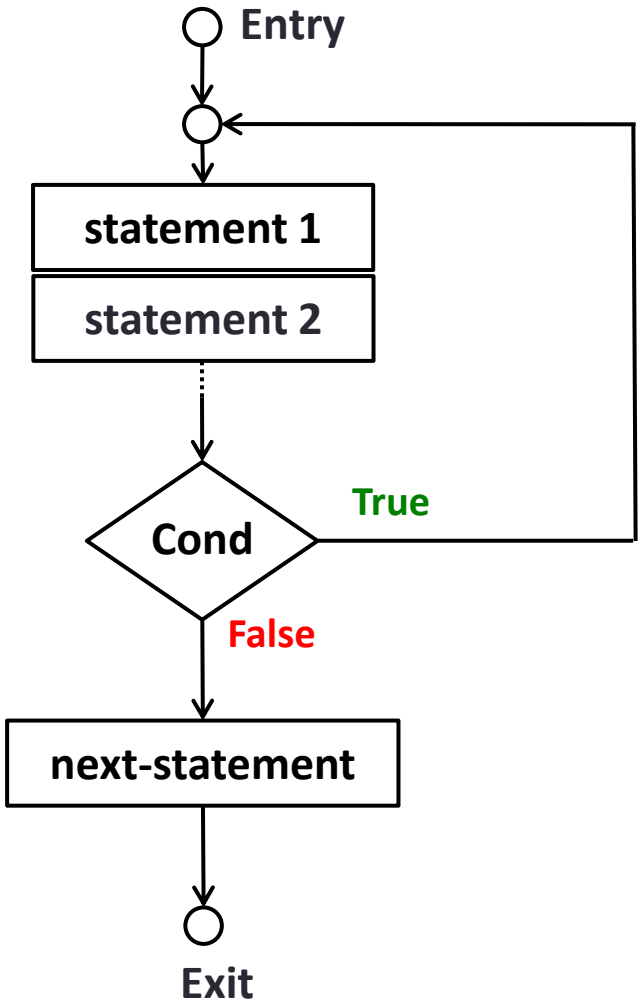
```cpp
#include <iostream>
using namespace std;

int main ()
{
  int s = 45;
  cout << (s>=80 ? 'A' : s>=70 ? 'B' : s>=60 ? 'C' : s>=50 ? 'D' : 'F');
}
```

```cpp
#include <iostream>
using namespace std;

int main ()
{
  int s = 45;
  if(s>=80) cout << 'A';
  else if(s>=70) cout << 'B';
  else if(s>=60) cout << 'C';
  else if(s>=50) cout << 'D';
  else cout << 'F';
}
```

do-while Statement

```
do {
   statement1;
   statement2;
   ...
} while (condition);
next-statement;
```

Don't forget



- Execute `statement1, statement2, …` one time before checking `condition`

Example 10-A: Nuclear Launch Code

```
Please input the nuclear launch code:i love u
Please input the nuclear launch code:123456789
Please input the nuclear launch code:6969
Nuclear weapons are launched!!!
บะคู้มมมมม...
```

```
1  #include <iostream>
2  #include <string>
3  using namespace std;
4
5  int main()
6  {
7      string input, password = "6969";
8      cout << "Please input the nuclear launch code:";
9      getline(cin,input);
10     while(input != password){
11         cout << "Please input the nuclear launch code:";
12         getline(cin,input);
13     }
14     cout << "\a\a\aNuclear weapons are launched!!!\n";
15     cout << "บะคู้มมมมม...";
16     return 0;
17 }
```

Example 10-A: Nuclear Launch Code

```
Please input the nuclear launch code:i love u
Please input the nuclear launch code:123456789
Please input the nuclear launch code:6969
Nuclear weapons are launched!!!
บะคู้มมมมม...
```

```
1  #include <iostream>
2  #include <string>
3  using namespace std;
4
5  int main()
6  {
7      string input, password = "6969";
8      do{
9          cout << "Please input the nuclear launch code:";
10         getline(cin,input);
11     }while(input != password);
12     cout << "\a\a\aNuclear weapons are launched!!!\n";
13     cout << "บะคู้มมมมม...";
14     return 0;
15 }
```

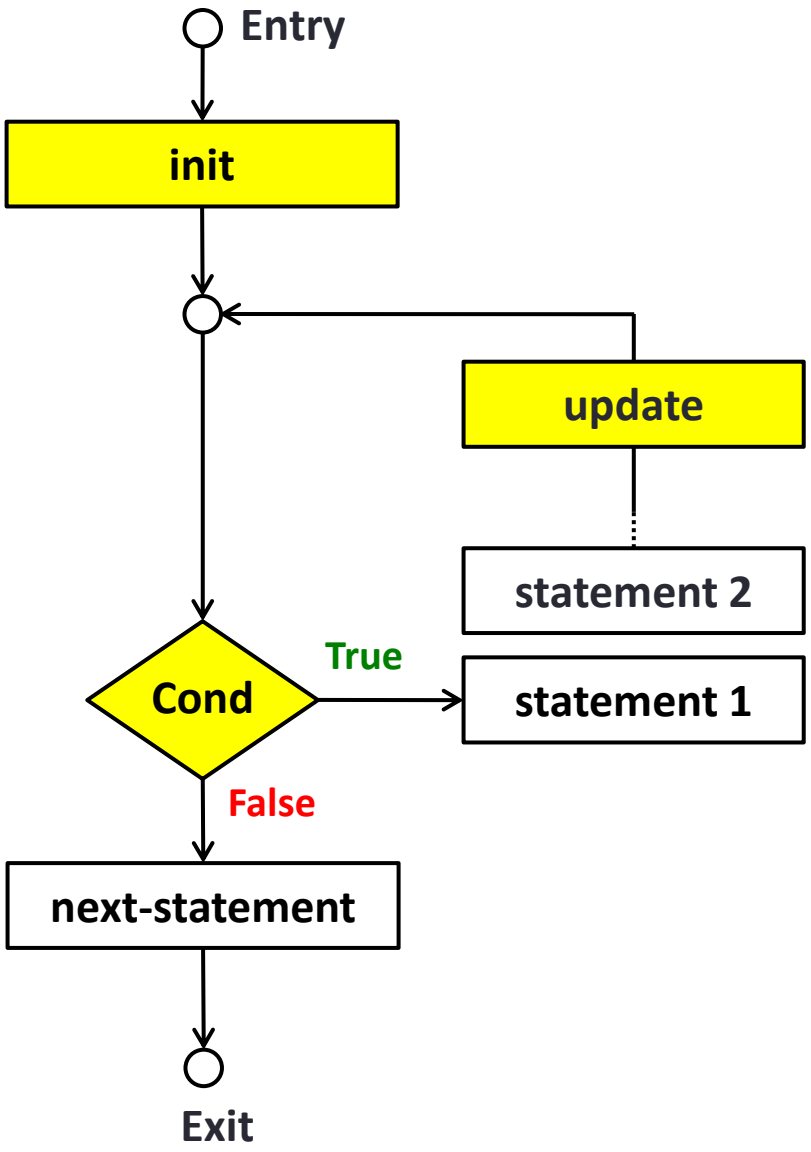
for Statement

Suitable for Counter-Controlled Repetition

```
for (init; cond; update) {
   statement1;
   statement2;
   ...
}
next-statement;
```

Equivalent

```
init;
while (cond) {
   statement1;
   statement2;
   ...
   update;
}
next-statement;
```



Example 10-B: Find Average

```
1  #include <iostream>
2  using namespace std;
3
4  int main()
5  {
6      int num, sum = 0, count = 1;
7      while(count <= 10){
8          cout << "Enter number [" << count << "]: ";
9          cin >> num;
10         sum += num;
11         count++;
12     }
13     cout << "Average = " << sum/10.0;
14     return 0;
15 }
```

Condition

Initialize

Update

Example 10-B: Find Average

```
1   #include <iostream>
2   using namespace std;
3
4   int main()
5   {
6       int num, sum = 0;
7       for(int count = 1;count <= 10;count++){
8           cout << "Enter number [" << count << "]: ";
9           cin >> num;
10          sum += num;
11      }
12      cout << "Average = " << sum/10.0;
13      return 0;
14  }
```

for Statement

```
 1  2  3  4  5
 6  7  8  9 10
11 12 13 14 15
16 17 18 19 20
21 22 23 24 25
26 27 28 29 30
31 32 33 34 35
36 37 38 39 40
41 42 43 44 45
46 47 48 49 50
51 52 53 54 55
56 57 58 59 60
61 62 63 64 65
66 67 68 69 70
71 72 73 74 75
76 77 78 79 80
81 82 83 84 85
86 87 88 89 90
91 92 93 94 95
96 97 98 99 100
```

```cpp
#include <iostream>
using namespace std;

int main()
{
    for(int i = 1;i <= 100 ;i++){
        if(i <= 9) cout << " ";
        cout << i;
        if(i%5 == 0) cout << "\n";
        else cout << " ";
    }
    return 0;
}
```

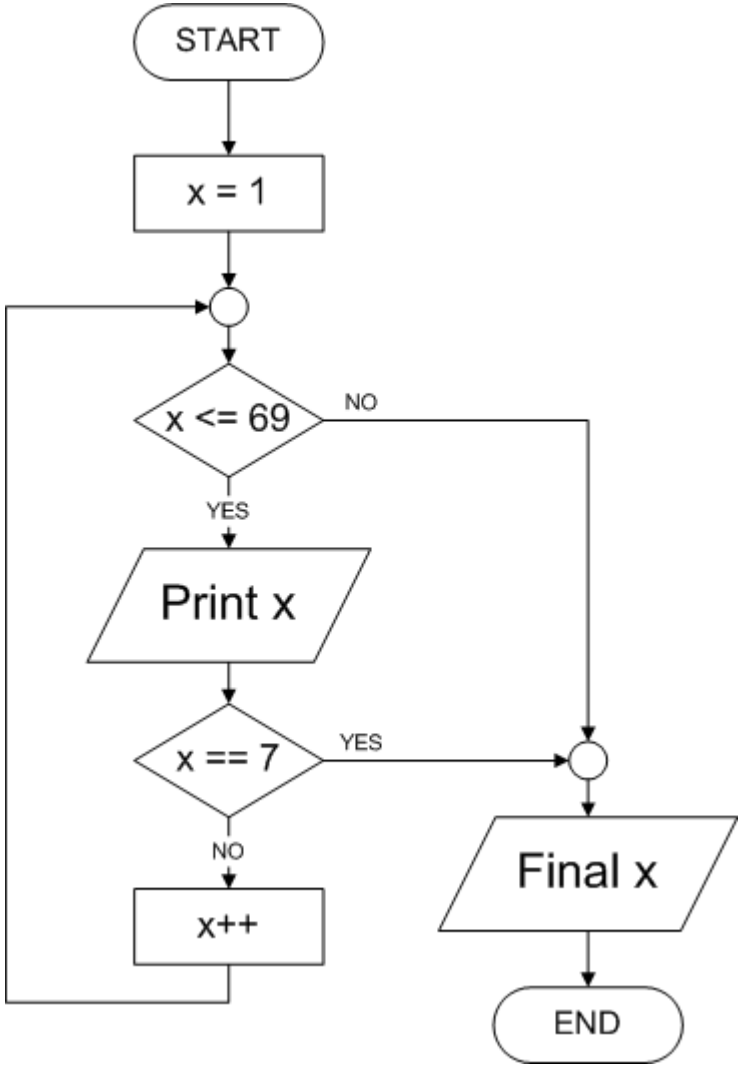
Example 10-C: Find Max

```
Enter number [1]: 2
Enter number [2]: 25
Enter number [3]: 0
Enter number [4]: 43
Enter number [5]: 7
Enter number [6]: 5
Enter number [7]: 6
Enter number [8]: 12
Enter number [9]: 24
Enter number [10]: 32
Maximum location = 4
Maximum value = 43
```

```cpp
#include <iostream>
using namespace std;

int main()
{
    int num, max, loc;
    for(int i = 1; i <= 10; i++){
        cout << "Enter number [" << i << "]: ";
        cin >> num;
        if(i == 1 || num > max){
            max = num;
            loc = i;
        }
    }
    cout << "Maximum location = " << loc << '\n';
    cout << "Maximum value = " << max;
    return 0;
}
```

break Statement

```
1  #include <iostream>
2  using namespace std;
3
4  int main()
5  {
6      int x = 1;
7      while(x <= 69){
8          cout << "Print " << x << endl;
9          if(x == 7) break;
10         x++;
11     }
12     cout << "Final x = " << x << endl;
13     return 0;
14 }
```



break Statement

```
 4  int main () {
 5
 6      while(1) {
 7
 8          ...
 9
10          if (cond)
11              break;
12
13          ...
14
15      }
16
17      return 0;
18  }
```

```
 4  int main () {
 5
 6      do {
 7
 8          ...
 9
10          if (cond)
11              break;
12
13          ...
14
15      } while(1);
16
17      return 0;
18  }
```

```
 4  int main () {
 5
 6      for (;;) {
 7
 8          ...
 9
10          if (cond)
11              break;
12
13          ...
14
15      }
16
17      return 0;
18  }
```

Infinity loop with **break** statement

break Statement

```
 5  int main()
 6  {
 7      string input, password = "6969";
 8      do{
 9          cout << "Please input the nuclear launch code:";
10          getline(cin,input);
11      }while(input != password);
12      cout << "\a\a\aNuclear weapons are launched!!!\n";
13      cout << "บะคู้มมมมม...";
14      return 0;
15  }
```

Equivalent

```
 5  int main()
 6  {
 7      string input, password = "6969";
 8      do{
 9          cout << "Please input the nuclear launch code:";
10          getline(cin,input);
11          if(input == password) break;
12      }while(true);
13      cout << "\a\a\aNuclear weapons are lauched!!!\n";
14      cout << "บะคู้มมมมม...";
15      return 0;
16  }
```

Example 10-A: Nuclear Launch Code

```
5   int main()
6   {
7       int count = 1;
8       string input, password = "6969";
9       do{
10          if(count >= 4){
11              cout << "Access denied. .... System is locked.\n";
12              break;
13          }
14          cout << "Please input the nuclear launch code:";
15          getline(cin,input);
16          if(input == password){
17              cout << "\a\a\aNuclear weapons are lauched!!!\n";
18              cout << "บะคู้มมมมม...";
19              break;
20          }
21          count++;
22      }while(true);
23
24      return 0;
```

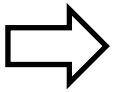
```
Please input the nuclear launch code:1
Please input the nuclear launch code:2
Please input the nuclear launch code:3
Access denied. .... System is locked.
```

```
Please input the nuclear launch code:I love you
Please input the nuclear launch code:6969
Nuclear weapons are lauched!!!
บะคู้มมมมม...
```

continue Statement

while

```
1  #include <iostream>
2  using namespace std;
3
4  int main()
5  {
6      int x = 0;
7      while(x <= 3){
8          x++;
9          cout << "Print " << x << '\n';
10         cout << "Print " << 2*x << '\n';
11         cout << "Print " << 3*x << '\n';
12         cout << "-----------------------\n";
13     }
14     return 0;
15 }
```



```
Print 1
Print 2
Print 3
-----------------------
Print 2
Print 4
Print 6
-----------------------
Print 3
Print 6
Print 9
-----------------------
Print 4
Print 8
Print 12
-----------------------
```

continue Statement

while

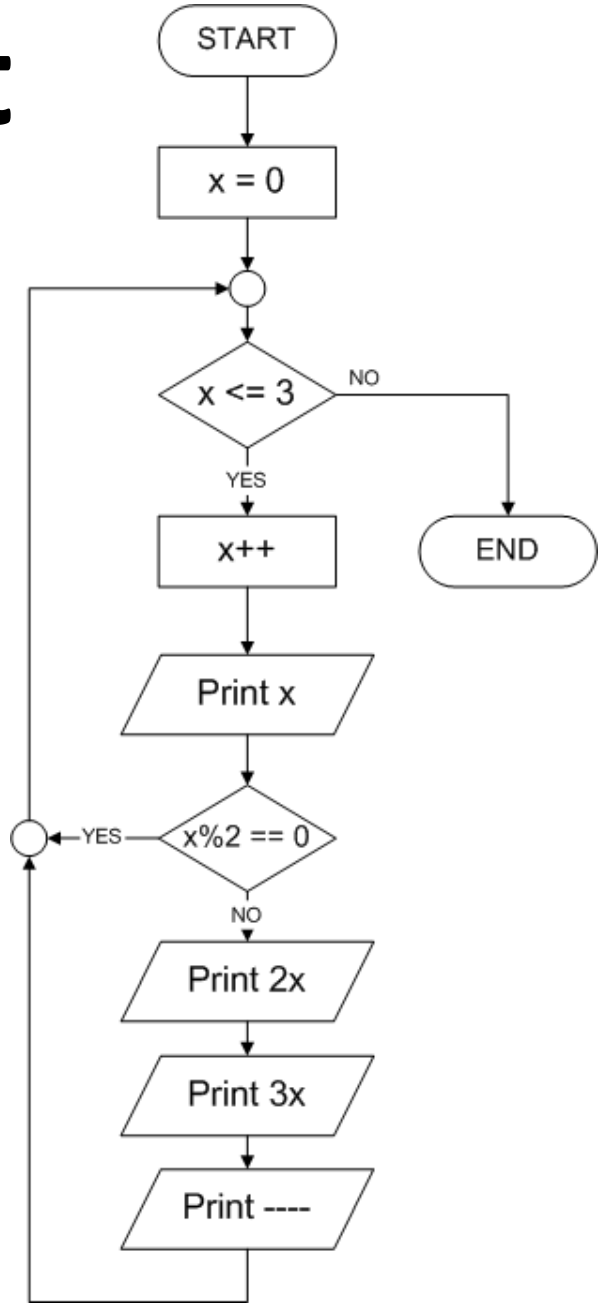
```
1  #include <iostream>
2  using namespace std;
3
4  int main()
5  {
6      int x = 0;
7      while(x <= 3){
8          x++;
9          cout << "Print " << x << '\n';
10         if(x%2 == 0) continue;
11         cout << "Print " << 2*x << '\n';
12         cout << "Print " << 3*x << '\n';
13         cout << "-----------------------\n";
14     }
15     return 0;
16 }
```

```
Print 1
Print 2
Print 3
-----------------------
Print 2
Print 3
Print 6
Print 9
-----------------------
Print 4
```

continue Statement

while

```
#include <iostream>
using namespace std;

int main()
{
    int x = 0;
    while(x <= 3){
        x++;
        cout << "Print " << x << '\n';
        if(x%2 == 0) continue;
        cout << "Print " << 2*x << '\n';
        cout << "Print " << 3*x << '\n';
        cout << "-----------------------\n";
    }
    return 0;
}
```

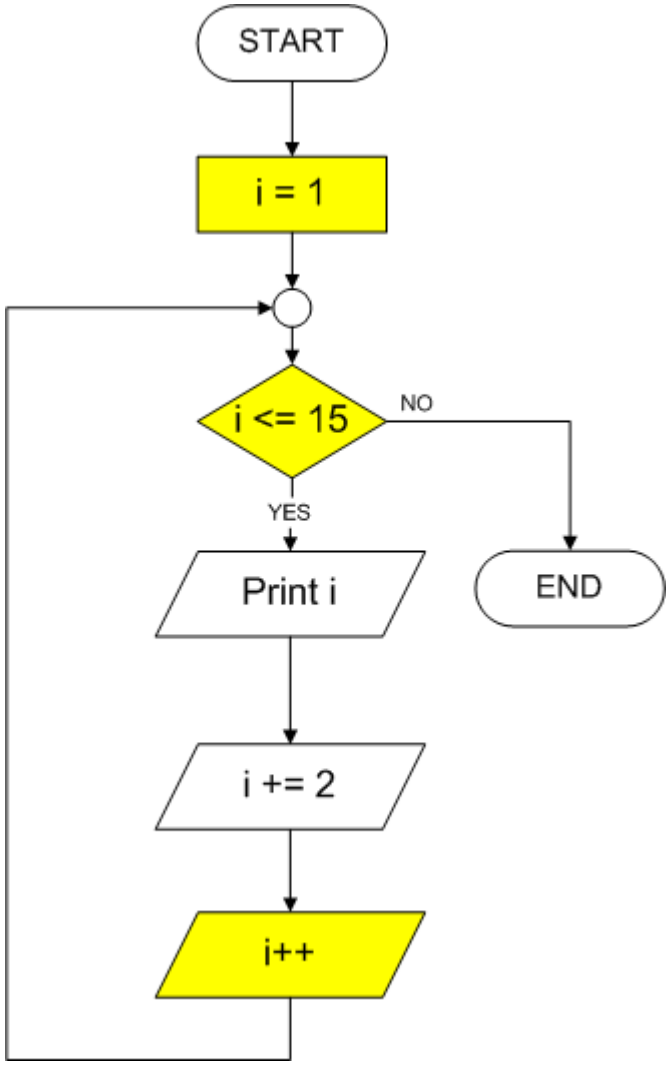


continue Statement

for

```
1   #include <iostream>
2   using namespace std;
3
4   int main()
5   {
6       for(int i = 1; i <= 15; i++){
7           cout << "Print" << i << "\n";
8           i+=2;
9       }
10      return 0;
11  }
```

```
Print1
Print4
Print7
Print10
Print13
```

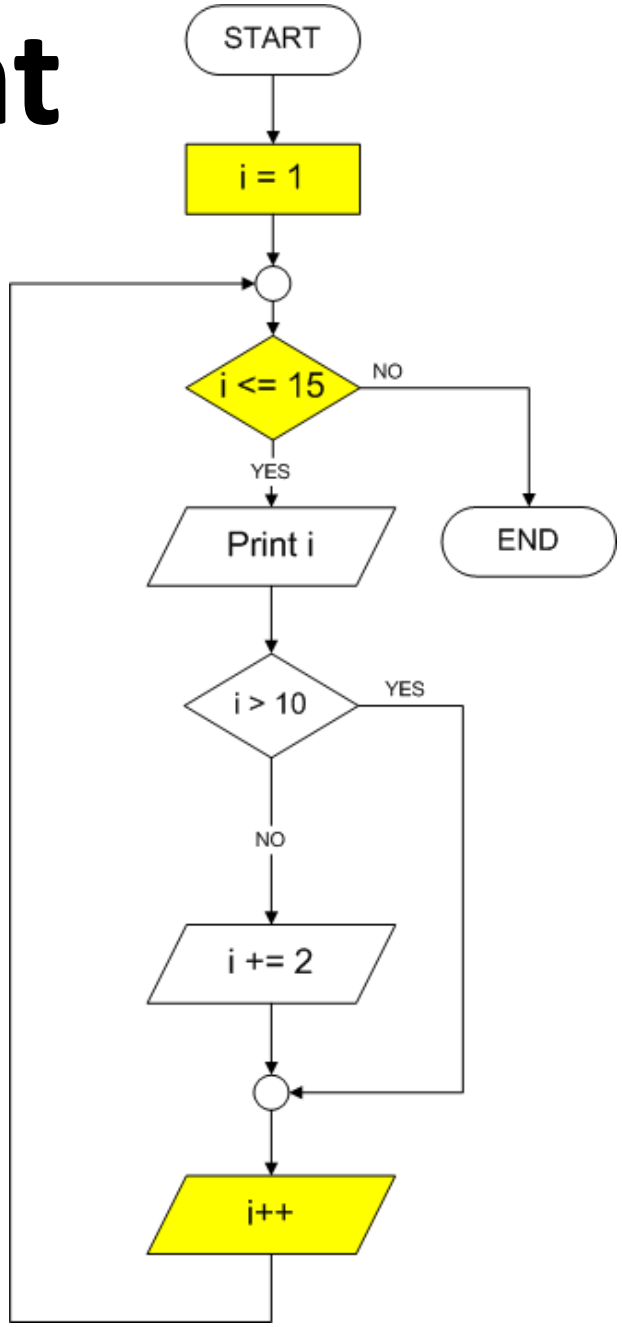


continue Statement

for

```
1  #include <iostream>
2  using namespace std;
3
4  int main()
5  {
6      for(int i = 1; i <= 15; i++){
7          cout << "Print" << i << "\n";
8          if(i > 10) continue;
9          i+=2;
10     }
11     return 0;
12 }
```

```
Print1
Print4
Print7
Print10
Print13
Print14
Print15
```



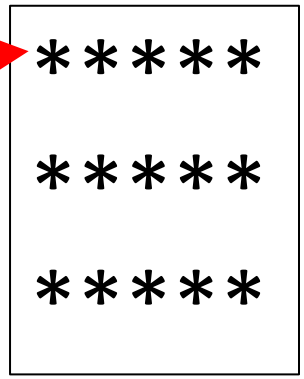
continue Statement

```cpp
#include <iostream>
using namespace std;

int main()
{
    for(int i = 1; i <= 15; i++){
        cout << "Print" << i << "\n";
        if(i > 10) continue;
        i+=2;
    }
    return 0;
}
```

```cpp
#include <iostream>
using namespace std;

int main()
{
    for(int i = 1; i <= 15; i++){
        cout << "Print" << i << "\n";
        if(i <= 10){
            i+=2;
        }
    }
    return 0;
}
```

Nested Loop Structure

```
1  #include <iostream>
2  using namespace std;
3
4  int main()
5  {
6      for(int i = 0; i < 3; i++){
7          for(int j = 0; j < 5; j++){
8              cout << '*';
9          }
10         cout << '\n';
11     }
12     return 0;
13 }
```



Nested Loop Structure

```cpp
#include <iostream>
using namespace std;

int main()
{
    for(int i = 0; i < 4; i++){
        for(int j = 0; j <= i; j++){
            cout << '*';
        }
        cout << '\n';
    }
    return 0;
}
```

```cpp
#include <iostream>
using namespace std;

int main()
{
    for(int i = 0; i < 4; i++){
        for(int j = 0; j < 4; j++){
            if(j > i) break;
            else cout << '*';
        }
        cout << '\n';
    }
    return 0;
}
```

File Stream

- `#include <fstream>`

- **ofstream**:
  Stream class to write on files
- **ifstream**:
  Stream class to read from files
- **fstream**:
  Stream class to both read and write from/to files.

ofstream (Output File Stream)

```cpp
#include <fstream>
using namespace std;

int main () {
    ofstream dest("C:\\Karn\\Temp\\Temp\\AV\\dest.txt");
    dest << "Hello 261102 to a file.\n";
    dest << "Yo!!!!.\n";
    dest << "Yo!!!! Yo!!!!.\n";
    dest.close();
    return 0;
}
```

dest.txt

Hello 261102 to a file.
Yo!!!!.
Yo!!!! Yo!!!!.

ofstream (Output File Stream)

Appending Mode
All output operations are performed at the end of the file, appending the content to the current content of the file.

```
1  #include <fstream>
2  using namespace std;
3
4  int main () {
5      ofstream dest("C:\\Karn\\Temp\\Temp\\AV\\dest.txt",ios::app);
6      dest << "Yo!!!! Yo!!!! Yo!!!!.\n";
7      dest << "Yo!!!! Yo!!!! Yo!!!! Yo!!!!.\n";
8      dest.close();
9      return 0;
10 }
```

dest.txt

Hello 261102 to a file.
Yo!!!!.
Yo!!!! Yo!!!!.
Yo!!!! Yo!!!! Yo!!!!.
Yo!!!! Yo!!!! Yo!!!! Yo!!!!.

New contents added

ifstream (Input File Stream)

```cpp
#include<iostream>
#include<fstream>
#include<string>
using namespace std;

int main(){
    ifstream source;
    source.open("C:\\Karn\\Temp\\Temp\\AV\\source.txt");
    string textline;
    getline(source, textline);
    cout << "LINE 1: " << textline << "\n";
    getline(source, textline);
    cout << "LINE 2: " << textline << "\n";

    return 0;
}
```

1. Eriku Momokuma
2. Ore Sahara
3. Loli Takisamu
4. Aisamu June
5. Sora Akai

source.txt

Line 1: 1. Eriku Momokuma
Line 2: 2. Ore Sahara

Console

ifstream (Input File Stream)

```cpp
#include<iostream>
#include<fstream>
#include<string>
using namespace std;

int main(){
    ifstream source;
    source.open("C:\\Karn\\Temp\\Temp\\AV\\source.txt");
    string textline;
    bool havetext;
    havetext = getline(source, textline);
    while(havetext){
        cout << textline << "\n";
        havetext = getline(source, textline);
    }
    cout << "---------END---------\n";
    return 0;
}
```

1. Eriku Momokuma
2. Ore Sahara
3. Loli Takisamu
4. Aisamu June
5. Sora Akai

source.txt

1. Eriku Momokuma
2. Ore Sahara
3. Loli Takisamu
4. Aisamu June
5. Sora Akai

--------END---------

Console

ifstream (Input File Stream)

```
10      bool havetext;
11      havetext = getline(source, textline);
```

- **getline** can return **bool**
  - Evaluated as **true** if the stream has no errors occurred
  - Evaluated as **false** if it failed to read

```
while(havetext)
```

- Read text line by line until it failed to read

ifstream (Input File Stream)

```
1  #include <iostream>
2  #include <fstream>
3  #include <string>
4  using namespace std;
5
6  int main () {
7      ifstream source;
8      source.open("source1.txt");
9      string textline;
10     while (getline(source,textline)){
11       cout << textline << '\n';
12     }
13     cout << "--------END of 1st file-------" << '\n';
14     source.close();
15     source.open("source2.txt");
16     while (getline(source,textline))    {
17       cout << textline << '\n';
18     }
19     cout << "--------END of 2nd file-------" << '\n';
20     return 0;
21 }
```

Hello!!! How are you?
My name is Fahsai.

source1.txt

I'm fine, thank you.
I'm your father.
May the force be with you.

source2.txt

Hello!!! How are you?
My name is Fahsai.
--------END of 1st file-------
I'm fine, thank you.
I'm your father.
May the force be with you
--------END of 2nd file-------

Console

Example 9-A: Find Average (from file)

```
1  #include <iostream>
2  #include <fstream>
3  #include <string>
4  #include <cstdlib>
5  using namespace std;
6
7  int main () {
8      int count = 0;
9      float sum = 0;
10     string textline;
11     ifstream source("mydata.txt");
12     while (getline(source,textline))
13     {
14       cout << textline << '\n';
15       sum += atof(textline.c_str());
16       count++;
17     }
18     cout << "N = "<< count << '\n';
19     cout << "Avg = "<< sum/count << '\n';
20     source.close();
21     return 0;
22 }
```

Convert String to number

mydata.txt

```
1.1
2.2
3.3
4.4
5.555
6.9
```

Console

```
1.1
2.2
3.3
4.4
5.555
6.9
N = 6
Avg = 3.90917
```

Datatype of Texts

- Character `char a = 'A';`
- Character Array `char a[] = "AAAAA";`
  (C-style string)
- String Object `string a = "AAAAA";`
  (C++ string object)

```
#include <string>
```

Convert String to Number

- `#include <cstdlib>`

- **atoi**: Convert C-string to integer
- **atol**: Convert C-string to long integer
- **atof**: Convert C-string to double

  The function first discards as many whitespace characters as necessary until the first non-whitespace character is found. Then, starting from this character, takes as many characters as possible that are valid following a syntax resembling that of floating point literals (see below), and interprets them as a numerical value. The rest of the string after the last valid character is ignored and has no effect on the behavior of this function.
- **strtod**: Convert C-string to double

  Interpreting C-string as a floating point number and returns its value as a float. The function also sets the value of *pointer* to point to the first character after the number.
- **strtof**: Convert C-string to float (C++11 Standard)

Convert String to Number

- `#include <string>`
- `C++11 Standard`

- **stoi**: Convert string to integer
- **stol**: Convert string to long integer
- **stoll**: Convert string to long long integer
- **stoul**: Convert string to unsigned long integer
- **stoull**: Convert string to unsigned long long integer
- **stof**: Convert string to float
- **stod**: Convert string to double
- **stold**: Convert string to long double

Convert String to Number

- `atoi`,`atol`,`atof`,`strtod`,`strtof`
  - need C-string (C-style character array **`char[]`**) as input not C++ string object (**`string`**).
- **`string_variable`**`.c_str()` is used to convert C++ string object to C-string (as a pointer)

```
1  #include <iostream>
2  #include <string>
3  using namespace std;
4
5  int main()
6  {
7    char ctext[] = "12345";
8    string cpptext = "12345";
9    cout << atoi(ctext)*10 << '\n';
10   cout << atoi(cpptext.c_str())*10 << '\n';
11 }
```

```
123450
123450
```

C- string (char array)

C++ string object

Convert String to Number

```cpp
#include <iostream>
#include <string>
#include <cstdlib>
using namespace std;

int main()
{
    string text = "    1.2345   6789abcd";
    cout << "atoi => " << atoi(text.c_str()) << endl;
    cout << "atof => " << atof(text.c_str()) << endl;
}
```

```
atoi => 1
atof => 1.2345
```

```cpp
string text = "ab123    1.23456789";
cout << "atoi => " << atoi(text.c_str()) << endl;
cout << "atof => " << atof(text.c_str()) << endl;
```

```
atoi => 0
atof => 0
```

```cpp
string text = "-123.69e-5";
cout << "atoi => " << atoi(text.c_str()) << endl;
cout << "atof => " << atof(text.c_str()) << endl;
```

```
atoi => -123
atof => -0.0012369
```

Convert String to Number

```cpp
#include <iostream>
#include <string>
#include <cstdlib>
using namespace std;

int main () {
    string  text  = "    12.34e5";
    cout << "stoi => " << stoi(text) <<  endl;
    cout << "stod => " << stod(text) <<  endl;
    return 0;
}
```

```
stoi => 12
stod => 1.234e+06
```

Local Variables (Function scope)

```
 1  #include <iostream>
 2  using namespace std;
 3
 4  void myFunction(){
 5      cout << "x in myFunction = " << x << "\n";
 6  }
 7
 8  int main(){
 9      int x = 14;
10      cout << "x in main = " << x << "\n";
11      myFunction();
12      return 0;
13  }
```

```
 In function 'void myFunction()':
5:37: error: 'x' was not declared in this scope
```

Local Variables (Function scope)

```
1  #include <iostream>
2  using namespace std;
3
4  int x = 14;
5
6  void myFunction(){
7      cout << "x in myFunction = " << x << "\n";
8      int x = 0;
9      cout << "x in myFunction = " << x << "\n";
10 }
11
12 int main(){
13     cout << "x in main = " << x << "\n";
14     int x = 55;
15     cout << "x in main = " << x << "\n";
16     x = x*10;
17     cout << "x in main = " << x << "\n";
18     myFunction();
19     myFunction();
20     return 0;
21 }
```

```
x in main = 14
x in main = 55
x in main = 550
x in myFunction = 14
x in myFunction = 0
x in myFunction = 14
```

Local Variables (Block scope)

```
1  #include <iostream>
2  using namespace std;
3
4  int main(){
5      int i;
6      for(i = 1;i <= 3;i++){
7          cout << i << " ";
8      }
9      cout << "\noutside loop i = " << i;
10     return 0;
11 }
```

Function Scope

```
1 2 3
outside loop i = 4
```

```
1  #include <iostream>
2  using namespace std;
3
4  int main(){
5      for(int i = 1;i <= 3;i++){
6          cout << i << " ";
7      }
8      cout << "\noutside loop i = " << i;
9      return 0;
10 }
```

Block Scope

```
error: 'i' was not declared in this scope
```

Unitary Scope Resolution Operator

```cpp
#include <iostream>
using namespace std;

int x = 14;

int main(){
    cout << "Global x = " << x << "\n";
    float x = 14.15 ;
    cout << "Local x = " << x << "\n";
    cout << "Global x = " << ::x << "\n";
    x = 0;
    ::x++;
    cout << "Local x = " << x << "\n";
    cout << "Global x = " << ::x << "\n";

    return 0;
}
```

```
Global x = 14
Local x = 14.15
Global x = 14
Local x = 0
Global x = 15
```

Function Call by Value

```
1  #include <iostream>
2  using namespace std;
3
4  int square(int x){
5      return x*x;
6  }
7
8  int main()
9  {
10   int num = 5;
11   cout << "Input argument = " << num << "\n";
12   cout << "Return value = " << square(num) << "\n";
13   cout << "Input argument = " << num << "\n";
14 }
```

```
Input argument = 5
Return value = 25
Input argument = 5
```

In this example, `x` and `num` are different variables. Value of `x` is initialized by copying value of `num`. When `x` is modified, `num` is not modified.)

Function Call by Reference

```
1  #include <iostream>
2  using namespace std;
3
4  void square(int &x){
5      x = x*x;
6  }
7
8  int main()
9  {
10   int num = 5;
11   cout << "Input argument = " << num << "\n";
12   square(num);
13   cout << "Input argument = " << num << "\n";
14 }
```

```
Input argument = 5
Input argument = 25
```

In this example, `x` and `num` are the same variable. `x` becomes another name of `num`. When `x` is modified, `num` is also modified.)

Random Number Generation

- **`rand`** function (**`<cstdlib>`**)
  - **`i = rand();`**
  - Generates unsigned integer between 0 and RAND_MAX (usually 32767)
- Scaling and shifting

  Number **=** *shiftingValue* **+** **`rand()`** **%** *scalingFactor*

  - shiftingValue = first number in desired range
  - scalingFactor = width of desired range
  - Example

```
i = rand() % 6 + 1;
```

    - "**`Rand() % 6`**" generates a number between **0** and **5** (scaling)
    - "**`+ 1`**" makes the range 1 to 6 (shift)

Random Number Generation

- Calling rand() repeatedly
  - Gives the same sequence of numbers
- Pseudorandom numbers
  - Preset sequence of "random" numbers
  - Same sequence generated whenever program run
- To get different random sequences
  - Provide a seed value
    - Like a random starting point in the sequence
    - The same seed will give the same sequence
  - `srand(seed);`
    - `<cstdlib>`
    - Used before `rand()` to set the seed

Random Number Generation

- Can use the current time to set the seed
  - No need to explicitly set seed every time
  - `srand( time( 0 ) );`
  - `time( 0 );`
    - `<ctime>`
    - Returns current time in seconds

Example 12-B: Who you are?

```cpp
#include <iostream>
#include <cstdlib>
#include <ctime>
using namespace std;

int main()
{
    cout << "ชื่อเกาหลีของคุณคืออะไร?";
    cin.get();
    srand(time(0));
    int x = rand()%5;
    if(x == 0) cout << "ชวน มี พุง";
    else if(x == 1) cout << "วอน ติน";
    else if(x == 2) cout << "ซอย เตียง ซุด";
    else if(x == 3) cout << "กู ยัง ซิง";
    else cout << "กัง เกง ลิง";

    return 0;
}
```

Output = ?

Example 12-C: Dice Game

```
#include<iostream>
#include<cstdlib>
#include<ctime>
using namespace std;

int dice1,dice2,status;
```

Global variables

```
void rollDice(){
    dice1 = rand()%6 + 1;
    dice2 = rand()%6 + 1;
}
```

```
int gameJudge(int pl,int op){
    if(pl >= 2*op) return 1;
    else if (op >= 2*pl) return -1;
    else return 0;

}
```

```
void printResult(){
    switch(status){
        case 0:
            cout << "DRAW...\n";
            break;
        case 1:
            cout << "YOU WIN!!!!\n";
            break;
        case -1:
            cout << "YOU LOSE!!!!\n";
            break;
    }
}
```

```
int main() {
    int playerPoint,oppPoint;
    srand(time(0));
    do{
        cout << "----------------------------------------\n";
        rollDice();
        oppPoint = dice1+dice2;
        cout << "Computer Rolled: " << dice1 << "+" << dice2;
        cout << " = " << oppPoint << "\n";
        cout << "<Player turn, please enter to roll dices.>";
        cin.get();
        rollDice();
        playerPoint = dice1+dice2;
        cout << "Player Rolled: " << dice1 << "+" << dice2;
        cout << " = " << playerPoint << "\n";
        status = gameJudge(playerPoint,oppPoint);
        printResult();
    }while(status == 0);

    return 0;
}
```

Wait for a user to press an enter key.

output

```
----------------------------------------
Computer Rolled: 4+2 = 6
<Player turn, please enter to roll dices.>
Player Rolled: 1+3 = 4
DRAW...
----------------------------------------
Computer Rolled: 1+3 = 4
<Player turn, please enter to roll dices.>
Player Rolled: 4+5 = 9
YOU WIN!!!!
```

Default Arguments

Function Call by Reference

```
1  #include <iostream>
2  using namespace std;
3
4  int c = 2;
5  void square(int &x = c){
6      x = x*x;
7  }
8
9  int main(){
10   cout << "Default argument c = " << c << "\n";
11   square();
12   cout << "Default argument c = " << c << "\n";
13   int y = 5;
14   square(y);
15   cout << "Default argument c = " << c << "\n";
16   cout << "Input argument y = " << y << "\n";
17 }
```

⇦ Set default reference parameter to variable **c**

```
Default argument c = 2
Default argument c = 4
Default argument c = 4
Input argument y = 25
```

Function Prototype

return-value-type *function-name* (*parameter-type-list*) ;

- Tells compiler argument type and return type of function
- Explained in more detail (function definition) later
- Only needed if function definition after function call
- Function signature = Part of prototype with name and parameters

```
int myFunction(int,int,int);
```

Function signature

Function Prototype

- Prototype must match function definition
  - Function prototype

```
double maximum( double, double, double );
```

  - Definition

```
double maximum( double x, double y, double z )
{
  …
}
```

Function Prototype

```
1  #include <iostream>
2  using namespace std;
3
4  void printPrint(char,int,int);
5
6  int main()
7  {
8      printPrint('#',2,2);
9      printPrint('-',2,9);
10     printPrint('#',2,2);
11 }
12
13 void printPrint(char x,int y,int z){
14     for(int i = 1;i<=y;i++){
15         for(int j = 1;j<=z;j++){
16             cout << x;
17         }
18         cout << '\n';
19     }
20 }
```

```
4  void printPrint(char x,int y,int z);
4  void printPrint(char a,int b,int c);
4  void printPrint(char a,int b=1,int c=1);
```

Parameters' name can be different from function definition

- Default value must be set in the prototype before function call with omitted parameters

Function Prototype

Function Call by Reference

```
1  #include <iostream>
2  using namespace std;
3
4  void square(int &);
5
6  int main()
7  {
8    int num = 5;
9    cout << "Input argument = " << num << "\n";
10   square(num);
11   cout << "Input argument = " << num << "\n";
12 }
13
14 void square(int &x){
15     x = x*x;
16 }
```

Function Prototype

Default Arguments

```
1  #include <iostream>
2  using namespace std;
3
4  int square(int x = 5);
5
6  int main(){
7      cout << square();
8  }
9
10 int square(int x){
11     return x*x;
12 }
```

OK

```
1  #include <iostream>
2  using namespace std;
3
4  int square(int);
5
6  int main(){
7      cout << square();
8  }
9
10 int square(int x = 5){
11     return x*x;
12 }
```

X

Function Overloading

- Function overloading
  - Functions with same name and different parameters
  - Should perform similar tasks
    - i.e., function to square **int**s and function to square **float**s

```
int square(int x) {return x * x;}
float square(float x) { return x * x; }
```

- Overloaded functions distinguished by signature
  - Based on name and parameter types (order matters)
  - Name mangling
    - Encodes function identifier with parameters
  - Type-safe linkage
    - Ensures proper overloaded function called

Function Overloading

```
1  #include <iostream>
2  using namespace std;
3
4  int square(int x){
5      cout << "int version called\n";
6      return x*x;
7  }
8
9  float square(float x){
10     cout << "float version called\n";
11     return x*x;
12 }
13
14 double square(double x){
15     cout << "double version called\n";
16     return x*x;
17 }
18
19 int main()
20 {
21     cout << "result = " << square(2) << "\n";
22     cout << "result = " << square(2.0f) << "\n";
23     cout << "result = " << square(2.0) << "\n";
24     return 0;
25 }
```

```
int version called
result = 4
float version called
result = 4
double version called
result = 4
```

Function Overloading

```
 1  #include <iostream>
 2  using namespace std;
 3
 4  int square(int x){
 5      return x*x;
 6  }
 7
 8  float square(float x){
 9      return x*x;
10  }
11
12  int main(){
13      cout << square(2.0);
14      return 0;
15  }
```

```
 In function 'int main()':
13:23: error: call of overloaded 'square(double)' is ambiguous
13:23: note: candidates are:
4:5: note: int square(int)
8:7: note: float square(float)
```

Function Templates

- Compact way to make overloaded functions
  - Generate separate function for different data types
- Format
  - Begin with keyword **`template`**
  - Formal type parameters in brackets **`<>`**
    - Every type parameter preceded by **`typename`** or **`class`** (synonyms)
    - Placeholders for built-in types (i.e., **`int`**) or user-defined types
    - Specify arguments types, return types, declare variables
  - Function definition like normal, except formal types used

Function Templates

- Example

```
template < class T > // or template <typename T>
T square( T value1 )
{
   return value1 * value1;
}
```

  - **T** is a formal type, used as parameter type
    - Above function returns variable of same type as parameter
  - In function call, **T** replaced by real type
    - If **int**, all **T**'s become **int**s

```
int x;
int y = square(x);
```

Function Templates

```
1  #include <iostream>
2  using namespace std;
3
4  template <typename currentType>
5  currentType justOneBefore(currentType x){
6      currentType y = x-1;
7      return y;
8  }
9
10 int main()
11 {
12     cout << "result = " << justOneBefore('Z') << "\n";
13     cout << "result = " << justOneBefore(0u) << "\n";
14     cout << "result = " << justOneBefore(-68) << "\n";
15     cout << "result = " << justOneBefore(6.55555) << "\n";
16     return 0;
17 }
```

```
result = Y
result = 4294967295
result = -69
result = 5.55555
```

Function Templates

```
 4  template <typename T>
 5  T justOneBefore(T);
 6
 7  int main()
 8  {
 9      cout << "result = " << justOneBefore('Z') << "\n";
10      cout << "result = " << justOneBefore(0u) << "\n";
11      cout << "result = " << justOneBefore(-68) << "\n";
12      cout << "result = " << justOneBefore(6.55555) << "\n";
13      return 0;
14  }
15
16  template <typename currentType>
17  currentType justOneBefore(currentType x){
18      currentType y = x-1;
19      return y;
20  }
```

Prototype

Different typename can be used in prototype and definition

```
result = Y
result = 4294967295
result = -69
result = 5.55555
```

Recursion (Recursive Function)

- Example: factorial

  $n! = n * (n - 1) * (n - 2) * \ldots * 1$

  – Recursive relationship (n! = n * (n – 1)!)

  $5! = 5 * 4!$

  $4! = 4 * 3!\ldots$

  – Base case (1! = 0! = 1)

Example 13-A: Factorial (recursive ver.)

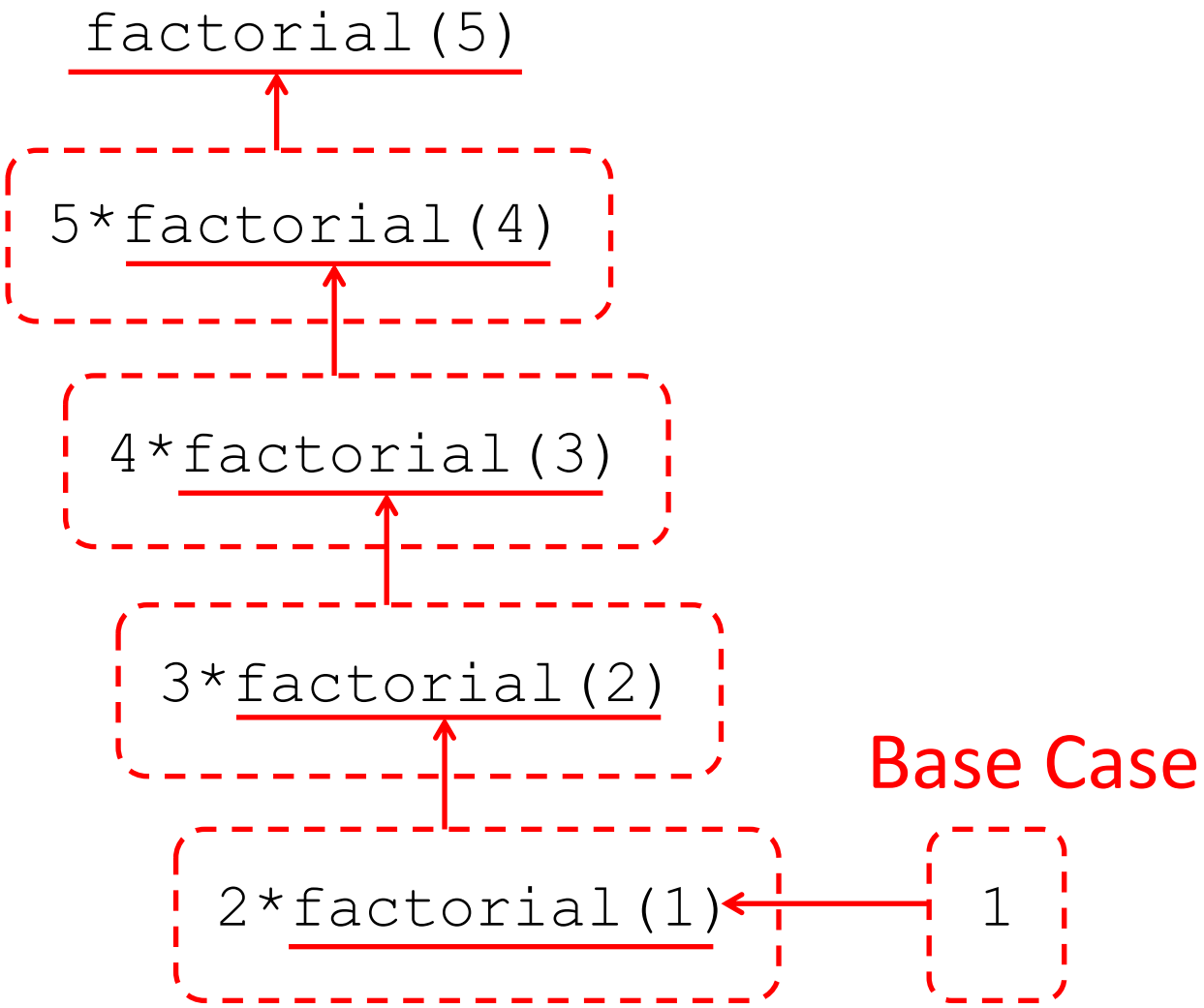
```
1   #include <iostream>
2   using namespace std;
3
4
5   int factorial (int x)
6   {
7       if(x <= 1){
8           return 1;
9       }else{
10          return x*factorial(x-1);
11      }
12
13  }
14
15  int main(){
16      cout << "5! = " << factorial(5);
17  }
```

Base case for 1! or 0! = 1

This function call itself inside
Factorial of `x` can be obtained
from factorial of smaller number
(`x-1`)

```
5! = 120
```

Example 13-A: Factorial (recursive ver.)



Example 13-A: Factorial (iteration ver.)

```
1   #include <iostream>
2   using namespace std;
3
4
5   int factorial(int x)
6   {
7       int y = 1;
8       for(;x>=2;x--){
9           y *= x;
10      }
11      return y;
12  }
13
14  int main(){
15      cout << "5! = " << factorial(5);
16  }
```

Use for-loop

```
5! = 120
```

Recursion vs. Iteration

จงหา 1+2+3+...+N

สิ่งที่ Iteration เห็น	สิ่งที่ Recursionเห็น	สิ่งที่นักศึกษาบางคนเห็น
$f(N) = \sum_{x=1}^{N} x$	$f(x) = x + f(x-1)$ $f(1) = 1$	*FFFFFFFFFFFF* *FFFFFFFFFFFF* *FFF* *FFFFFFFFFF* *FFF* *FFF*

Example 13-B: Print *

```cpp
#include <iostream>
using namespace std;

void printStar(int N){
    if(N == 1){
        cout << "*\n";
    }else if(N > 1){
        cout << "*";
        printStar(N-1);
    }
}

int main(){
    printStar(0);
    printStar(1);
    printStar(3);
    printStar(5);
}
```

```cpp
#include <iostream>
using namespace std;

void printStar(int N){
    if(N > 0){
        for(int i = 0; i < N;i++){
            cout << "*";
        }
        cout << "\n";
    }
}
```

```
*

***

*****
```

Example 13-C: Digital Root

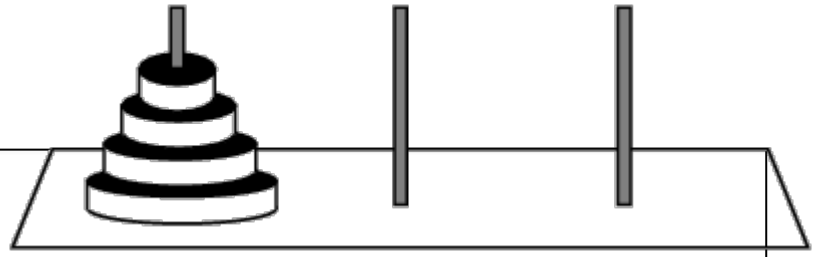
Digital root of 578698 = Digital root of 5+7+8+6+9+8 = Digital root of 43 = Digital root of 4+3 = Digital root of 7 = 7

```cpp
#include <iostream>
using namespace std;

int sumAllDigits(int x){
    int sum = 0;
    for(;x > 0;x/=10){
        sum += x%10;
    }
    return sum;
}

int digitalRoot(int x){
    if(x < 10) return x;
    else{
        int y = sumAllDigits(x);
        return digitalRoot(y);
    }
}
```

```cpp
int main(){
    cout << digitalRoot(123) << "\n";
    cout << digitalRoot(578689) << "\n";
    cout << digitalRoot(999) << "\n";
    cout << digitalRoot(69) << "\n";
    cout << digitalRoot(0) << "\n";
}
```

```
6
7
9
6
0
```

Example 13-D: Towers of Hanoi



```
1  #include <iostream>
2  using namespace std;
3
4  void hanoi(int N,int s=1, int d=3)
5  {
6      if(N == 1) cout << "Move from " << s << " to " << d << '\n';
7      else{
8          int buffer = 6-(s+d);
9          hanoi(N-1,s,buffer);
10         cout << "Move from " << s << " to " << d << '\n';
11         hanoi(N-1,buffer,d);
12     }
13 }
14
15 int main(){
16     hanoi(3);
17 }
```

```
Move from 1 to 3
Move from 1 to 2
Move from 3 to 2
Move from 1 to 3
Move from 2 to 1
Move from 2 to 3
Move from 1 to 3
```

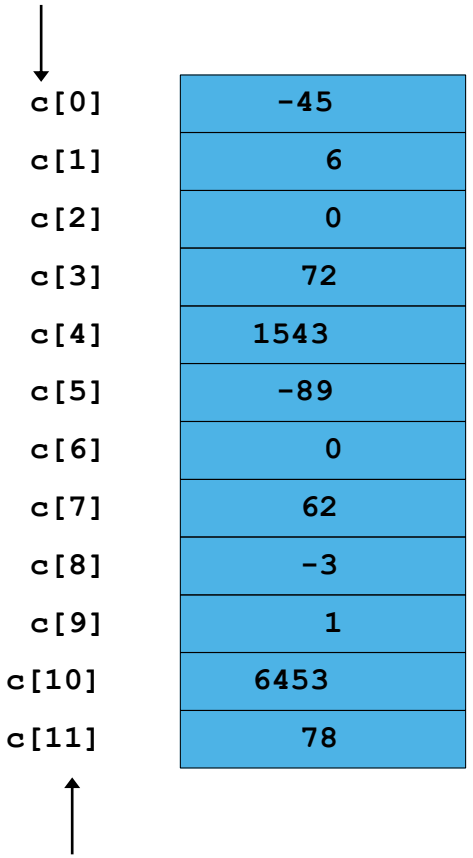
Arrays

```
1  #include <iostream>
2  using namespace std;
3
4  int main()
5  {
6      int c[12] = {-45,6,0,72,1543,-89,
7                   0,62,-3,1,6453,78};
8
9      int sum = 0;
10     for(int i = 0;i<12;i++){
11         sum += c[i];
12     }
13
14     cout << "size of element = " << sizeof(c[0]);
15     cout << "\nsize of array = " << sizeof(c);
16     cout << "\nsum = " << sum;
17 }
```

output

```
size of element = 4
size of array = 48
sum = 8078
```

Name of array
(Note that all elements of this array have the same name, **c**)



Position number of the element within array **c**

Arrays

```
1  #include <iostream>
2  using namespace std;
3
4  int main()
5  {
6      double a[5] = {5.5};
7      double b[10] = {5,5,5};
8      double c[] = {5,6,7};
9      double d[9] = {};
10     unsigned int i;
11
12     cout << "array a = ";
13     for(i=0; i < sizeof(a)/sizeof(a[0]); i++) cout << a[i] << " ";
14
15     cout << "\narray b = ";
16     for(i=0; i < sizeof(b)/sizeof(b[0]); i++) cout << b[i] << " ";
17
18     cout << "\narray c = ";
19     for(i=0; i < sizeof(c)/sizeof(c[0]); i++) cout << c[i] << " ";
20
21     cout << "\narray d = ";
22     for(i=0; i < sizeof(d)/sizeof(d[0]); i++) cout << d[i] << " ";
23
24     return 0;
25 }
```

```
array a = 5.5 0 0 0 0
array b = 5 5 5 0 0 0 0 0 0 0
array c = 5 6 7
array d = 0 0 0 0 0 0 0 0 0
```

Arrays

Value of name of array is address of first element

```
1   #include <iostream>
2   using namespace std;
3
4   int main()
5   {
6       double a[5] = {5.5};
7       double b[10] = {5,5,5};
8       double c[] = {5,6,7};
9       double d[9] = {};
10
11      cout << "a = " << a << "\n";
12      cout << "b = " << b << "\n";
13      cout << "c = " << c << "\n";
14      cout << "d = " << d << "\n";
15
16      return 0;
17  }
```

```
a = 0x22fd50
b = 0x22fdd0
c = 0x22fe20
d = 0x22fd80

--------------------------------
Process exited after 0.1177 seconds
Press any key to continue . . .
```

Arrays

```
Input number of your data: 3
Input your data [1]: 1.2
Input your data [2]: 3.6
Input your data [3]: 6.9
Your data = 1.2 3.6 6.9
```

```
1  #include <iostream>
2  using namespace std;
3
4  int main()
5  {
6      int N;
7      cout << "Input number of your data: ";
8      cin >> N;
9      float c[N];
10     for(int i=0;i<N;i++){
11         cout << "Input your data [" << i+1 << "]: ";
12         cin >> c[i];
13     }
14
15     cout << "Your data = ";
16     for(int i=0;i<N;i++) cout << c[i] << " ";
17
18     return 0;
19 }
```

Character Arrays (c-style string)

- Arrays of characters = Strings
- All strings end with `null` (`'\0'`)
- Examples
  - `char string1[] = "hello";`
    - `Null` character implicitly added
    - `string1` has 6 elements
  - `char string1[] = { 'h', 'e', 'l', 'l', 'o', '\0' };`
- Subscripting is the same

  `string1[ 0 ]` is `'h'`

  `string1[ 2 ]` is `'l'`

Character Arrays (c-style string)

```
CPE: size = 4
CPE: size = 4
CPE: size = 6
C = ASCII code 67
P = ASCII code 80
E = ASCII code 69
 = ASCII code 0
 = ASCII code 0
 = ASCII code 0
 = ASCII code 0
```

```
1  #include <iostream>
2  using namespace std;
3
4  int main()
5  {
6      char text1[4] = {'C','P','E','\0'};
7      char text2[] = "CPE";
8      char text3[6] = "CPE";
9
10     cout << text1 << ": size = "<< sizeof(text1) <<  "\n" ;
11     cout << text2 << ": size = "<< sizeof(text2) <<  "\n" ;
12     cout << text3 << ": size = "<< sizeof(text3) <<  "\n" ;
13     cout << text1[0] << " = ASCII code " << (int) text1[0] << "\n";
14     cout << text1[1] << " = ASCII code " << (int) text1[1] << "\n";
15     cout << text1[2] << " = ASCII code " << (int) text1[2] << "\n";
16     cout << text1[3] << " = ASCII code " << (int) text1[3] << "\n";
17     cout << text3[3] << " = ASCII code " << (int) text3[3] << "\n";
18     cout << text3[4] << " = ASCII code " << (int) text3[4] << "\n";
19     cout << text3[5] << " = ASCII code " << (int) text3[5] << "\n";
20
21     return 0;
22 }
```

Passing Arrays to Functions

- Specify name without brackets
  - To pass array **myArray** to **myFunction**

```
int myArray[24];
myFunction( myArray, 24 );
```

  - Array size usually passed, but not required
    - Useful to iterate over all elements

Passed-by-reference

	Normal Variable	Array
Prototype	`int func(int &);`	`int func(int[]);`

Definition	`int func(int &x){`
	
	`}`
		`int func(int A[]){`
		
		`}`

Calling	`func(x)`	`func(A)`

If `const` keyword is used to parameter(s) of the function when it declared

```
int func(const int[]);
```

then the function can't modify that parameter although it passed by reference

Passing Arrays to Functions

```
1   #include <iostream>
2   using namespace std;
3
4   double findRange(double[],int);
5
6   int main()
7   {
8       double data[6] = {2.5,-5.5,-6.9,13.25,4.77,-0.25};
9
10      cout << "Range of data = " << findRange(data,6);
11
12      return 0;
13  }
14
15  double findRange(double d[],int N){
16      double max = d[0],min = d[0];
17      for(int i=1;i<N;i++){
18          if(d[i]>max) max=d[i];
19          if(d[i]<min) min=d[i];
20      }
21      return max-min;
22  }
```

Prototype

This input parameter is an array of double

Function Call

Pass by reference

Pass by value

Definition

Range of data = 20.15

Passing Arrays to Functions

```
1  #include <iostream>
2  using namespace std;
3
4  long double findRange(long double[],int);
5
6  int main(){
7      long double data[6] = {2.5,-5.5,-6.9,13.25,4.77,-0.25};
8
9      cout << "Size of data = " << sizeof(data) << "\n";
10     cout << "Value of data[0] = " << data[0] << "\n";
11     cout << "Size of data[0] = " << sizeof(data[0]) << "\n";
12     cout << "Range of data = " << findRange(data,6);
13
14     return 0;
15 }
16
17 long double findRange(long double d[],int N){
18     long double max = d[0],min = d[0];
19     cout << "Value of d[0] = " << d[0] << "\n";
20     cout << "Size of d[0] = " << sizeof(d[0]) << "\n";
21     cout << "Value of d = " << d << "\n";
22     cout << "Size of d = " << sizeof(d) << "\n";
23     for(int i=1;i<N;i++){
24         if(d[i]>max) max=d[i];
25         if(d[i]<min) min=d[i];
26     }
27     return max-min;
28 }
```

```
Size of data = 96
Value of data[0] = 2.5
Size of data[0] = 16
Value of d[0] = 2.5
Size of d[0] = 16
Value of d = 0x7829d103
Size of d = 8
Range of data = 20.15
```

Passing Arrays to Functions

```
1   #include <iostream>
2   using namespace std;
3
4   void half(double[],int);
5
6   int main(){
7       double data[3] = {12.2,5.6,3.7};
8       half(data,3);
9       cout << "new data = ";
10      for(int i=0;i < 3; i++) cout << data[i] << " ";
11
12      return 0;
13  }
14
15  void half(double d[],int N){
16      for(int i=0;i < N; i++) d[i] /= 2;
17  }
```

new data = 6.1 2.8 1.85

Example 14-A: Adding Two Arrays

```
 1  #include <iostream>
 2  using namespace std;
 3
 4  void addArray(const double[],const double[],double[],int);
 5
 6  int main(){
 7      double x[3] = {1.2,3.4,5.6};
 8      double y[3] = {-0.5,6.9,4.1};
 9      double z[3];
10      addArray(x,y,z,3);
11      cout << "result = ";
12      for(int i=0;i < 3; i++) cout << z[i] << " ";
13
14      return 0;
15  }
16
17  void addArray(const double src1[],const double src2[],double dest[],int N){
18      for(int i=0;i < N; i++) dest[i] = src1[i]+src2[i];
19  }
```

Do not allow function to modify **src1** and **src2**

result = 0.7 10.3 9.7

Allow function to modify **dest**

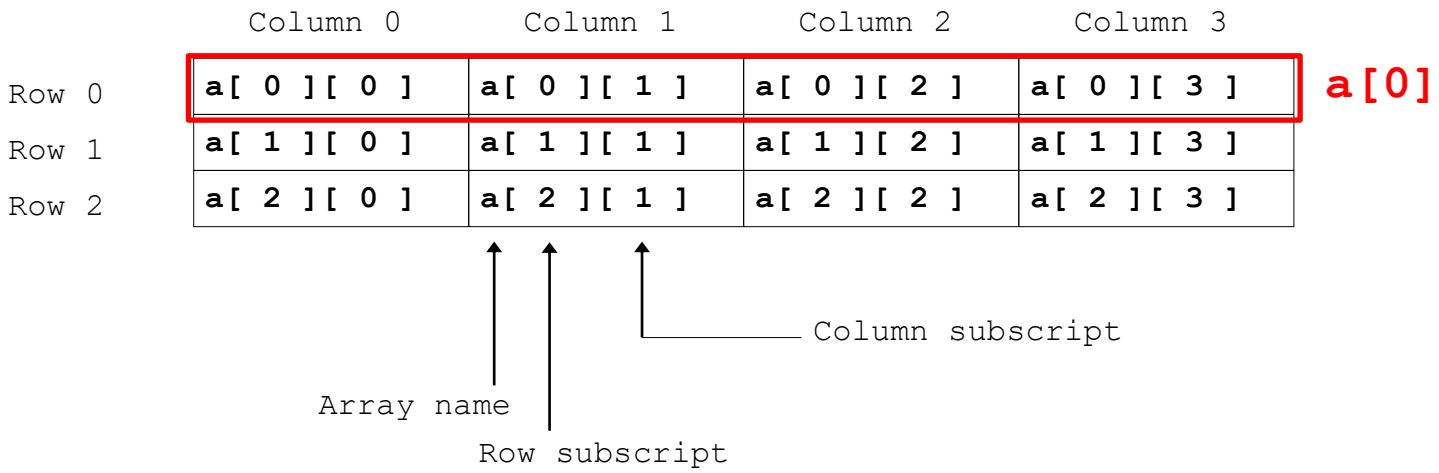
Passing Array Elements to Functions

```
1  #include <iostream>
2  using namespace std;
3
4  void check(double);
5
6  int main(){
7      double score[3] = {12.3,69,55.55};
8      for(int i=0;i < 3; i++) check(score[i]);
9
10     return 0;
11 }
12
13 void check(double s){
14     if(s >= 50) cout << "Pass\n";
15     else cout << "Failed\n";
16 }
```

```
Failed
Pass
Pass
```

Multiple-Subscripted Arrays

- Multiple subscripts (Multidimensional arrays)
  - `a[ i ][ j ]`
  - Tables with rows and columns
  - Specify row, then column
  - "Array of arrays"
    - `a[0]` is an array of 4 elements
    - `a[0][0]` is the first element of that array



Multiple-Subscripted Arrays

- To initialize
  - Default of 0
  - Initializers grouped by row in braces

```
int b[ 2 ][ 2 ] = { { 1, 2 }, { 3, 4 } };
```

Row 0 Row 1

1	2
3	4

```
int b[ 2 ][ 2 ] = { { 1 }, { 3, 4 } };
```

1	0
3	4

Multiple-Subscripted Arrays

- Referenced like normal

```
cout << b[ 0 ][ 1 ];
```

1	0
3	4

  - Outputs 0
  - Cannot reference using commas

```
cout << b[ 0, 1 ];
```

    - Syntax error

- Function prototypes
  - Must specify sizes of subscripts
    - First subscript not necessary, as with single-scripted arrays
  - `void printArray( int [][ 3 ] );`

Multiple-Subscripted Arrays

```
1  #include <iostream>
2  using namespace std;
3
4  int main(){
5      int A[2][4] = {{1,2,3,4},{5,6,7,8}};
6
7      for(int i=0;i < 2; i++){
8          for(int j=0;j < 4; j++){
9              cout << A[i][j] << " ";
10         }
11         cout << "\n";
12     }
13     cout << "size A = " << sizeof(A) << "\n";
14     cout << "size A[0] = " << sizeof(A[0]) << "\n";
15     cout << "size A[0][0] = " << sizeof(A[0][0]) << "\n";
16     return 0;
17 }
```

```
1 2 3 4
5 6 7 8
size A = 32
size A[0] = 16
size A[0][0] = 4
```

Multiple-Subscripted Arrays

Initializer Lists	Results
int A[2][4] = {{1,2,3,4}};	1 2 3 4 0 0 0 0
int A[2][4] = {1,2,3,4,5,6};	1 2 3 4 5 6 0 0
int A[2][4] = {1};	1 0 0 0 0 0 0 0
int A[2][4] = {{1},{2}};	1 0 0 0 2 0 0 0
int A[2][4] = {{},{2,3}};	0 0 0 0 2 3 0 0
int A[2][4] = {};	0 0 0 0 0 0 0 0
int A[2][4] = {1,2,3,4,5,6,7,8,9};	too many initializers

Example 14-B: Determinant (2x2)

$$\begin{vmatrix} a & b \\ c & d \end{vmatrix} = ad - bc$$

```
 1  #include <iostream>
 2  using namespace std;
 3
 4  double det(double[][2]);
 5
 6  int main(){
 7      double A[2][2];
 8      cout << "Input First Row:";
 9      cin >> A[0][0] >> A[0][1];
10      cout << "Input Second Row:";
11      cin >> A[1][0] >> A[1][1];
12      cout << "Determinant = " << det(A);
13      return 0;
14  }
15
16  double det(double x[][2]){
17      return x[0][0]*x[1][1]-x[1][0]*x[0][1];
18  }
```

```
Input First Row:1 2
Input Second Row:3 4
Determinant = -2
```

Example 14-C: Score Table

```
 1  #include <iostream>
 2  #include <iomanip>
 3  #include <string>
 4  using namespace std;
 5
 6  void findColAvg(const int [][3],double [],int);
 7  void showTable(const string[], const int [][3], const double [],int);
 8
 9  int main()
10  {
11      int score[][3] = {{10,20,30},{20,39,69},{10,12,10},{30,40,88}};
12      string name[] = {"Sansanee","Roger","Dome","Yuthapong"};
13      double avg[3] = {};
14      findColAvg(score,avg,4);
15      showTable(name,score,avg,4);
16  }
```

```
      Quiz        1        2        3
-------------------------------------
  Sansanee       10       20       30
     Roger       20       39       69
      Dome       10       12       10
 Yuthapong       30       40       88
-------------------------------------
   Average    17.50    27.75    49.25
```

Example 14-C: Score Table

```
18 void findColAvg(const int src[][3],double dest[],int N){
19     for(int i=0;i<N;i++){
20         dest[0]+=src[i][0];
21         dest[1]+=src[i][1];
22         dest[2]+=src[i][2];
23     }
24     dest[0]/=N;
25     dest[1]/=N;
26     dest[2]/=N;
27 }
```

Quiz	1	2	3
Sansanee	10	20	30
Roger	20	39	69
Dome	10	12	10
Yuthapong	30	40	88
Average	17.50	27.75	49.25

```
29 void showTable(const string n[], const int s[][3],const double a[],int N){
30     cout << setw(12) << "Quiz";
31     for(int j=1;j <= 3;j++)cout << setw(8) << j;
32     cout << "\n---------------------------------------\n";
33     for(int i=0;i<N;i++){
34         cout << setw(12) << n[i];
35         for(int j=0;j < 3;j++)cout << setw(8) << s[i][j];
36         cout << "\n";
37     }
38     cout << "---------------------------------------\n";
39     cout << fixed << setprecision(2);
40     cout << setw(12) << "Average";
41     for(int j=0;j < 3;j++)cout << setw(8) << a[j];
42 }
```

Bubble Sort

```cpp
#include <iostream>
using namespace std;

void swap(double d[],int x,int y){
    double temp = d[x];
    d[x] = d[y];
    d[y] = temp;
}

void show(double d[],int N){
    for(int i = 0; i < N; i++){
        cout << d[i] << " ";
    }
    cout << "\n";
}

void moveMax2End(double d[],int e){
    for(int i = 0; i < e-1; i++){
        if(d[i] > d[i+1]) swap(d,i,i+1);
    }
}

void bubbleSort(double d[],int N){
    for(int end = N; end > 1; end--){
        moveMax2End(d,end);
        show(d,N);
    }
}
```

```cpp
int main(){
    double data[] = {7,4,5,2,1,4};
    int N = sizeof(data)/sizeof(data[0]);
    cout << "Original: ";
    show(data,N);
    cout << "Sorting Process:\n";
    bubbleSort(data,N);
    return 0;
}
```

```
Original: 7 4 5 2 1 4
Sorting Process:
4 5 2 1 4 7
4 2 1 4 5 7
2 1 4 4 5 7
1 2 4 4 5 7
1 2 4 4 5 7
```

Bubble Sort

```
30  int main(){
31      double data[] = {1,3,2,5,12,9,8};
32      int N = sizeof(data)/sizeof(data[0]);
33      cout << "Original: ";
34      show(data,N);
35      cout << "Sorting Process:\n";
36      bubbleSort(data,N);
37      return 0;
38  }
```

```
Original: 1 3 2 5 12 9 8
Sorting Process:
1 2 3 5 9 8 12
1 2 3 5 8 9 12
1 2 3 5 8 9 12
1 2 3 5 8 9 12
1 2 3 5 8 9 12
1 2 3 5 8 9 12
```

Bubble Sort

```
17 void moveMax2End(double d[],int e){
18     for(int i = 0; i < e-1; i++){
19         if(d[i] > d[i+1]) swap(d,i,i+1);
20     }
21 }
22
23 void bubbleSort(double d[],int N){
24     for(int end = N; end > 1; end--){
25         moveMax2End(d,end);
26         show(d,N);
27     }
28 }
```

```
Original: 1 3 2 5 12 9 8
Sorting Process:
1 2 3 5 9 8 12
1 2 3 5 8 9 12
1 2 3 5 8 9 12
```

```
16 bool moveMax2End(double d[],int e){
17     bool swap_flag = false;
18     for(int i = 0; i < e-1; i++){
19         if(d[i] > d[i+1]){
20             swap(d,i,i+1);
21             swap_flag = true;
22         }
23     }
24     return swap_flag;
25 }
26
27 void bubbleSort(double d[],int N){
28     for(int end = N; end > 1; end--){
29         bool didSwap = moveMax2End(d,end);
30         show(d,N);
31         if(!didSwap) break;
32     }
33 }
```

Linear Search

```
1  #include <iostream>
2  using namespace std;
3
4  int linearSearch(int data[], int N, int key){
5      for(int i=0; i< N;i++){
6          if(data[i] == key) return i;
7      }
8      return -1;
9  }
10
11 int main()
12 {
13     int data[] = {5,8,9,69,23,55,10,19,7,30,1,2};
14     int N = sizeof(data)/sizeof(data[0]);
15     int key;
16     cout << "Input a number to search: ";
17     cin >> key;
18
19     int loc = linearSearch(data,N,key);
20
21     if(loc == -1) cout << "Not found";
22     else cout << "Found at data[" << loc << "]";
23
24     return 0;
25 }
```

```
Input a number to search: 69
Found at data[3]
```

Example 15-A: Overweight

```cpp
#include <iostream>
#include <string>
using namespace std;

int linearSearch(string data[], int N, string key){
    for(int i=0; i< N;i++){
        if(data[i] == key) return i;
    }
    return -1;
}

int main()
{
    string name[] = {"Hashida","Haruyuki","Machvise","Choji","Jaiko","Biscuit","Linlin"};
    double weight[] = {72.5,75.2,88,73,70.6,84.9,99.99};
    int N = sizeof(weight)/sizeof(weight[0]);
    string key;
    cout << "Input name to find weight: ";
    cin >> key;

    int loc = linearSearch(name,N,key);

    if(loc == -1) cout << "Not found";
    else cout << key << "'s weight is " << weight[loc] << "kg";

    return 0;
}
```

```
Input name to find weight: Hashida
Hashida's weight is 72.5kg
```

```
Input name to find weight: Linlin
Linlin's weight is 99.99kg
```

Binary Search

```
#include <iostream>
using namespace std;

void swap(int d[],int x,int y){
    int temp = d[x];
    d[x] = d[y];
    d[y] = temp;
}

void show(int d[],int N){
    for(int i = 0; i < N; i++){
        cout << d[i] << " ";
    }
    cout << "\n";
}

void moveMax2End(int d[],int e){
    for(int i = 0; i < e-1; i++){
        if(d[i] > d[i+1]) swap(d,i,i+1);
    }
}

void bubbleSort(int d[],int N){
    for(int end = N; end > 1; end--){
        moveMax2End(d,end);
    }
}
```

```
int main()
{
    int data[] = {5,8,9,69,23,55,10,19,7,30,1,2};
    int N = sizeof(data)/sizeof(data[0]);
    int key;
    cout << "Input a number to search: ";
    cin >> key;

    cout << "Sorted data: ";
    bubbleSort(data,N);
    show(data,N);

    int loc = binarySearch(data,N,key);
    cout << "\n";

    if(loc == -1) cout << "Not found";
    else cout << "Found at data[" << loc << "]";

    return 0;
}
```

```
Input a number to search: 69
Sorted data: 1 2 5 7 8 9 10 19 23 30 55 69
Comparison Sequence: 9 23 55 69
Found at data[11]
```

Binary Search

```
Input a number to search: 69
Sorted data: 1 2 5 7 8 9 10 19 23 30 55 69
Comparison Sequence: 9 23 55 69
Found at data[11]
```

```
ta[] = {5,8,9,69,23,55,10,19,7,30,1,2};
= sizeof(data)/sizeof(data[0]);
y;
47      cout << "Input a number to search: ";
48      cin >> key;
49
50      cout << "Sorted data: ";
51      bubbleSort(data,N);
52      show(data,N);
53
54      int loc = binarySearch(data,N,key);
55      cout << "\n";
56
= -1) cout << "Not found";
t << "Found at data[" << loc << "]";
;
```

```
29  int binarySearch(int data[], int N, int key){
30      cout << "Comparison Sequence: ";
31      int first = 0, last = N-1, mid;
32      do{
33          mid = (first+last)/2;
34          cout << data[mid] << " ";
35          if(data[mid] == key)            ;
36          else if(data[mid] > key)              ;
37          else                ;
38      }while(first <= last);
39      return -1;
40  }
```

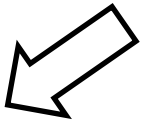
Example 15-B: Over-grade

```
41  int main()
42  {
43      unsigned int StudentID[] = {770610806,770610784,770610744,770610781,770612088,
44                                  770610778,770610810,770610841,770610777,770612084,
45                                  770610845,770610304,770610834};
46      string grade102[] = {"F","D","D+","W","D","W","F","D+","W","F","D+","F","D"};
47      int N = sizeof(StudentID)/sizeof(StudentID[0]);
48      unsigned int key;
49      cout << "Input a Student ID to search: ";
50      cin >> key;
51
52      bubbleSort(StudentID,grade102,N);
53      int loc = binarySearch(StudentID,N,key);
54
55      if(loc == -1) cout << "Student ID " << key << "was not found." ;
56      else cout << "Student ID " << key << " got " << grade102[loc] << " in 261102";
57
58      return 0;
59  }
```

```
Input a Student ID to search: 770610781
Student ID 770610781 got W in 261102
```

Example 15-B: Over-grade

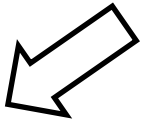
Use function template make function more flexible to the variation of data types



```
29  template <typename T>
30  int binarySearch(T data[], int N, T key){
31      int first = 0, last = N-1, mid;
32      do{
33          mid = (first+last)/2;
34          if(data[mid] == key) return mid;
35          else if(data[mid] > key) last = mid-1;
36          else first = mid+1;
37      }while(first <= last);
38      return -1;
39  }
```

Example 15-B: Over-grade

```
1   #include <iostream>
2   #include <string>
3   using namespace std;
4
5   template <typename T>
6   void swap(T d[],int x,int y){
7       T temp = d[x];
8       d[x] = d[y];
9       d[y] = temp;
10  }
11
12  template <typename T1, typename T2>
13  void moveMax2End(T1 d[], T2 v[],int e){
14      for(int i = 0; i < e-1; i++){
15          if(d[i] > d[i+1]){
16              swap(d,i,i+1);
17              swap(v,i,i+1);
18          }
19      }
20  }
21
22  template <typename T1, typename T2>
23  void bubbleSort(T1 d[], T2 v[],int N){
24      for(int end = N; end > 1; end--){
25          moveMax2End(d,v,end);
26      }
27  }
```



Function template with two type names