261102
Computer Programming

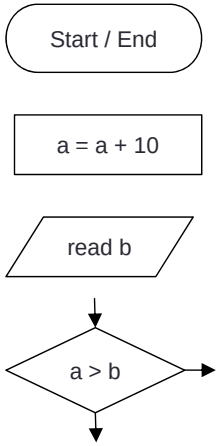
Lecture 3: Selection Structures

Algorithms

- Computing problems
  - Solved by executing a series of actions in a specific order
- Algorithm a procedure determining
  - Actions to be executed
  - Order to be executed
  - Example: recipe
- Program control
  - Specifies the order in which statements are executed

Flowchart

- Graphical representation of an algorithm
- Special-purpose symbols connected by arrows (flowlines)



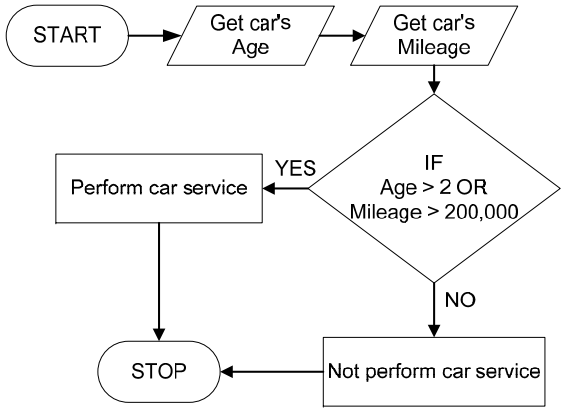
Terminal Symbol : start or end of a process

Process Symbol : typical process/computation

Input/Output Symbol : receive the input or display the output

Decision Symbol : contain a Yes/No question and has two arrows coming out, with one corresponding to Yes or True, and the other one corresponding to No or False

Flowchart



Pseudocode

- Pseudo Code = Informal language used to develop algorithms without having to worry about the details of language syntax.
- Normally describes only executable statements (not including the variable declaration)

```
1  Prompt the user to enter the first integer
2  Input the first integer
3
4  Prompt the user to enter the second integer
5  Input the second integer
6
7  Add first integer and second integer, store result
8  Display result
```

```
1  function divide (x, y);
   Input: Two n-bit integers x and y, where y ≥ 1
   Output: The quotient and remainder of x divided by y
2  if x = 0 then
3      return (q, r) = (0, 0)
4  else
5      set (q, r) = divide(⌊x/2⌋, y);
6      q = 2 × q, r = 2 × r;
7      if x is odd then
8          r = r + 1
9      end
10     if r ≥ y then
11         r = r − y, q = q + 1
12     end
13     return (q, r)
14 end
```


Control Structures

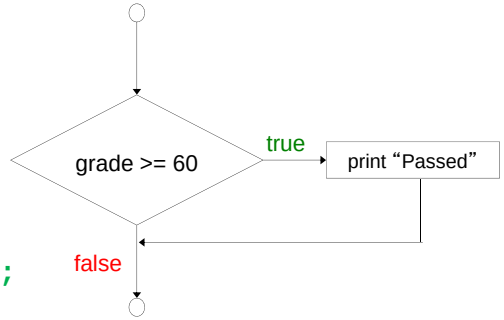
- Normally, statements in a program execute one after the other in the order in which they're written. This is called sequential execution.
- Transfer of control = Next statement executed not next one in sequence
- There are 3 control structures:
  - Sequence structure
    - Programs executed sequentially by default
  - Selection structures
    - **`if`**, **`if/else`**, **`switch`**
  - Repetition structures
    - **`while`**, **`do/while`**, **`for`**

if Selection Structure

- Choose among alternative courses of action
- Pseudocode example:

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

- Flowchart example:

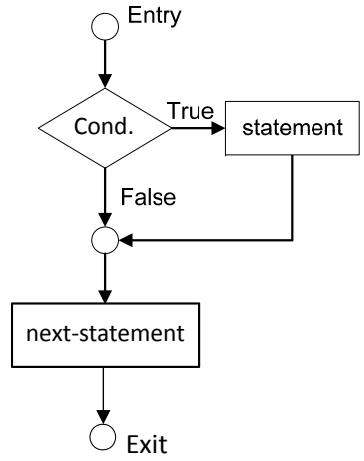


- Translation into C++

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


if Single Statement

```
if (condition)
  statement;
next-statement;
```



- If the **`condition`** is **`true`**
  - **`statement`** executed, program continues to **`next-statement`**
- If the **`condition`** is **`false`**
  - **`statement`** ignored and program continues to **`next-statement`**

if Single Statement

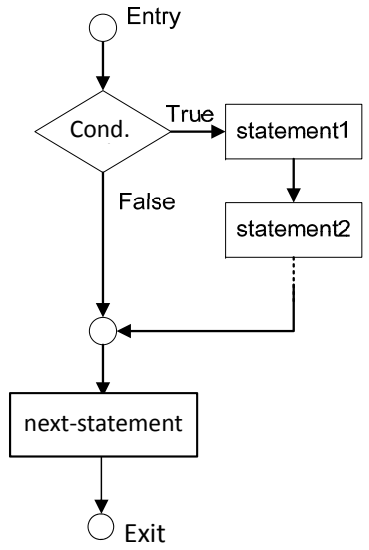
```
#include <iostream>
using namespace std;
int main() {
    int number;

    cout << "Type in your number: ";
    cin >> number;

    if (number < 0)
        number = - number;
    cout << "The absolute value is " << number << endl;
    return 0;
}
```

```
Type in your number: 5
The absolute value is 5
```

```
Type in your number: -7
The absolute value is 7
```


if Compound 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

if Compound Statement

```
#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
```

```
#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
```

```
#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
```

```
#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;
}
```

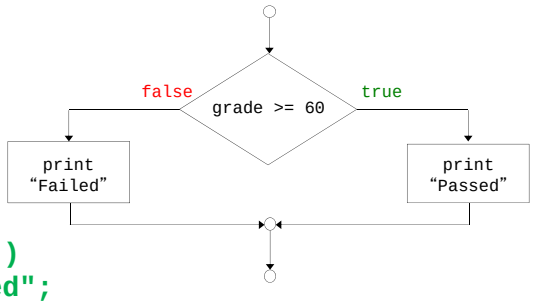
```
Print 4
Print 5
```


if-else Statement

- **if** = Performs action if condition **true**
- **if/else** = Different actions if conditions **true** or **false**
- Pseudocode example:

if student's grade is greater than or equal to 60
* print "Passed"*
else
* print "Failed"*

- Flowchart example:



- Translation into C++

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

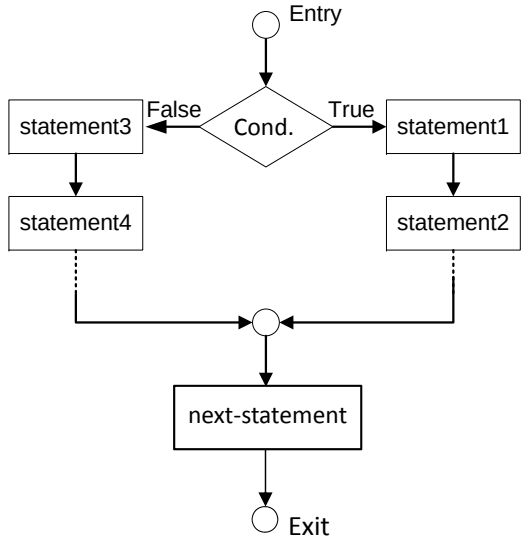
if-else Statement

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

- If the **condition** is **true**
  - **statement1, statement2, …** are executed in order then program continues to **next-statement**
- If the **condition** is **false**
  - **statement3, statement4, …** are executed in order then program continues to **next-statement**

if-else Statement

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



Example 3-A: Temperature Conversion

```
1  #include <iostream>
2  #include <iomanip>
3  using namespace std;
4  int main() {
5      char type;
6      float temp, fahren, celcius;
7      cout << "Enter temperature: ";
8      cin >> temp;
9      cout << "Enter f (fahrenheit) or c (celcius) for unit: ";
10     cin >> type;
11     cout << fixed << showpoint << setprecision(2);
12     if (type == 'f') {
13         celcius = (5.0/9.0) * (temp-32.0);
14         cout << "The equivalent celcius temp. is " << celcius << endl;
15     } else {
16         fahren = (9.0/5.0) * temp + 32.0;
17         cout << "The equivalent fahrenheit temp. is " << fahren << endl;
18     }
19     return 0;
20 }
```

```
Enter temperature: 98.9
Enter f (farenheit) or c (celcius) for unit: f
The equivalent celcius temp. is  37.17
```

Note: Indenting makes programs easier to read

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

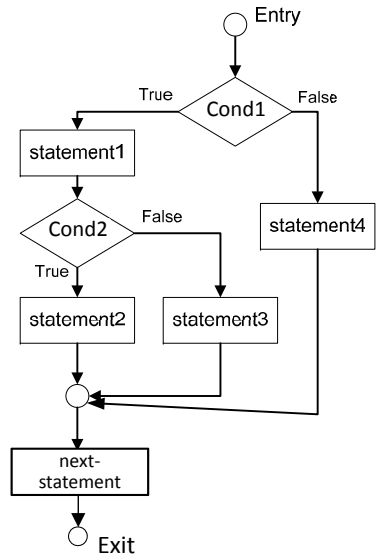
- **statement1** is always executed when **condition1** is **true**
- **statement2** is executed after **statement1** only when **condition1** and **condition2** both are **true**
- **statement3** is executed after **statement1** only when **condition1** is **true** but **condition2** is **false**
- **statement4** is always executed when **condition1** is **false**

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



Example 3-B: Check input numbers

```
1  #include <iostream>
2  using namespace std;
3
4  int main()
5  {
6    int num1, num2;
7    cout << "Program to check if num1 is [0-9] and num2 is [10-99]\n";
8    cout << "Enter num1 and num2: ";
9    cin >> num1 >> num2;
10
11   if (num1>=0 && num1<=9)
12     if (num2>=10 && num2<=99)
13         cout << "Both numbers are entered correctly";
14     else
15         cout << "only num1 is entered correctly";
16   else
17     if (num2>=10 && num2<=99)
18         cout << "only num2 is entered correctly";
19     else
20         cout << "Both numbers are NOT entered correctly";
21
22   return 0;
23 }
```


if-else if Statement

- Nested structure that one if-else statement inside another else statement.
- Pseudocode example:

```
if student's grade is greater than or equal to 90
      Print "A"
else
      if student's grade is greater than or equal to 80
        Print "B"
      else
        if student's grade is greater than or equal to 70
          Print "C"
        else
          if student's grade is greater than or equal to 60
            Print "D"
          else
                Print "F"
```


if-else if Statement

- Translation into C++:

```
if ( grade >= 90 )                  // 90 and above
   cout << "A";
else
    if ( grade >= 80 )              // 80-89
       cout << "B";
    else
       if ( grade >= 70 )           // 70-79
          cout << "C";
       else
          if ( grade >= 60 )        // 60-69
             cout << "D";
          else                      // less than 60
             cout << "F";
```

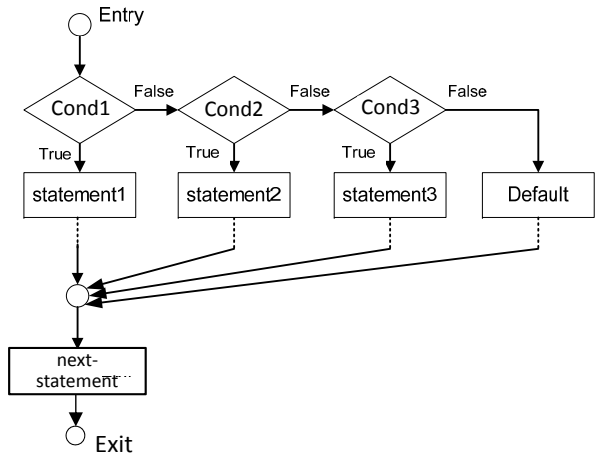
if-else if Statement

- Typical layout for `if-else if` statement

```
if ( grade >= 90 )       // 90 and above
   cout << "A";
else if ( grade >= 80 )  // 80-89
   cout << "B";
else if ( grade >= 70 )  // 70-79
   cout << "C";
else if ( grade >= 60 )  // 60-69
   cout << "D";
else                     // less than 60
   cout << "F";
```


if-else if Statement

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



if-else if Statement

```
if ( grade >= 90 )
   cout << "A";
else if ( grade >= 80 )
   cout << "B";
else if ( grade >= 70 )
   cout << "C";
else if ( grade >= 60 )
   cout << "D";
else
   cout << "F";
```

```
if ( grade >= 90 )
   cout << "A";
if ( grade >= 80 )
   cout << "B";
if ( grade >= 70 )
   cout << "C";
if ( grade >= 60 )
   cout << "D";
else
   cout << "F";
```

What is a difference?

Example 3-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 == value1**

Used for terminate from **switch** statement after **statement1, statement2, …** are executed

- do not test the variable for the next value
- **break** is executed only when **variable == value1**

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 == value2** or **variable == value3**

Used for exit from **switch** statement after **statement3, statement4, …** are executed

- do not execute the statements in the following default case
- **break** is executed only when **variable == value2 || variable == value3**

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

```
#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;
}
```

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

```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";
            .....
        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;
}
```

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

Example 3-D: Ranking Rewards

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

int main() {
    char rank;
    cout << "Input your rank: ";
    cin >> rank;
    switch(rank){
        case 'S': cout << "Congrats!!! You have received Super Ultimate Rare Unit!!!\n";
        case 'A': cout << "Congrats!!! You have received 5 gems.\n";
        case 'B': cout << "Congrats!!! You have received 1 gems.\n";
        case 'C': cout << "Congrats!!! You have received 2000 coins.\n";
        case 'D': cout << "Congrats!!! You have received 500 coins.\n";
    }
    return 0;
}
```

```
Input your rank: S
Congrats!!! You have received Super Ultimate Rare Unit!!!
Congrats!!! You have received 5 gems.
Congrats!!! You have received 1 gems.
Congrats!!! You have received 2000 coins.
Congrats!!! You have received 500 coins.
```

```
Input your rank: C
Congrats!!! You have received 2000 coins.
Congrats!!! You have received 500 coins.
```


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**.

```cpp
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 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
```


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';
}
```