Flow Charts

Fall 2015

Jinkyu Jeong
(jinkyu@skku.edu)
Flow Chart

- A flowchart is a type of diagram that represents an *algorithm or process*, showing the steps as various symbols, and their order by connecting them with arrows.

- The diagrammatic representation shows a *solution* to a given *problem*. 
A Quick Glance at Flow Chart

Start

Var a, b, c

Input a, b, c

\[-b + \sqrt{b^2 - 4ac} \quad , \quad -b - \sqrt{b^2 - 4ac}\]

\[
\frac{-b + \sqrt{b^2 - 4ac}}{2a} \quad , \quad \frac{-b - \sqrt{b^2 - 4ac}}{2a}
\]

Stop
Why Flow Chart?

• Flow charts are useful in analyzing, designing, documenting or managing a process or program.
  – The process is more understandable for humans.
  – Potential errors/mistakes/exceptions are minimized.
  – You can easily organize large problems.

• Well-established flow charts can be easily translated to computer programs (here, to "C" language).
Flow Chart Components

• **Arrows:**  
  • Shows “flow of control”

• **Start and end symbols:**  
  
  - Represents the start and end of a process

• **Generic processing steps:**  
  
  - Represents generic computations (e.g., add 1 to x)

• **Prepare conditional (declare variables):**  
  
  - Shows operations which have no effect other than preparing a value for a subsequent conditional or decision step
  - Declare variables for the flow chart
Flow Chart Components

• **Input/Output:**
  - Data input/output given by a user
  
  ![Input/Output Example](read x)

• **Document:**
  - Report data so that humans can read them
  
  ![Document Example](print x or “hello”)

• **Decision (or Conditional):**
  - Make a decision on which branch the flow goes to.
  - Commonly asks yes/no or true/false question
  
  ![Decision Example](x > 1)

---

GEDB029: Basis and Practice in Programming | Fall 2015 | Jinkyu Jeong (jinkyu@skku.edu)
How to Make Flow Chart

- Construct it from the primary basic elements
- Expand it step by step
- Test whether all the cases are handled
- Test again whether correct outputs are obtained.
Given \( a, b \) and \( c \), find the solutions of \( ax^2 + bx + c = 0 \)

\[
\begin{align*}
\text{Start} \\
\text{Var } a, b, c \\
\text{Input } a, b, c \\
\frac{-b + \sqrt{b^2 - 4ac}}{2a}, \frac{-b - \sqrt{b^2 - 4ac}}{2a} \\
\text{Stop}
\end{align*}
\]
How to Make Flow Chart: Examples

- Given $a$, $b$ and $c$, find the solutions of $ax^2+bx+c=0$

```
Given a, b and c, find the solutions of ax^2+bx+c=0

How to Make Flow Chart: Examples

Start

Var a, b, c

Input a, b, c

a = 0

T

-c/b

F

-b+sqrt(b^2-4ac) -b-sqrt(b^2-4ac)
--------------------- , ---------------------
2a 2a

Stop
```
Given $a$, $b$ and $c$, find the solutions of $ax^2+bx+c=0$
How to Make Flow Chart: Examples

- Given \( a, b \) and \( c \), find the solutions of \( ax^2 + bx + c = 0 \)

```
Start

Var \( a, b, c \)

Input \( a, b, c \)

\[ \text{T} \]
\[ c = 0 \]
\[ \text{T} \]
\[ b = 0 \]
\[ \text{T} \]
\[ a = 0 \]
\[ \text{F} \]
\[ b^2 - 4ac \geq 0 \]

\[ \text{T} \]
\[ -b + \sqrt{b^2 - 4ac} \]
\[ \frac{2a}{2a} \]
\[ \text{Stop} \]

\[ \text{T} \]
\[ -c/b \]

\[ \text{F} \]
\[ \text{F} \]

\[ \text{F} \]

\[ \text{Nothing} \]

\[ \text{All real numbers} \]

\[ \text{No real numbers} \]
```
Example 1

- Read a number and print it

```
Start
Var x
Input x
Print x
Stop
```
Example 2

- Read a number and print "Even" if it is even or "Odd" if it is odd

```
Start
Var x
Input x
x is even ?
T
"Even"
F
"Odd"
Stop
```
Example 3

- Add all integers between 1 and 100

```
Start
Var i, sum
i <- 1
sum <- 0
i <= 100
F
sum <- sum + i
i <- i + 1
T
Stop
```

```
Example 4

Add all even integers between 1 and 100

Start

Var i, sum

i ← 1, sum ← 0

i ≤ 100

i is even?

sum ← sum + i

i ← i + 1

Stop

F

T
Example 5

- Find the maximum among 3 numbers

```
Start
Var a, b, c
Input a, b, c

a > b

T
a > c

a

F
b > c

b

c

Stop
```
Example 6

- Sort 3 numbers

```plaintext
Start

Var a, b, c

Input a, b, c

a > b

b > c

a > c

Stop
```

- Example 6: Sort 3 numbers

```plaintext
Example 6: Sort 3 numbers

Start

Var a, b, c

Input a, b, c

a > b

b > c

a > c

Stop
```
Example 7

- Find the largest \( n \) such that \( 1+2+\ldots+n<1000 \)

```
Start

Var \( n \), \( sum \)

\( n \leftarrow 1, sum \leftarrow 0 \)

\( sum \leftarrow sum + n \)

\( sum < 1000 \) 

\( n \leftarrow n + 1 \) 

\( sum < 1000 \)

\( n-1 \) 

Stop
```
Example 8

- $n$-th term of Fibonacci sequence
  - $a_1=1$, $a_2=1$, $a_n=a_{n-1}+a_{n-2}$

```
Start
Var a, b, c, k, n
Input n
n = 1 or 2
F
  T
  1

k <= n
T
  c <- a + b
  a <- b
  b <- c
  k <- k + 1
F
a <- 1, b <- 1, k <- 3

Stop
```

$\text{c}$
Example 9

- **Given** n, 1+2+3+...+n

```
Var n, a, k

Input n

k <- 1, a <- 0

k <= n

F  |  a

T  |  a <- a + k
    |  k <- k + 1

Stop
```
Example 10

- $1+(1+2)+(1+2+3)+\ldots+(1+\ldots+10)$

```
Start

Var n, a, sum

sum <- 0

n <= 10

T

sum <- sum + a

n <- n + 1

a <- a + n

F

sum

Stop
```
Flow Chart and C
Basic Structure of C program

- In the beginning, there are a couple of `#include <...>`
  - Include standard library (routines) for basic functionalities

- All programs have one `main()` function which is an entry of each program
  - `void main(void)`
    
    ```c
    { 
      statements...
    }
    ```

- Most of the statements should be terminated with `;`

- Case-sensitive
  - E.g., `Printf` and `printf` are different

```c
#include <stdio.h>

void main()
{
  int x;
  scanf( "%d", &x );
  printf( "%d\n", x*x );
  return;
}
```
1. Conversion of Input/Output

- **Arrows**:  
  - Nothing necessary to the next statement  
  - Otherwise, convert it to “goto label;”

- **Start/End**:  
  - Start: no conversion  
  - Stop: convert to return;

- **Declaration of variables**:  
  - Convert to int i, j, k;  
  - ‘int’ is a variable type (abbreviation of ‘integer’).  
  - The variable name is made of alphanumerics, but should start with an alphabet.
1. Conversion of Input/Output

- **Input:**
  - `scanf( "%d", &x);`
  - `scanf( "%d%d", &x, &y);`

- **Output:**
  - **Printing numbers:**
    - `printf( "%d\n", x);`
    - `printf( "%d %d\n", x, y);`
  - **Printing characters:**
    - `printf( "abc\n" );`
Example 1

- Read a number and print it

```c
#include <stdio.h>

main()
{
    int x;
    scanf( "%d", &x );
    printf( "%d\n", x );
    return;
}
```
2. Conversion of Conditional (1)

- **Type 1**
  
  ```
  if(condition) {
    A;
    B;
  }
  C;
  D;
  ```

- **Type 2**
  
  ```
  if(condition) {
    A;
    B;
  }
  else {
    C;
    D;
  }
  E;
  ```
2. Conversion of Conditional (2)

- if statement

```
if (condition) {
    A;
    B;
} else {
    C;
    D;
}
E;
```

Things to do if the condition is true

Things to do if the condition is false

Things to do after the if statement
2. Conversion of Conditional (3)

- **Example**

```java
if(condition1 )
{
    if(condition2 )
    {
        C;
    }
    else
    {
        B;
    }
    D;
}
else
{
    A;
}
E;
```
Example 2

- Read a number and print "Yes" if it is 2 or "No"

```
#include <stdio.h>

main() {
    int x;
    scanf( "%d", &x );
    if( x == 2 ) {
        printf( "Yes\n" );
    } else {
        printf( "No\n" );
    }
    return;
}
```
Example 3

- Find the maximum among 3 numbers

```c
#include <stdio.h>

main() {
    int a, b, c;
    if(a > b )
    {
        if(a > c ) {
            printf( "%d\n", a );
        } else {
            printf( "%d\n", c );
        }
    } else {
        if( b > c ) {
            printf( "%d\n", b );
        } else {
            printf( "%d\n", c );
        }
    }
    return;
}
```
2. Conversion of Conditional (4)

Comparison symbols:

- Comparison operators for conditional statements

<table>
<thead>
<tr>
<th>Meaning</th>
<th>C Operators</th>
<th>Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>=</td>
<td>==</td>
<td>a == 2</td>
</tr>
<tr>
<td></td>
<td>!=</td>
<td>a != 3</td>
</tr>
<tr>
<td>&lt;</td>
<td>&lt;</td>
<td>a &lt; 3</td>
</tr>
<tr>
<td>&gt;</td>
<td>&gt;</td>
<td>a &gt; 4</td>
</tr>
<tr>
<td>&lt;=</td>
<td></td>
<td>a &lt;= 5</td>
</tr>
<tr>
<td>&gt;=</td>
<td></td>
<td>a &gt;= 3</td>
</tr>
<tr>
<td>not</td>
<td>!</td>
<td>!(a &lt; 3)</td>
</tr>
<tr>
<td>and</td>
<td>&amp;&amp;</td>
<td>(3 &lt; a) &amp;&amp; (a &lt; 5)</td>
</tr>
<tr>
<td>or</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

3 < a < 5 ?
2. Conversion of Conditional (5)

• Examples: try on your own
  • a is greater than 3 and less than 5
  • a is greater than 3 and not less than 5
  • negation of “A is greater than 3 and not less than 5”
  • a is equal to or greater than 3
  • a is less than 3 or greater than 5
  • a is not equal to 3
  • a is not greater than 3
3. Conversion of Processing

• Processing symbol:
  • Processing symbol can contain many statements.
  • Convert ‘<-’ to ‘=’ (assignment operator)
  • \( a \leftarrow a + 1 \rightarrow a = a + 1; \)

- Arithmetic operators

<table>
<thead>
<tr>
<th>meaning</th>
<th>C operators</th>
<th>examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>+ (addition)</td>
<td>+</td>
<td>a + 2</td>
</tr>
<tr>
<td>– (subtraction)</td>
<td>–</td>
<td>a – 3</td>
</tr>
<tr>
<td>(multiplication)</td>
<td>*</td>
<td>a * 3</td>
</tr>
<tr>
<td>/ (division)</td>
<td>/</td>
<td>a / 4</td>
</tr>
<tr>
<td>Modulo (remainder)</td>
<td>%</td>
<td>a % 5</td>
</tr>
</tbody>
</table>
4. Conversion of Loop

• Type 3
  • Repeat the statements while the condition is true

```c
while (condition)
{
    A;
    B;
}
C;
```
Example 6

Add all integers between 1 and 100

Start

Var $i$, $sum$

$i$ <- 1

$sum$ <- 0

$i$ <= 100

T

$sum$ <- $sum$ + $i$

$i$ <- $i$ + 1

F

$sum$

Stop

#include <stdio.h>
#include <math.h>

main() {
    int $i$, $sum$;
    $i$ = 1;
    $sum$ = 0;

    while ($i$ <= 100 ) {
        $sum$ = $sum$ + $i$;
        $i$ = $i$ + 1;
    }

    printf( "%d", $sum$ );

    return;
}
Example 7

- Add all even integers between 1 and 100

```
#include <stdio.h>
#include <math.h>

main() {
    int i, sum;
    i = 0;
    sum = 0;
    while( i <= 100 )
    {
        if( i % 2 == 0 )
        {
            sum = sum + i;
        }
        i = i + 1;
    }
    printf( "%d", sum );
    return;
}
```
Example 8

- $n$-th term of Fibonacci sequence
- $a_1=1$, $a_2=1$, $a_n = a_{n-1} + a_{n-2}$

```c
#include <stdio.h>
#include <math.h>

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

    scanf( "%d", &n );

    if( (n == 1) || (n == 2) )
    {
        printf( "%d", 1 );
    }
    else
    {
        a = 1;
        b = 1;
        k = 3;

        while( k <= n )
        {
            c = a + b;
            a = b;
            b = c;
            k = k + 1;

            printf( "%d", c );
        }
        printf( "%d", c );
    }

    return;
}
```
Summary

- **Flowchart**
  - A diagram to represent a process a computer needs to perform to solve a problem
  - Easy to understand for human
  - Easy to be converted to programming languages including C

- **Draw a flowchart before coding a program**

- **Lab introduction**
  - In the room #400212