Control Structures and Data Files
- Repetition
- Selection
- Sequence

Use simple control structures to organize the solution to a problem.
Sequence

---

start main

read time

velocity = 0.00001 \cdot time^3 + 0.75995 \cdot time + 181.3566

acceleration = 3 - 0.000062 \cdot velocity^2

print velocity, acceleration

stop main
Repetition

Increment time by 1

Print velocity

Compute velocity

Yes

Loop: While time does not equal 10

No

Yes

No

Time = 0

Loop: While time does not equal 10

Time = 0
Program walkthrough

Program validation and verification

Test each of the error conditions

Generation of test data

Be clear about specifications

Do not trust user; check the data. A=b/c!

Error condition

Which is the best (e.g., faster, memory req.)

A problem can be solved in many different ways

Evaluation of alternative solution

Extras
and/or logical operators

expressions combined with relational

Conditional Expression is composed of

- FALSE (value of 0)
- TRUE (any value > 0) or

evaluated to be

A condition is an expression that can be

Conditional Expressions, so we will first discuss them

Selection and Repetition Structures use

Conditional Expressions
v"i i i i i = b \iff a = b

\( \forall y = x \)  \quad \text{greater than equal to}  \quad \Rightarrow
\( y \geq x \)

\( 0 \Rightarrow x \)  \quad \text{less than equal to}  \quad \Leftarrow
\( y > x \)

\( y < x \)  \quad \text{greater than}  \quad \rightarrow
\( y < x \)

\( 0 = i \vee \)  \quad \text{less than}  \quad \rightarrow
\( y > x \)

\( = i \)

\( \exists x \equiv x \)  \quad \text{equality}  \quad \equiv

\text{Relational Operators}
Examples
<table>
<thead>
<tr>
<th>False</th>
<th>False</th>
<th>True</th>
<th>True</th>
<th>True</th>
<th>True</th>
</tr>
</thead>
<tbody>
<tr>
<td>True</td>
<td>False</td>
<td>True</td>
<td>False</td>
<td>False</td>
<td>False</td>
</tr>
<tr>
<td>False</td>
<td>True</td>
<td>True</td>
<td>False</td>
<td>False</td>
<td>False</td>
</tr>
<tr>
<td>False</td>
<td>True</td>
<td>True</td>
<td>False</td>
<td>False</td>
<td>False</td>
</tr>
<tr>
<td>True</td>
<td>True</td>
<td>True</td>
<td>False</td>
<td>False</td>
<td>False</td>
</tr>
<tr>
<td>False</td>
<td>False</td>
<td>True</td>
<td>A &amp;&amp; B</td>
<td>A &amp;&amp; B</td>
<td>A &amp;&amp; B</td>
</tr>
<tr>
<td>B</td>
<td>B</td>
<td>B</td>
<td>B</td>
<td>B</td>
<td>B</td>
</tr>
</tbody>
</table>

```

\[(x > 0) \lor (0 < x) \lor (x = 0) \land (x > 0) \land (x = 0) \]

Logical Operators
```
If one side of $a$ is false, the whole expression is false.

\[ a = 3, \quad b = 5, \quad t = 4 \]

\[ a + b > 2 \quad \text{and} \quad |t - 2| > 10 \]

**Examples**
<table>
<thead>
<tr>
<th>Associativity</th>
<th>Operation</th>
<th>Precedence</th>
</tr>
</thead>
<tbody>
<tr>
<td>Right to left</td>
<td>= % /= =* == += =</td>
<td>6</td>
</tr>
<tr>
<td>Left to right</td>
<td></td>
<td>8</td>
</tr>
<tr>
<td>Left to right</td>
<td></td>
<td>7</td>
</tr>
<tr>
<td>Left to right</td>
<td></td>
<td>6</td>
</tr>
<tr>
<td>Left to right</td>
<td></td>
<td>5</td>
</tr>
<tr>
<td>Left to right</td>
<td></td>
<td>4</td>
</tr>
<tr>
<td>Right to left (unary)</td>
<td></td>
<td>3</td>
</tr>
<tr>
<td>Innermost first</td>
<td>( )</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>++ - + % / *</td>
<td>1</td>
</tr>
</tbody>
</table>

Relational, and Logical Operators
Precedence for Arithmetic
Exercise
Selection Statements
Selection Statements

- switch
- if else
- if
{ 
statement n; 
statement; 
more than one statement; } 
if(Boolean expression) 
if(Boolean expression) 
if(Boolean expression) 
If statement; // single statement 
If statement; // single statement
### Examples

```c
{ 
  \text{if} (x > 0) 
  \{ 
    ++x 
    x = \text{sqrt}(x) 
  \} 
  \text{if} (x < 0) 
  \{ 
    ++k 
  \} 
}
```
if (boolean expression) {
  statement block1
} else {
  statement block2
}

if (boolean expression) {
  x = y;
} else {
  k = 3;
}

if - else statement
temp = y;
else
    temp = x;
if (x < y)
    # That x, y, temp are declared.
What does the following program do? Assume

If-else statement
Split the following statement into two separate if-else statements:

```
if (x<y)
    temp = x;
else
    temp = y;
```
Exercise

Write an if-else statement to find both the maximum and minimum of two numbers. Assume that x, y, min, max are declared.

\{
    \begin{align*}
    \text{min} &= x; \\
    \text{max} &= y;
    \end{align*}
\}

\text{else}
\{
    \begin{align*}
    \text{min} &= y; \\
    \text{max} &= x;
    \end{align*}
\}

\text{if} (x < y) - \text{If equal, max and min will be the same.}

\text{Ex: } x = 10, y = 5

9 = y, \ y = 6,
4 = \ y, \ y = 3,
nested if-else
After execution of if statement?
What are the values of \( j \), \( k \) and \( m \)?

<table>
<thead>
<tr>
<th>( j )</th>
<th>( m )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

\[
\begin{align*}
\text{else} & \quad m++ \\
\text{else} & \quad k++ \\
( z > \bar{\lambda} ) & \quad \text{if} \\
( \bar{\lambda} < x ) & \quad \text{if} \\
\text{int} \ x=9, \bar{\lambda}=7, z=2, k=0, m=0, j=0.
\end{align*}
\]

Exercise
After execution of if statement, what are the values of $j$, $k$, and $m$?

<table>
<thead>
<tr>
<th>$j$</th>
<th>$m$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

$++! \quad j = 1$

$\begin{align*}
& z \quad w \\
& 2 \quad 2
\end{align*}$

$++! \quad k$

$(z > k) \quad \forall (x < k) \quad \forall (x = 3) \quad \forall (x) = 3$

Exercise
What is the output of the following program?

```c
#include <stdio.h>

int main() {
    int a = 5, b = 3;
    printf("a = %d, b = %d\n", a, b);
    
    if (a > 10) {
        b = 20;
    }
    printf("b = %d\n", b);
    
    if (a > 10) {
        a = 50;
    }
    printf("a = %d\n", a);
    
    if (a > 10) {
        printf("a = %d, b = %d\n", a, b);
    }
    
    b = 20;
    printf("b = %d\n", b);
    
    if (a > 10) {
        printf("a = %d, b = %d\n", a, b);
    }
    
    return 0;
}
```
<table>
<thead>
<tr>
<th>Grade</th>
<th>Score Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>D</td>
<td>0-39</td>
</tr>
<tr>
<td>C</td>
<td>40-59</td>
</tr>
<tr>
<td>B</td>
<td>60-79</td>
</tr>
<tr>
<td>A</td>
<td>80-100</td>
</tr>
</tbody>
</table>

Program to find the corresponding grade.

Given a score and the following grading scale write a function to return the corresponding letter grade.
print("Invalid score\n")
else
    grade = 'D'!
else if (score >= 0) 
    grade = 'C'!
else if (score >= 40) 
    grade = 'B'!
else if (score >= 60) 
    grade = 'A'!
    else if (score >= 80) 
        grade = 'A'!

Solution
```python
print("Invalid Score")
else
grade = 'D'
else if (score <= 40)
grade = 'C'
else if (score <= 60)
grade = 'B'
else if (score <= 80)
grade = 'A'
else if (score > 80)
print("Invalid Score")
```

**Alternate Solution**
Exercise

What is the value of `a` at the end of the following if-else statement (Chapter 3.2)?

```
if (a > 500) {
    a = 10;
} else if (a > 750) {
    a = 2;
} else if (a > 1000) {
    a = 0;
} else {
    a = 3;
}
```
minimum = 2, maximum = 3, median = 3

\[ a = 2, b = 2, c = 3 \]

minimum = 2, maximum = 5, median = 3

\[ a = 2, b = 5, c = 3 \]

Example:

Example: median of the numbers.

from user and computes minimum, maximum and

Write a program that reads 3 numbers a, b and c.

Exercise
\[
\begin{align*}
\text{if } (a > b) \text{ and } (c > b) \\
\text{else if } (c > a) \text{ and } (c > b) \\
\text{else } \text{if } (b > a) \text{ and } (b > c) \\
\text{else } \text{if } (a > b) \text{ and } (a > c) \\
\text{min} = a'
\end{align*}
\]
max = c'

else if (c > a) && (c > b)

max = c

else if (b > a) && (b > c)

max = b

else

max = a'

if (a > b) && (a > c)

Maximum
The median is defined as:

\[
\text{median} = \begin{cases} 
  c & \text{if } (a \geq c) \text{ and } (c \geq b) \\
  a & \text{if } (b \geq c) \text{ and } (c \geq a) \\
  a & \text{if } (c \geq a) \text{ and } (a \geq b) \\
  b & \text{if } (b \geq a) \text{ and } (a \geq c) \\
  b & \text{if } (c \geq b) \text{ and } (b \geq c) \\
\end{cases}
\]
median = c'
  elseif (b > c) REM (c > a)
    median = a;
  elseif (b > a) REM (a > c)
    median = b;
  else
    if ((a > b) REM (b > c))
      median = a;
    elseif ((c > a) REM (a > b))
      median = b;
  end

Alternate Solution for Median
Find median of 3 numbers

Similarly find maximum of 3 numbers

Combine T-3 into a single nested if-else statement

Write an if statement to check if minimum is a

What should

be condition?

\( \text{min} = a \)

if (condition)

Write an if statement to check if minimum is a

Find minimum of 3 numbers

Exercise continued
Point in Region 3
Enter x', y': -1 - 5

Point in Region 4
Enter x', y': -1 - 1

For example

user and prints the region it resides in.

Write a program that reads a point (x', y') from

Exercise
Solution
Alternate Solution
Switch Statement

```plaintext
switch (expression)
{
    case constant:
        statement(s);
        break;
    case constant:
        statement(s);
        break;
    /* default is optional */
    default:
        statement(s);
}
```
Switch Statement

- To be executed.
- The keyword `break` statement is required unless you want all subsequent statements to be executed.
- The keyword `case` must be followed by a constant expression must be of type integer or character.
```java
switch (op_code)
{
    case 'M':
        break;
    case 'N':
        break;
    default:
        break;
    print("Error\n");
    else if (op_code == 'M')
        print("Maintenance Needed\n");
    else if (op_code == 'N')
        break;
    print("Normal\n");
}
```

**Exercise:** Convert the switch statement into if statement.
to a switch statement

Convert the following nested if/else statements

```
{ 
  if (rank==5) 
    printf("Invalid rank\n");
  else if (rank==4) 
    printf("Graduate student\n");
  else if (rank==3) 
    printf("Upper division\n");
  else if (rank==2) 
    printf("Lower division\n");
}
```
```c
switch (rank) {
    case 1:
        printf("Lower Division\n");
        break;
    case 2:
        printf("Upper Division\n");
        break;
    case 3:
        printf("Graduate Student\n");
        break;
    default:
        printf("Invalid Rank\n");
        break;
}
```
January, March, May, July, August, October, December has 31 days.
April, June, September, November has 30 days.
February has 28 days.
1-12 represents months January – December.
switch (month)
    case January:
    case March:
    case May:
    case July:
    case August:
    case October:
    case December:
        System.out.println(31); // 31 days
    default:
        System.out.println(30); // 30 days
    }

Write a switch statement that prints the number of days in the month given an integer corresponding to month.
```c
{ 
    break;
    print("Invalid month\n");
    default:
    break;
    print("30 days\n");
    case 4: case 6: case 9: case 11:
    break;
    print("31 days\n");
    case 1: case 3: case 5: case 7: case 8: case 10: case 12:
    break;
    print("28 days\n");
    case 2:
}
switch(mouth)
```

Solution
Loop (Repetition) Structures
Loop (repetition) statements

- for statement
- do while statement
- while statement
```plaintext
while x > 5;
  x = x + 1;
  print x;=\"\"x\"\";%d;\"\",
  x = x + 1;
end
```

**While Statement**

```
while expression
  statement
  statement
  while expression
    statement
    statement
end
```

**While Statement**
```c
#include <stdio.h>

#define PI 3.141593

Example: Lecture24.c
```
Once the expression is tested after the statement(s) are executed, so statements are executed at least once.

```
while (x > 5) {
  printf("x = ", x);
  x = x + 1;
}
do while x = x
```

```
while (expression) {
  \begin{itemize}
    \item statement 1
    \item statement 2
  \end{itemize}
}
do while expression
```
#include <stdio.h>

#define PI 3.141593

int main(void)
{
    int degrees = 0;
    double radians = 0;

    do
    {
        printf("Degrees to Radians \n\n");
        radians = degrees * PI / 180;

        printf("%.6f \n", radians);

        degrees += 10;
    } while (degrees <= 360);

    return 0;
}
\begin{align*}
\text{for}\ (x \geq 5; x = x + 1; \text{test}\ x > 5) \\
\text{for}\ (x = 1; x \leq 2; \text{test}\ x \text{ is even}) \\
\text{for}\ (x \geq 5; x = x + 1; \text{test}\ x > 5) \\
\text{for}\ (x = 1; x \leq 2; \text{test}\ x \text{ is even})
\end{align*}
for statement

Test

Initialize

Increment/Decrement

Statement(s)

True
Examples

```c
int fact = 1;
for (n = 5; n > 1; n--)
    fact *= n;

int fact = 1;
int sum = 0;
for (i = 1; i <= 2; i++)
    sum += i;

int sum = 0;
```
```c
{ return 0;
 { printf("\n\n", "degrees", radians);
 radians = degrees * PI / 180;
 }
 for (degrees = 0; degrees <= 360; degrees += 10)
 printf("\n\n", "Degrees to Radians \n\n");
 double radians;
 int degrees;
 }
 int main(void)

#define PI 3.141593
#include <stdio.h>
```
Determine the number of times that each of the following loops are executed.

Exercise
Hello 2
Hello 1
Hello

Change it to print the count besides each hello

Change it to print Hello 20 times

Write a loop to print Hello 10 times

Exercise
Result = 45
Result = 55

Write a program to compute the following using a for loop.

Exercise
Exercise

Convert the following for loop to while loop

```c
int i = 5;
while (i < 10)
{
    printf("i = \%d \n", i);
    i++;
}
```

Initialization must be before loop.
Write a loop to compute $x^y$ for integer $y$ without using the `pow()` function.
Write a program to compute the following:

\[ \sum_{m=0}^{\infty} x^m + \cdots + x^4 + x^3 + x^2 + x + 0 x = \prod_{m=1}^{\infty} (1 - x^m) \]
\[ n \quad \ldots \quad \frac{7}{2} + \frac{6}{3} + \frac{5}{4} + \frac{2}{1} = \frac{1}{1} \]

Write a program to compute the following:

\[ In_2 = \text{In}_2 + 1.0 / 1 \]

\[ \text{else} \]
\[ In_2 = \text{In}_2 - 1.0 / 1 \]

\[ \text{if} (\text{!} \% 2 == 0) \quad \rightarrow \text{check if even} \]
\[ \text{for} (\text{!} = 1; \text{!} >= 0; \text{!}++) \]
\[ \text{In}_2 = 0. \]

Exercise
Exercise

What is the output of the following program?

Available on class webpage: lecture27.c

```c
{ }

printf("\n");

for (i=1; i>=4; i++)
{
    printf("\n");
    printf("*");
}

for (i=1; i>=5; i++)
{
    moves to next line
    only next line part
    no parentheses
    for (i=1; i>=4; i++)
    (++)
    for (i=1; i>=5; i++)
    (++)
    for (i=1; i>=5; i++)
    (++)
    for (i=1; i>=5; i++)
    (++)
    for (i=1; i>=5; i++)
    (++)
    for (i=1; i>=5; i++)
    (++)
    for (i=1; i>=5; i++)
    (++)
    for (i=1; i>=5; i++)
    (++)
    for (i=1; i>=5; i++)
    (++)
    for (i=1; i>=5; i++)
    (++)
    for (i=1; i>=5; i++)
    (++)
    for (i=1; i>=5; i++)
    (++)
    for (i=1; i>=5; i++)
    (++)
    for (i=1; i>=5; i++)
    (++)
    for (i=1; i>=5; i++)
    (++)
    for (i=1; i>=5; i++)
    (++)
    for (i=1; i>=5; i++)
    (++)
    for (i=1; i>=5; i++)
    (++)
    for (i=1; i>=5; i++)
    (++)
    for (i=1; i>=5; i++)
    (++)
    for (i=1; i>=5; i++)
    (++)
    for (i=1; i>=5; i++)
    (++)
    for (i=1; i>=5; i++)
    (++)
    for (i=1; i>=5; i++)
    (++)
    for (i=1; i>=5; i++)
    (++)
    for (i=1; i>=5; i++)
    (++)
    for (i=1; i>=5; i++)
    (++)
    for (i=1; i>=5; i++)
    (++)
    for (i=1; i>=5; i++)
    (++)
    for (i=1; i>
Available on class webpage: Lecture28.c

```c
{
    printf("\n\n");
    printf("\n");
    for (j=1; j>=1; j++)
        for (i=1; i>=5; i++)
            Output Program? What is the output of the following
```

Exercise
Exercise

- Modify the following program to produce the output.

```c
for (i=A; i<=B; i++)
    {
        for (j=C; j<=D; j++)
            printf("\*" );
        printf("\n") ;
    }
```

Output

```
*****
****
***
**
* 
```
produce the output
Write a program using loop statements to

Exercise
Available on class webpage: Lecture29.c

```c
printf("Sum = \n\n", sum);
{
    x+=sum
    break
    if (x < 10.0)
        scanf("\n\n", &x);
}
for (k=1; k>=5; k++)
    sum = 0;
```

Loop execution continues with the first statement following the break statement.
continue statement

- continue;
  - forces next iteration of the loop, skipping any remaining statements in the loop

```c
sum = 0;
for (k=1; k<=5; k++)
{
    scanf("%lf",&x);
    if (x > 10.0)
    continue;
    sum +=x;
}
printf("Sum = %.1f \n",sum);
```

- Available on class webpage: lecture2e10.c
Consider the following example:

Files are stored on disk.

Write output of a program to a file.

Read input of a program from a file.
Each data file must have a file pointer.

**Data Files**
I/O Statements

- Output file - use fprintf instead of print
- fprintf(printf, "%f %f\n", time,
  height, velocity);
- fprintf(balloon, "%f %f\n", time,
  fill, pointer);
- scanf(sensor, "%f %f\n", &a, &motion);
- scanf(file, "%d", &b);
- input file - use fscanf instead of scanf
Teo.f (fileptr) > 0 when EOF reached
while loop

When file is created EOF is inserted
end of file controlled loop
for loop

First line in file contains count
counter controlled loop

Reading Data Files
Available on webpagae as scores2.txt

FILE

Available on webpagae as lecture2.txt

Counter controlled loop

Usually first line in file contains the count

RES

close(scorefile)
{
    printf("%d\n", score);
    fscanf(scorefile, "%d", &score);
    read item from file and
    for (i=1; i>count; i++)
        fscanf(scorefile, "%d", &count);
    scorefile = fopen("scores2.txt", "r");
    int score, count;
    FILE *scorefile;
    FILE *scorefile;
    int main()
    
#include <stdio.h>
```c
#include <stdio.h>

FILE *scorefile;
int score;

FILE *openfile = fopen("scores.txt", "r");
if (openfile == NULL) {
    printf("Error opening file!
");
    return (0);
}

FILE *scorefile = fopen("scores.txt", "r");
if (scorefile == NULL) {
    printf("Error opening file!
");
    return (0);
}

int main() {
    while (fget(scorefile) >= 0) {
        /* Read each line */
        if (fget(scorefile) == EOF) {
            printf("End of file!
");
            break;
        }
        /* Process the line */
    }
    return 0;
}
```

End of file controlled loop
Solution available on webpage as Lecture213.c

```
File
63
85
24
93
78
56

open file
Algorithm to find minimum in a file
//
number in a file;
Given a file of integers, write a program that finds the minimum

close file
display the minimum
min = x
if (x > min)
read next number x from file

set minimum to a large value
```
Exercise
Solution available on webpage as Lecture25.c

```c
{
    printf("outfile, " output, score);
    fscanf(scorefile, "d, &score)

    while (fread(scorefile, ) == 0)
    
    outfile = fopen("newscores.txt", "w");
    scoresfile = fopen("scores.txt", "r");
    int score;
    FILE *outfile;
    FILE *scoresfile;
}
```

Read from one file and write to another

Exercise
Exercise

Write a program that reads a number and prints all the numbers from 1 to n to file "numbers.txt".