From Algorithms to Program

Marie Yvette B. de Robles
Institute of Computer Science
University of the Philippines Los Baños

Objectives

At the end of the meeting, students should be able to:

- Create programs using the different operations on variables: assignment, arithmetic, comparison
- Identify the three types of loops.
- Create programs with selections and iterations.
- Create programs with nested loops.

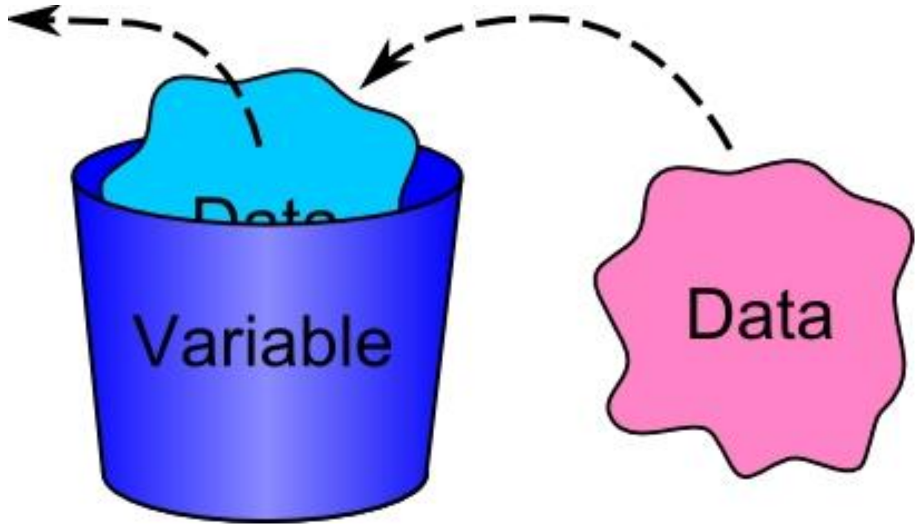
From Algorithms to Program

A typical programming task can be divided into two phases:

- ***Problem solving phase***
  - produce an ordered sequence of steps that describe solution of problem
  - this sequence of steps is called an ***algorithm***
- ***Implementation phase***
  - implement the program in some programming language

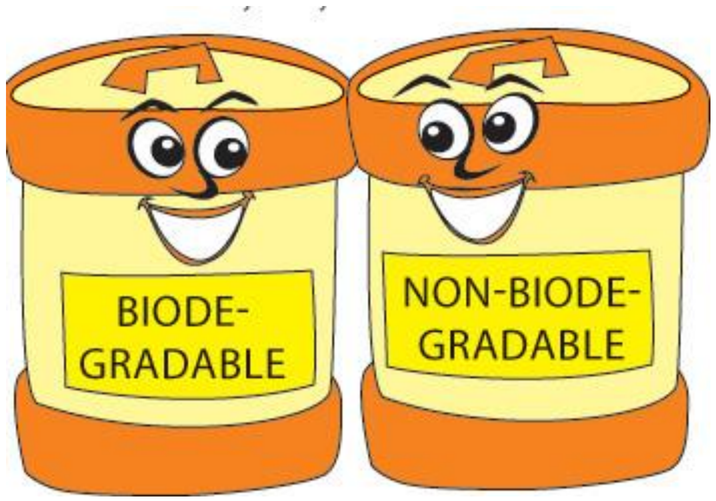
Variables

- variables are used for the temporary storage of values in the computer's memory



Variables and their Types

- all variables are declared in a program along with their types
- most commonly used types are integers **(int)**, floating point numbers with decimal points (**float)**, and characters **(char)**



Syntax

- basic syntax or format for variable declarations

<type> <one or more variables separated by commas>;

- Examples:

  int age;

  float inches, cm;

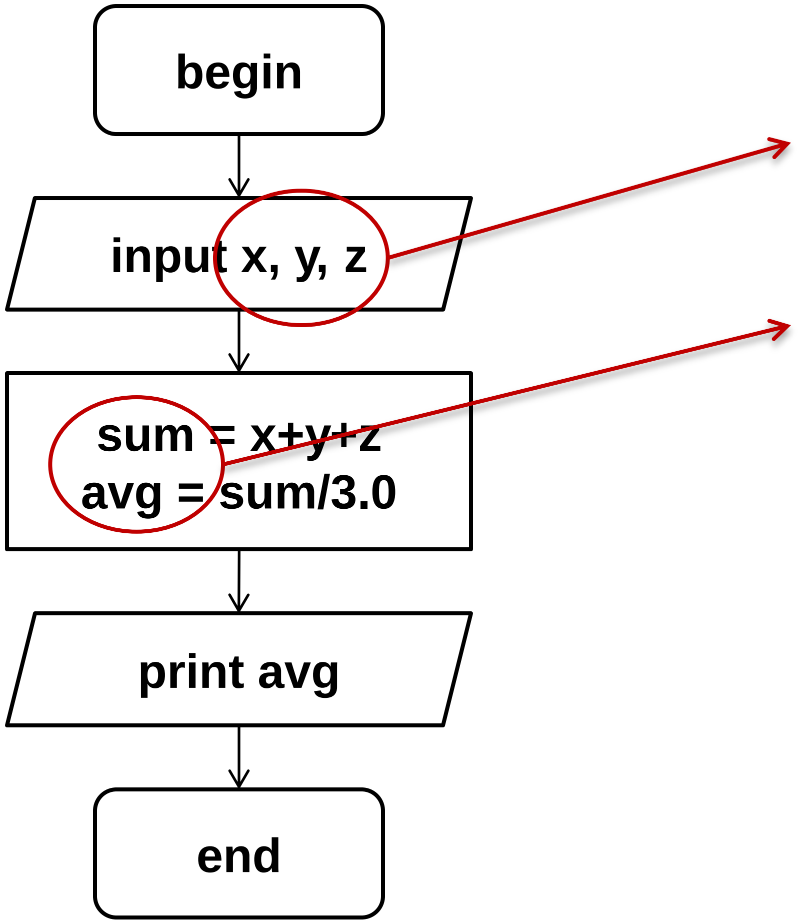
  char middle_initial;

Variables and their Types

- We can combine these basic types to form more complex types, e.g., a list of integers, or a string of characters

- Examples:

```
int quizzes[5];
        /* up to five integers */
char firstname[20], surname[20];
        /* up to 20 characters  long */
```



int x, y, z;
int sum;
float avg;

Operations on variables: assignment

- Data can be stored (and later retrieved) in variables

  Syntax:

  ```
  <variable> = <value>
  ```

  Examples:

  int x = 10;
  float weight = 47.2;
  char middle_initial = ‘b’;

Operations on variables: assignment

```
#include<stdio.h>
main()
{
    int x=10;
    printf("%d", x);
}
```

x $\boxed{10}$

Stores 10 in the variable named x, then prints the contents of x

%d is the format code for an integer

Operations on variables: arithmetic

- Basic arithmetic (add +, subtract -, mult *, divide /, remainder %) can be performed

```
#include<stdio.h>
main()
{
    int x=10;
    printf("%d %d %d %d %d",
            x+2, x-2, x*2, x/2, x%2);
}
```

Operations on variables: arithmetic

- Arithmetic expressions can be used in the right side of assignment statements

```
main()
{
    int x = 10, y, z;
    y = (2*x)+1;
    z = 2*(x+1);
    x = x+1;
    printf("%d %d %d", x, y, z);
}
```

Operations on variables: arithmetic

- Arithmetic expressions can be used in the right side of assignment statements

```
main()
{
    int x = 10, y, z;
    y = (2*x)+1;
    z = 2*(x+1);
    x = x+1;
    printf("%d %d %d", x, y, z);
}
```

1. Compute Right Side

```
   (2*x)+1
 = (2*10)+1
 = 21
```

Operations on variables: arithmetic

- Arithmetic expressions can be used in the right side of assignment statements

```
main()
{
    int x = 10, y, z;
    y = (2*x)+1;
    z = 2*(x+1);
    x = x+1;
    printf("%d %d %d", x, y, z);
}
```

2. Assign right(value) to left(variable)
   y = 21;

Operations on variables: arithmetic

- Arithmetic expressions can be used in the right side of assignment statements

```
main()
{
    int x = 10, y, z;
    y = (2*x)+1;
    z = 2*(x+1);
    x = x+1;
    printf("%d %d %d", x, y, z);
}
```

1. Compute Right Side

```
    x+1
  = 10+1
  = 11
```

Operations on variables: arithmetic

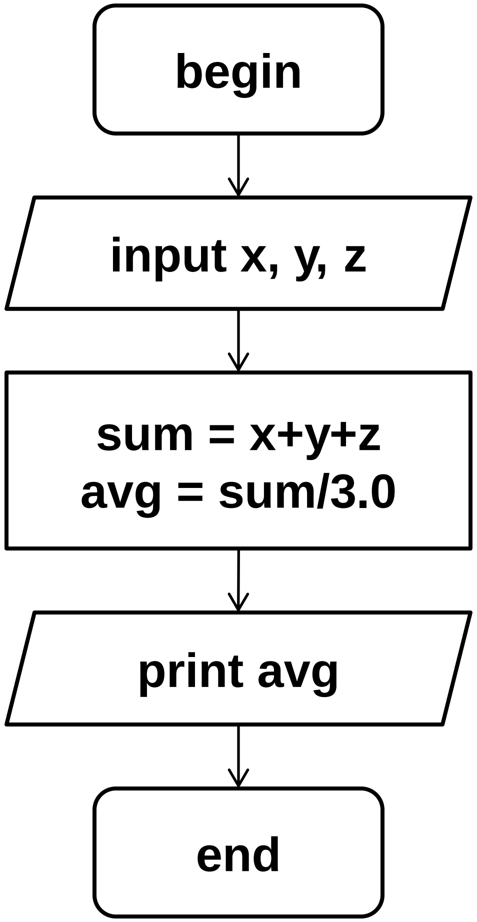
- Arithmetic expressions can be used in the right side of assignment statements

```
main()
{
    int x = 10, y, z;
    y = (2*x)+1;
    z = 2*(x+1);
    x = x+1;
    printf("%d %d %d", x, y, z);
}
```

2. Assign right(value) to left(variable)
   x = 11;

Example 1:

- Given 3 numbers in any order, find their average.



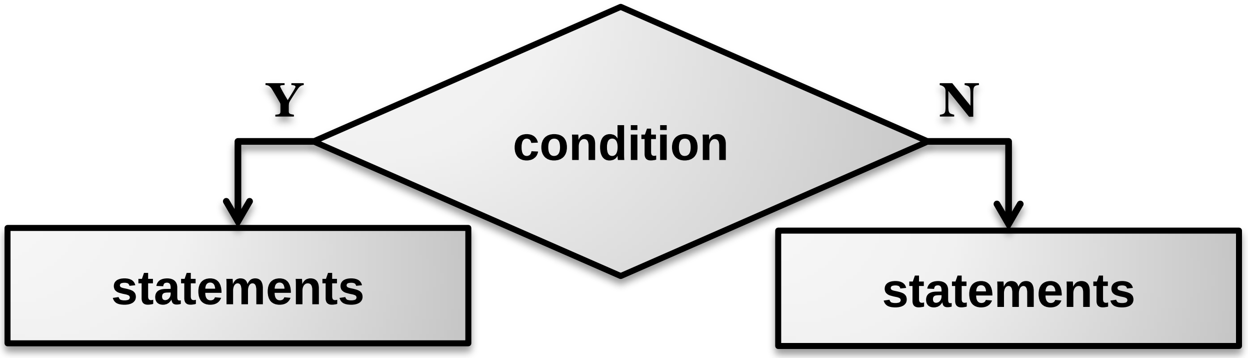
```
int x, y, z, sum;
float avg;

scanf(“%d %d %d”, &x, &y, &z);

sum = x+y+z;
avg = sum/3.0;

printf(“%f”, avg);
```

%f is the format code for a float

Operations on variables: comparisons

Syntax: (note: the else clause is optional)

```
if (condition) {
    statements to be performed if the condition is true;
}
else {
    statements to be performed if the condition is false;
}
```

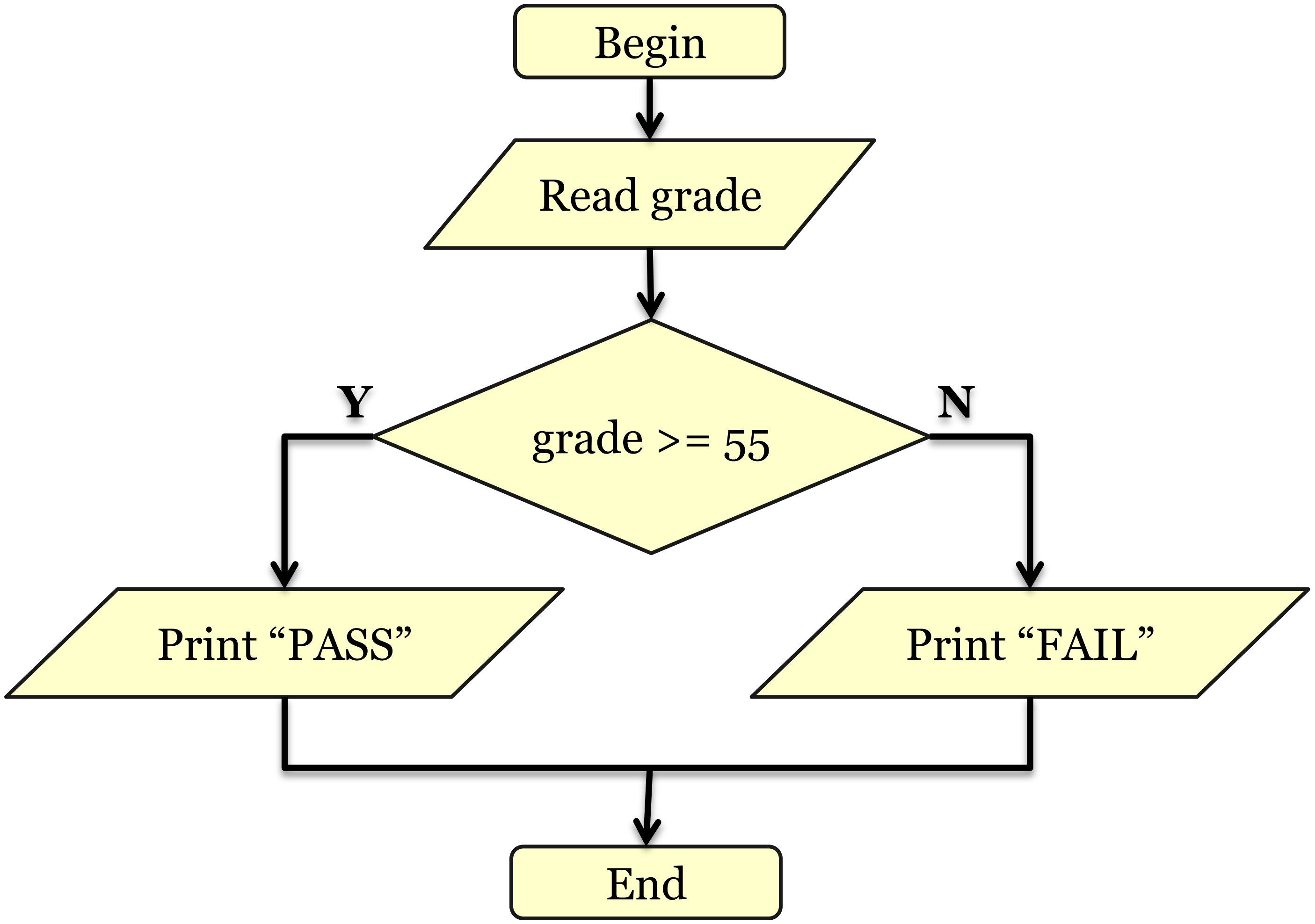


Operations on variables: comparisons

- a condition is a logical (or Boolean) expression which evaluates to either true or false;
- relational operators are often used for comparing values of expressions

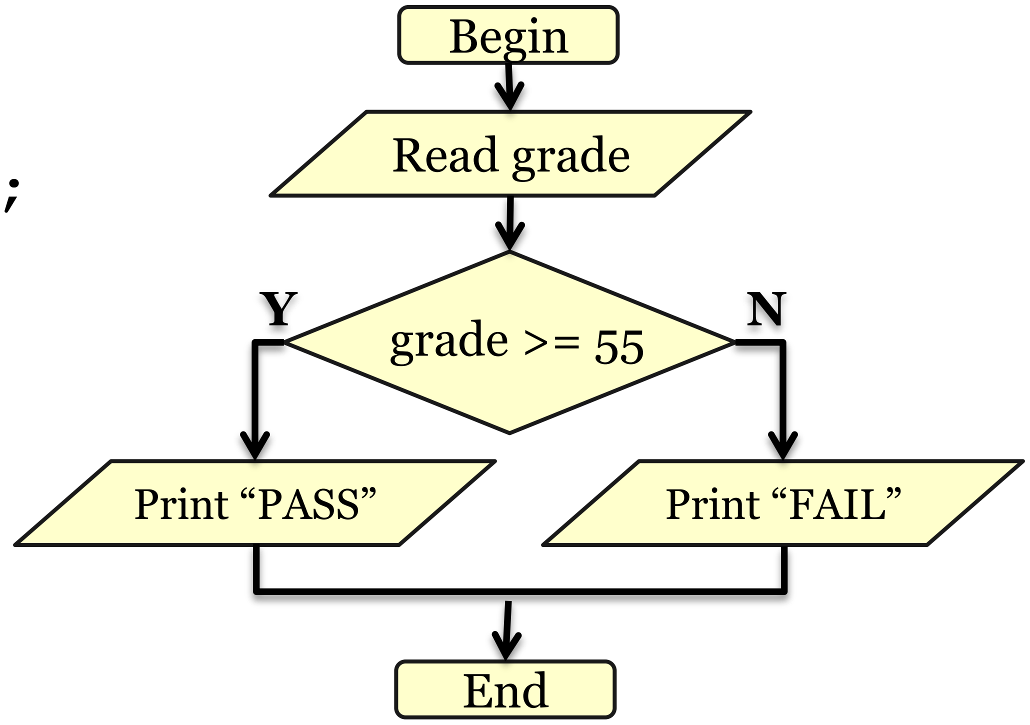
==	equal
<	less than
<=	less than or equal
!=	not equal
>	greater than
>=	greater than or equal

Example 2:



Example 2:

```
float grade;

scanf(“%f”, &grade);

if(grade >= 55){
 printf(“Pass\n”);
}
else{
 printf(“Fail\n”);
}
```



Conditions can be simple, or complex with the use of logical operators

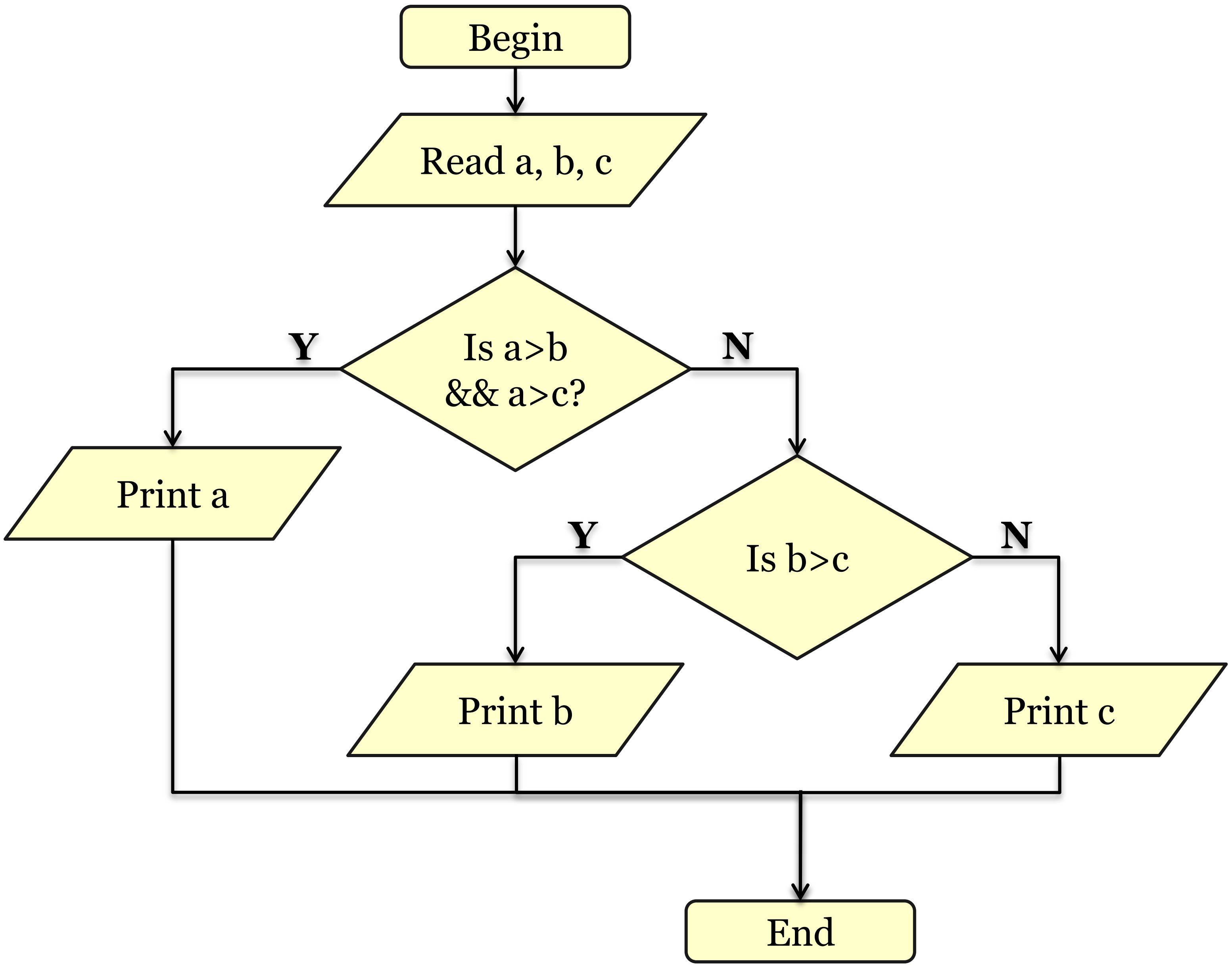
- **!** Means **NOT**
  - ***(!A) is true*** if and only if ***A is false***
- **&&** means **AND**
  - ***(A && B) is true*** if and only if ***both A and B are true***
- **||** means **OR**
  - ***(A || B)*** is true if and only if ***at least one of A or B is true***

A	B	A && B	A || B
T	T	T	T
T	F	F	T
F	T	F	T
F	F	F	F

A	!A
T	F
F	T

Example 3:

- Input any 3 numbers (in random order), and find and print the largest value.


Begin
Read a, b, c
Is a>b && a>c?
Y
N
Print a
Is b>c
Y
N
Print b
Print c
End

Example 4:

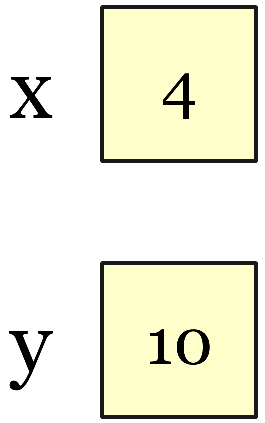
Enter a temperature in Fahrenheit, convert and print the equivalent temperature in Celsius, and output exactly one of the following messages: “too cold” (< 10C), “too hot” (> 40C), or “just right” (greater that or equal to 10C but less than or equal to 40C).

Example 5:

Input any 2 numbers (in random order), and print them in sorted (ascending) order.

How do you swap two numbers?

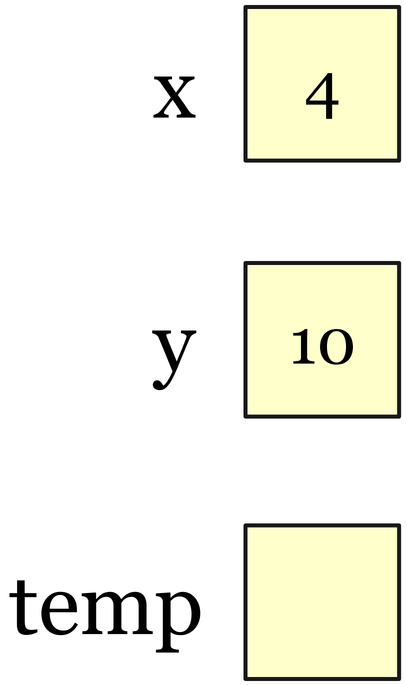
```
x = y;
y = x;
```



What will be the value of `x` and y after the two statements?

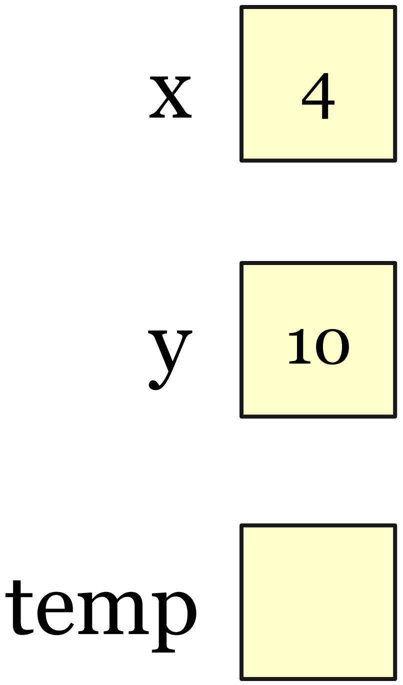
How do you swap two numbers?

We need another variable.



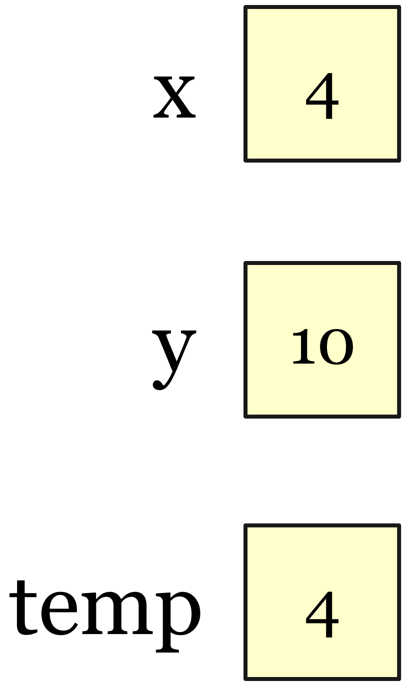
How do you swap two numbers?

before `x = y`,
we save the value
of x to temp
(`temp = x`)



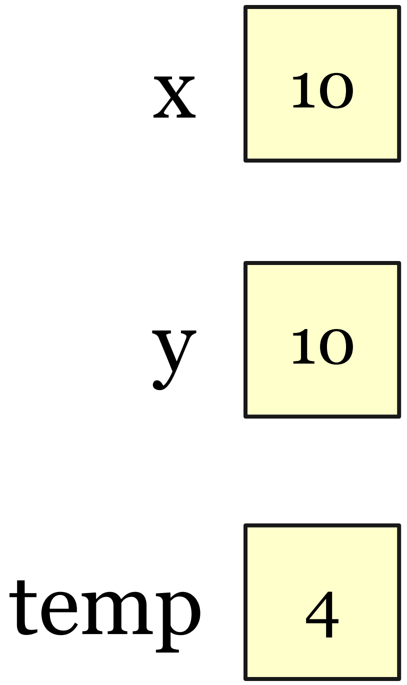
How do you swap two numbers?

before `x = y`,
we save the value
of x to temp
(`temp = x`)



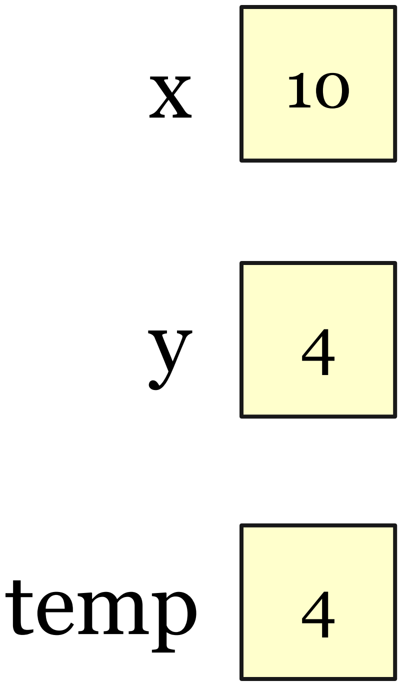
How do you swap two numbers?

```
temp = x
x = y
```



How do you swap two numbers?

```
temp = x
x = y
y = temp
```



Example 6:

Input any 3 numbers (in random order), and print them in sorted (ascending) order.

Hint: One possible algorithm is to do the ff.

{ *sort the first adjacent pair; sort the last adjacent pair; sort again the first adjacent pair;* }

Programming tips

- Use **meaningful** variable names to help document your programs:
  - **x, y, z** are valid names but they do not mean much. **Fahrenheit, Celsius** and **age** in our examples are better names
  - In C, variable names must start with a letter and may be followed by more letters, digits, or underscore
  - C is **case-sensitive** so be careful when you type: **age**, **Age**, **AGE**, and **aGe** can all represent different variables/memory locations

Programming tips

- Improve program layout
  - Use indentation to indicate which parts of the code go together (e.g., statements in an if-branch block should all be indented together)
  - Add extra spaces, extra lines to avoid crowding
  - Use English comments to help explain unclear code **/* comment */ or // comment**

Loops in programs

- **Types of loops**

  - Do-while loops, while-loops, for-loops

  - Loops with break statements
- **Examples and more examples**

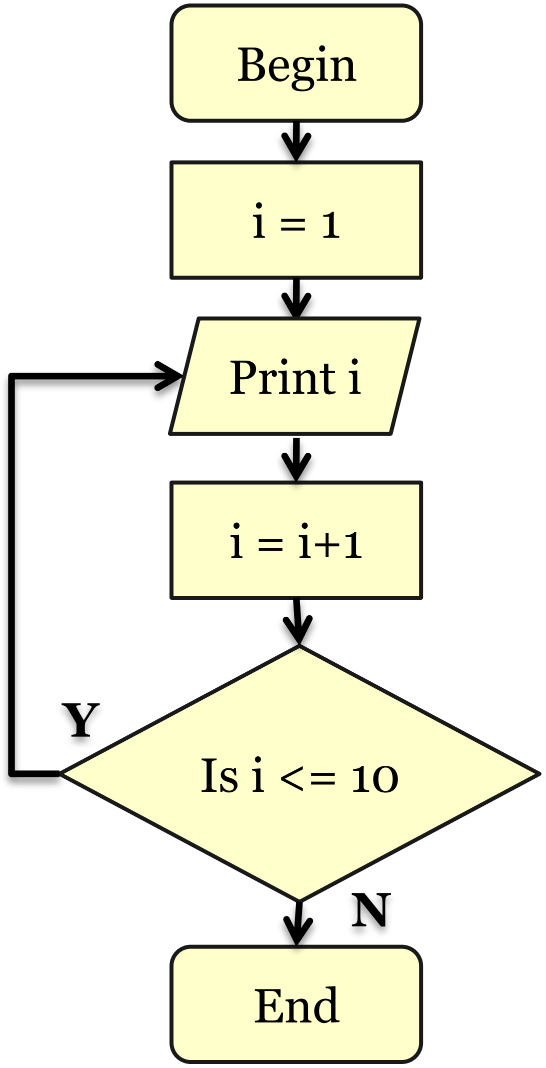
  - Numerical and non-numerical applications
- **Structured programming**

  - Branches inside loops, loops within loops, etc.

Types of Loops

- **do-while** loops (test-condition-at-the-**end**)

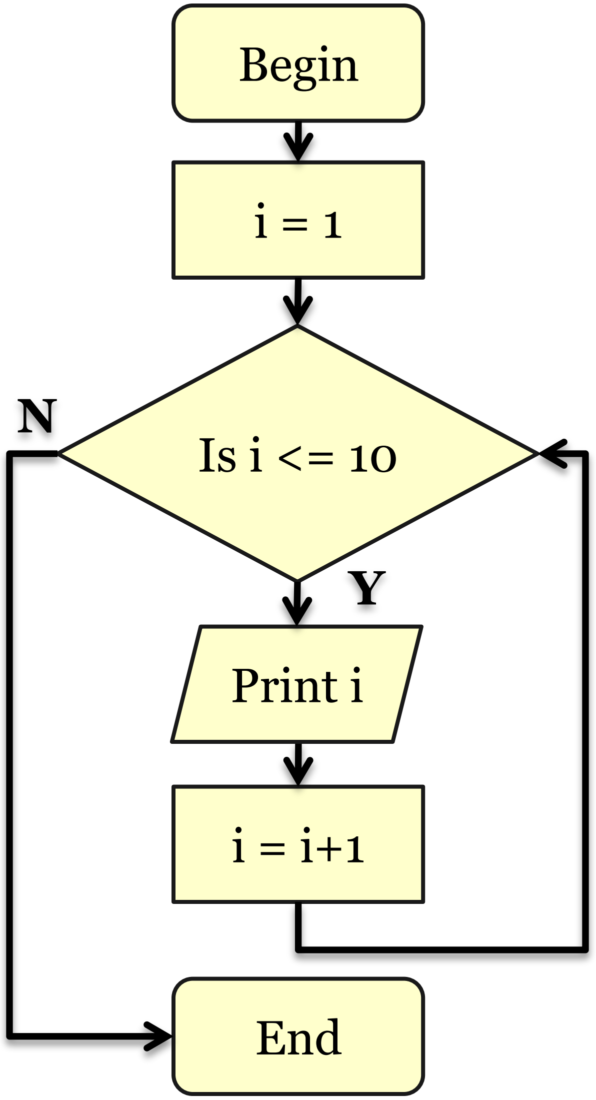
```
int i;

i = 1;
do{
    printf("%d\n", i);
    i = i+1;
}while(i<=10);
```



Types of Loops

- **while** loops (test-condition-at-the-**start**)

```
int i;

i = 1;
while(i<=10){
    printf("%d\n", i);
    i = i+1;
}
```

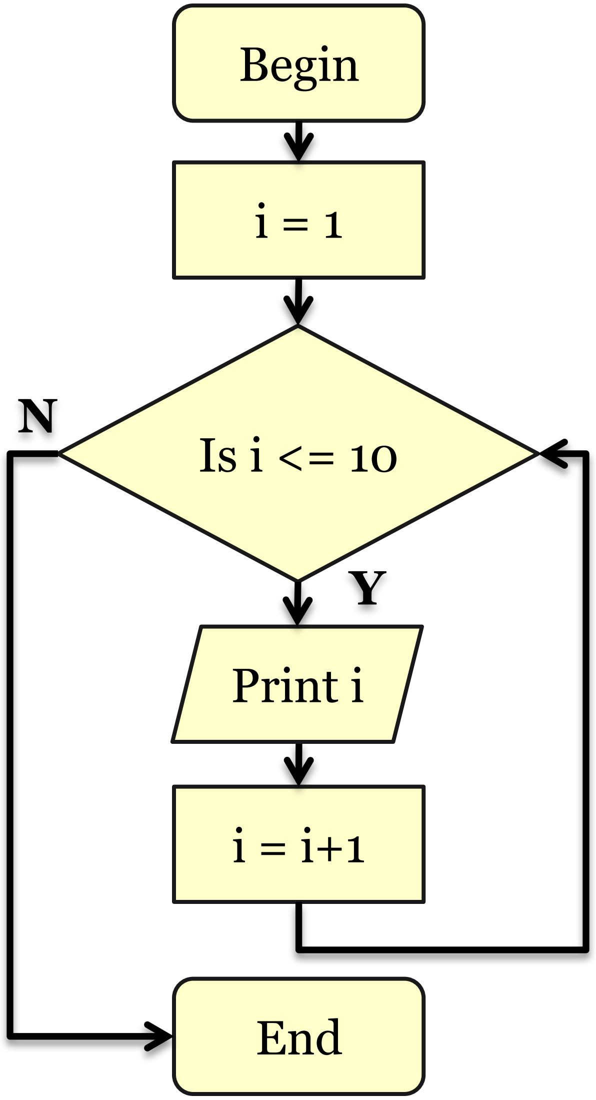


Types of Loops

- **for** loops

  Syntax:
  for (initialization; condition; step) {
          // statements in the body of the loop
  }

```
int i;

for(i=1; i<=10; i++){
    printf("%d\n", i);
}
```



Avoid infinite loops

- All algorithms must terminate, hence programs should always have a way to get out of loops
- Example of an infinite loop (use control-c to escape from some infinite loops)

```
do {
    printf("makulit daw ako....");
} while ( 1+1 == 2 ); // always true
```

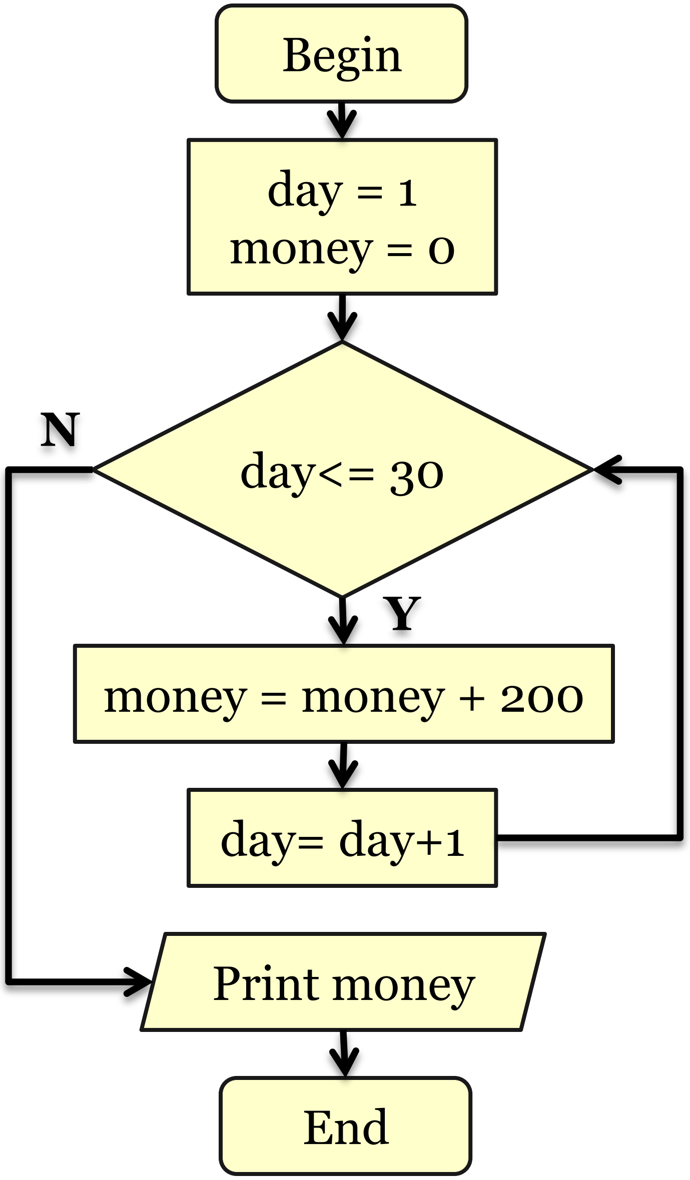
Problem 1

- Suppose a rich uncle offers you three allowance plans for **30 days.**
  - **Plan A**: P200/day for 30 days
  - **Plan B**: P1 on the 1st day, P4 on the 2nd day, P9 on the 3rd day, … (n^2 pesos on the nth day)
  - **Plan C**: P15 on the 1st day, P30 on the 2nd day, P45 on the 3rd, … (15n pesos on the nth day)

Which plan would allow you to earn more?

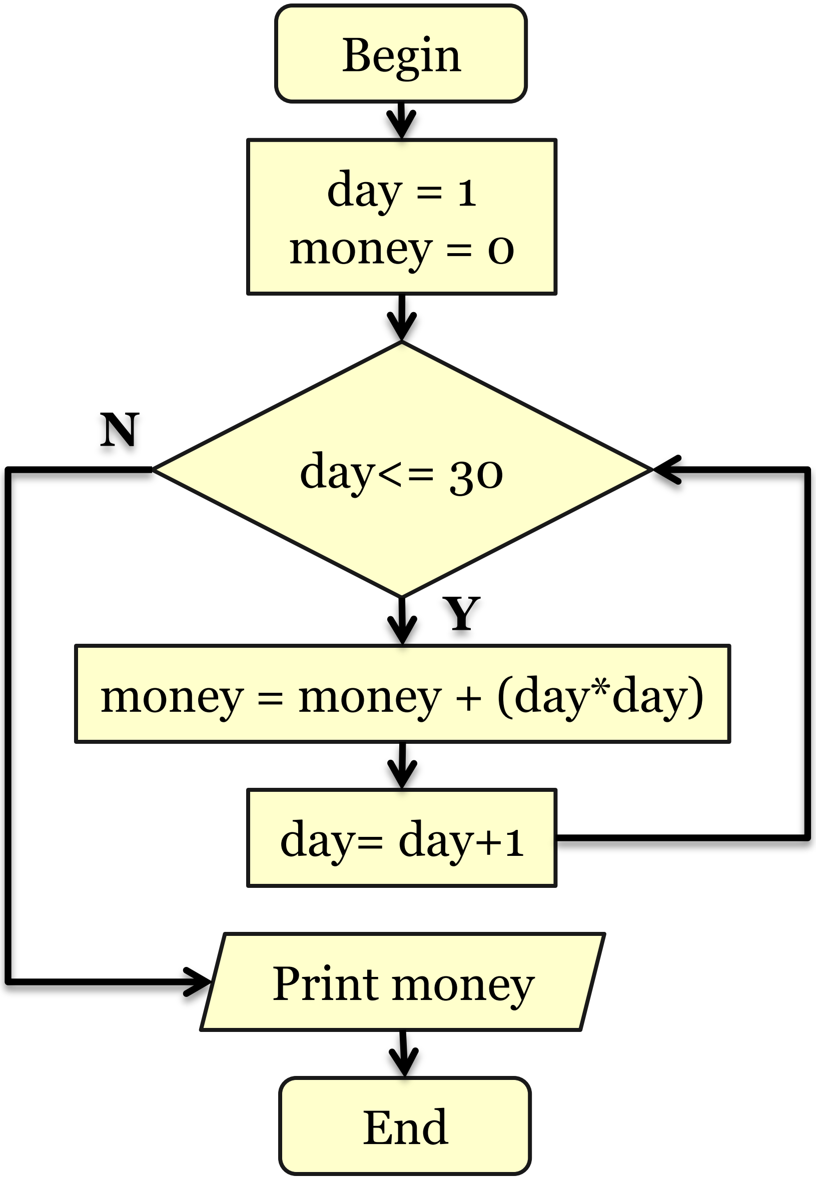
Solution: Plan A

```
day = 1;
money = 0;

while(day <= 30){
    money = money+200;
    day = day+1;
}
printf("%d\n", money);
```



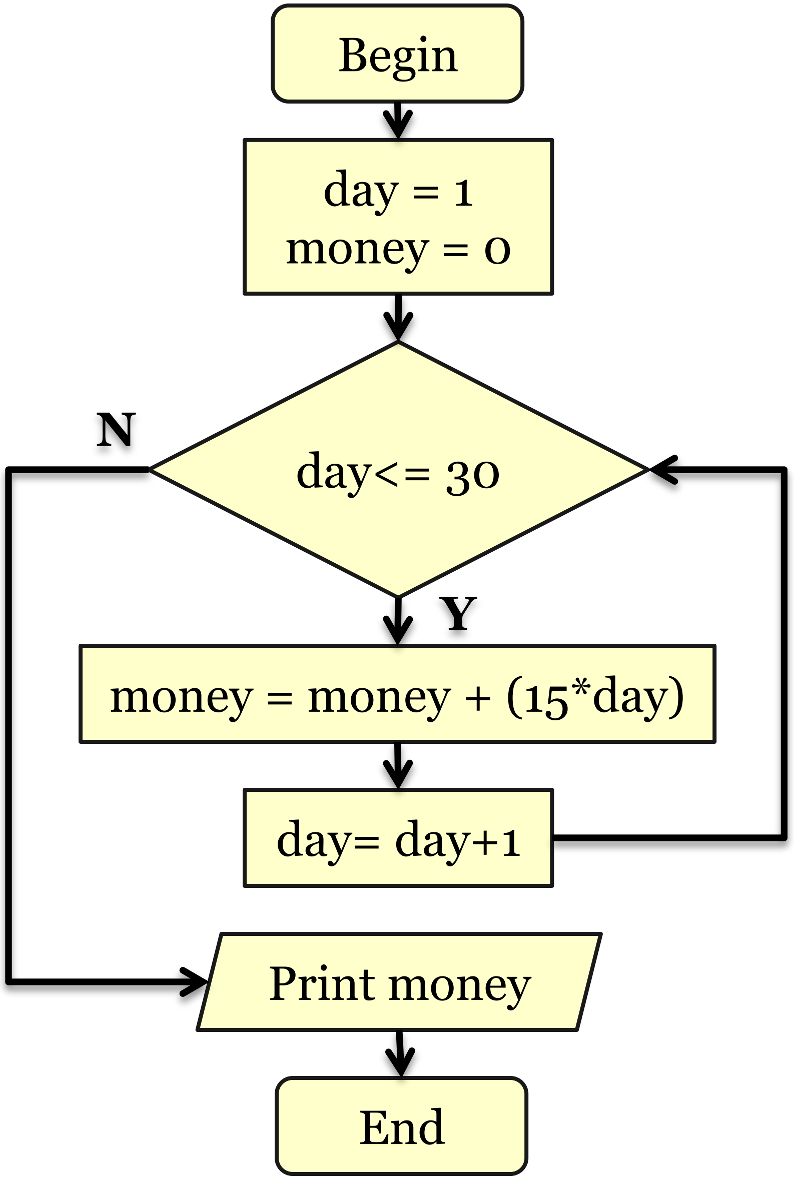
Solution: Plan B

```
day = 1;
money = 0;

while(day <= 30){
    money = money+(day*day);
    day = day+1;
}
printf("%d\n", money);
```



Solution: Plan C

```
day = 1;
money = 0;

while(day <= 30){
   money = money+(15*day);
   day = day+1;
}
printf("%d\n", money);
```



Problem 1: Answer

- Plan A: 6000 pesos
- Plan B: 9455 pesos
- Plan C: 6975 pesos

Example 2: A Bank Application

- Suppose your bank gives a **10% interest on your balance** every year, how much would your balance be after **20 years** assuming no other transactions? Also assume the bank adds the interest to your balance every year.

Example 2: Solution

```
float initial_deposit, balance, interest;

scanf("%f", &initial_deposit);
balance = initial_deposit;
for (year=1; year<=20; year++) {
    interest = 0.10 * balance;
    balance = balance + interest;
}
printf("After 20 years, the balance is P%f\n",
  balance);
```

year++
is essentially the same as
year = year+1

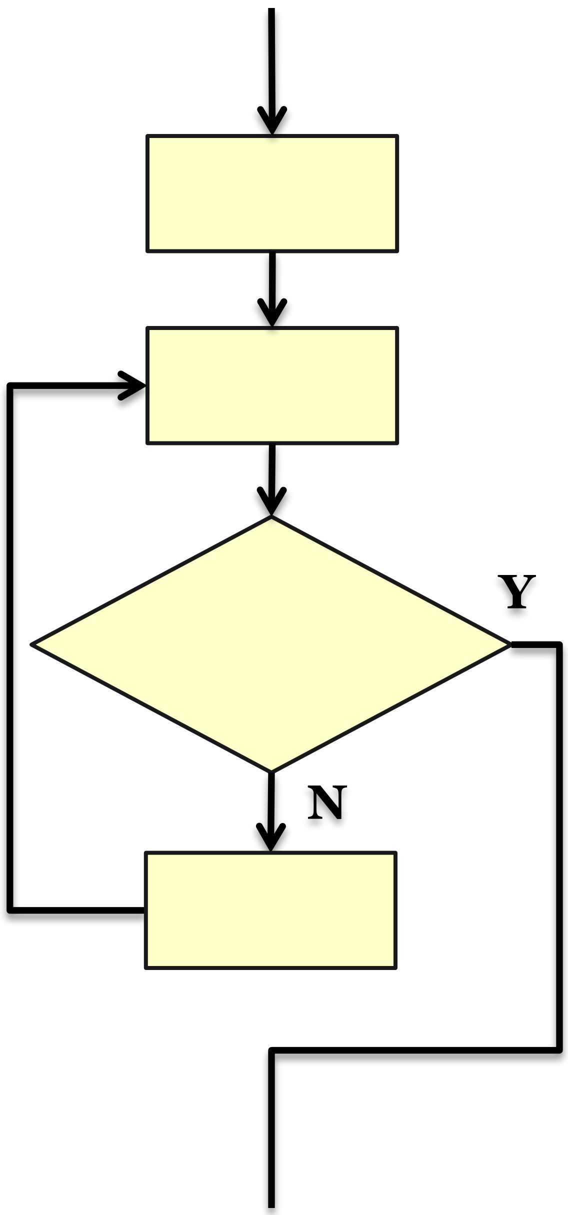
Loops can also be tested in the middle

- **An infinite for-loop**

```
for ( ; ; ) {
    // body of an infinite loop
}
```

- **For-loop with a test in the middle**

```
for ( ; ; ) {
    ....
    // test a condition and escape if true
    if ( condition ) break;
    ....
}
// go here when the condition is true
```



Two-person checkers game

```
//any integer except for 0 is considered true
while(1) {
    if (someone_has_won() || someone_wants_to_quit() == TRUE) {
        break;
   }

   take_turn(player1);

   if (someone_has_won() || someone_wants_to_quit() == TRUE) {
        break;
   }

   take_turn(player2);
}
```

A toy calculator

```
main(){
    int x, y, z;
    char op;
    printf("welcome to my toy integer calculator\n");
    for (;;) {
        printf("enter a simple integer expression: ");
        scanf("%d%c%d", &x, &op, &y);
        if ( op == '+' ){
            z = x + y;
        }
        else {
            printf("%c is an unknown operator\n", op);
            break;
        }
        printf("result is %d\n\n", z);
    }
        printf("thanks for using my calculator\n");
}
```

```
Welcome…

expression: 1+1
result is 2

Expression: 10-1
- is an unknown op

thanks…
```

Extending our toy calculator

- Extend our toy calculator to allow **multiplication** x*y, **integer division** x/y, **and the remainder operator** x%y (be sure to test for division-by-zero errors)
  - 9/2 evaluates to 4
  - 9%2 evaluates 1
  - 1/0 (error…. invalid operation)

- Extend our toy calculator to allow integer powers of x
  - 2^4 = 2*2*2*2 and evaluates to 16
  - 2^0 evaluates to 1
  - 2^-3 = 1/(2*2*2) = 0.125 (use **float to make sense)**

Computing integral powers, x^y

```
int x, y; // operands
char op; // operator
float result; // result of calculation
int j; // loop index variable
…
scanf("%d%c%d", &x, &op, &y);
…
  if (y >= 0){
        result = 1.0; // use a loop to compute 1*x*x*…*x
        for (j=1; j<=y; j++) {
              result = result * x;
        }
  }
  else // … what to do if the exponent y is negative?
…
```

Multiply by x, y times

More Exercises on Loops

- Input a positive integer n, and compute and print n factorial (*n*!) the product of all the integers from 1 to *n*.
- Input 2 positive integers A and B, and find the greatest common factor of A and B, i.e., the biggest integer that divides both A and B exactly.
- Input 2 positive integers A and B, and find the least common multiple of A and B, i.e., the smallest integer that is both a multiple of A and B.
- Input a positive integer *n*, and determine whether *n* is prime (no divisors except 1 and itself) or composite (has divisors other than 1 and itself).

Nested Loops

```
int i, j;

for(i=0; i<4; i++){
  for(j=0; j<5; j++){
      printf("*");
  }
  printf("\n");
}
```

What is the output?

Nested Loops

```
int i, j;

for(i=0; i<4; i++){
  for(j=0; j<5; j++){
     printf("*");
  }
  printf("\n");
}
```

What is the output?

i $\boxed{0}$

j $\boxed{?}$

OUTPUT:

Nested Loops

```
int i, j;

for(i=0; i<4; i++){
  for(j=0; j<5; j++){
     printf("*");
  }
  printf("\n");
}
```

What is the output?

i	0
j	?

OUTPUT:

Nested Loops

```
int i, j;

for(i=0; i<4; i++){
  for(j=0; j<5; j++){
     printf("*");
  }
  printf("\n");
}
```

What is the output?

i $\boxed{0}$

j $\boxed{0}$

OUTPUT:

Nested Loops

```
int i, j;

for(i=0; i<4; i++){
  for(j=0; j<5; j++){
     printf("*");
  }
  printf("\n");
}
```

What is the output?

i 0

j 0

OUTPUT:

Nested Loops

```
int i, j;

for(i=0; i<4; i++){
  for(j=0; j<5; j++){
     printf("*");
  }
  printf("\n");
}
```

What is the output?

i	0
j	0

OUTPUT:

```
*
```

Nested Loops

```
int i, j;

for(i=0; i<4; i++){
  for(j=0; j<5; j++){
     printf("*");
  }
  printf("\n");
}
```

What is the output?

i	0
j	1

OUTPUT:

```
*
```

Nested Loops

```
int i, j;

for(i=0; i<4; i++){
  for(j=0; j<5; j++){
     printf("*");
  }
  printf("\n");
}
```

What is the output?

i 0

j 1

OUTPUT:
*

Nested Loops

```
int i, j;

for(i=0; i<4; i++){
  for(j=0; j<5; j++){
     printf("*");
  }
  printf("\n");
}
```

What is the output?

i 0

j 1

OUTPUT:

```
**
```

Nested Loops

```
int i, j;

for(i=0; i<4; i++){
  for(j=0; j<5; j++){
     printf("*");
  }
  printf("\n");
}
```

What is the output?

i	0
j	2

OUTPUT:

```
**
```

Nested Loops

```
int i, j;

for(i=0; i<4; i++){
  for(j=0; j<5; j++){
      printf("*");
  }
  printf("\n");
}
```

What is the output?

i | 0

j | 2

OUTPUT:

```
**
```

Nested Loops

```
int i, j;

for(i=0; i<4; i++){
  for(j=0; j<5; j++){
    printf("*");
  }
  printf("\n");
}
```

What is the output?

i	0
j	2

OUTPUT:

```
***
```

Nested Loops

```
int i, j;

for(i=0; i<4; i++){
  for(j=0; j<5; j++){
     printf("*");
  }
  printf("\n");
}
```

What is the output?

i	0
j	3

OUTPUT:

```
***
```

Nested Loops

```
int i, j;

for(i=0; i<4; i++){
  for(j=0; j<5; j++){
    printf("*");
  }
  printf("\n");
}
```

What is the output?

i 0

j 3

OUTPUT:
```
***
```

Nested Loops

```
int i, j;

for(i=0; i<4; i++){
  for(j=0; j<5; j++){
     printf("*");
  }
  printf("\n");
}
```

What is the output?

i 0

j 3

OUTPUT:

```
****
```

Nested Loops

```
int i, j;

for(i=0; i<4; i++){
  for(j=0; j<5; j++){
     printf("*");
  }
  printf("\n");
}
```

What is the output?

i 0

j 4

OUTPUT:

```
****
```

Nested Loops

```
int i, j;

for(i=0; i<4; i++){
  for(j=0; j<5; j++){
    printf("*");
  }
  printf("\n");
}
```

What is the output?

i	0
j	4

OUTPUT:

```
****
```

Nested Loops

```
int i, j;

for(i=0; i<4; i++){
  for(j=0; j<5; j++){
     printf("*");
  }
  printf("\n");
}
```

What is the output?

i 0

j 4

OUTPUT:

```
*****
```

Nested Loops

```
int i, j;

for(i=0; i<4; i++){
  for(j=0; j<5; j++){
     printf("*");
  }
  printf("\n");
}
```

What is the output?

i: 0

j: 5

OUTPUT:

```
*****
```

Nested Loops

```
int i, j;

for(i=0; i<4; i++){
  for(j=0; j<5; j++){
     printf(“*”);
  }
  printf(“\n”);
}
```

What is the output?

i 0

j 5

OUTPUT:

```
*****
```

Nested Loops

```
int i, j;

for(i=0; i<4; i++){
  for(j=0; j<5; j++){
      printf("*");
  }
  printf("\n");
}
```

What is the output?

i 0

j 5

OUTPUT:

```
*****
```

Nested Loops

```
int i, j;

for(i=0; i<4; i++){
  for(j=0; j<5; j++){
      printf("*");
  }
  printf("\n");
}
```

What is the output?

i 1

j 5

OUTPUT:
```
*****
```

Nested Loops

```
int i, j;

for(i=0; i<4; i++){
  for(j=0; j<5; j++){
     printf("*");
  }
  printf("\n");
}
```

What is the output?

i $\boxed{1}$

j $\boxed{5}$

OUTPUT:

```
*****
```

Nested Loops

```
int i, j;

for(i=0; i<4; i++){
  for(j=0; j<5; j++){
    printf("*");
  }
  printf("\n");
}
```

What is the output?

```
i  1

j  5
```

OUTPUT:

```
*****
*****
```

Nested Loops

```
int i, j;

for(i=0; i<4; i++){
  for(j=0; j<5; j++){
     printf("*");
  }
  printf("\n");
}
```

What is the output?

i 2

j 5

OUTPUT:

```
*****
*****
```

Nested Loops

```
int i, j;

for(i=0; i<4; i++){
  for(j=0; j<5; j++){
     printf("*");
  }
  printf("\n");
}
```

What is the output?

i 2

j 5

OUTPUT:

```
*****
*****
```

Nested Loops

```
int i, j;

for(i=0; i<4; i++){
  for(j=0; j<5; j++){
    printf(“*”);
  }
  printf(“\n”);
}
```

What is the output?

i 2

j 5

OUTPUT:

```
*****
*****
*****
```

Nested Loops

```
int i, j;

for(i=0; i<4; i++){
  for(j=0; j<5; j++){
     printf("*");
  }
  printf("\n");
}
```

What is the output?

i 3

j 5

OUTPUT:

```
*****
*****
*****
```

Nested Loops

```
int i, j;

for(i=0; i<4; i++){
  for(j=0; j<5; j++){
     printf("*");
  }
  printf("\n");
}
```

What is the output?

i 3

j 5

OUTPUT:

```
*****
*****
*****
```

Nested Loops

```
int i, j;

for(i=0; i<4; i++){
  for(j=0; j<5; j++){
    printf("*");
  }
  printf("\n");
}
```

What is the output?

i 3

j 5

OUTPUT:

```
*****
*****
*****
*****
```

Nested Loops

```
int i, j;

for(i=0; i<4; i++){
  for(j=0; j<5; j++){
     printf("*");
  }
  printf("\n");
}
```

What is the output?

i 4

j 5

OUTPUT:

```
*****
*****
*****
*****
```

Nested Loops

```
int i, j;

for(i=0; i<4; i++){
  for(j=0; j<5; j++){
     printf("*");
  }
  printf("\n");
}
```

What is the output?

i 4

j 5

OUTPUT:

```
*****
*****
*****
*****
```

Nested Loops

```
int i, j;

for(i=0; i<4; i++){
  for(j=0; j<5; j++){
      printf("*");
  }
  printf("\n");
}
```

What is the output?

i 4

j 5

OUTPUT:

```
*****
*****
*****
*****
```

Example 1

- Print a *10* x *10* multiplication table.

How will you print the first row?

```
int i;

for(i=1; i<=10; i++){
 printf("%d ", i);
}
```

How will you print the second row?

```
int i;

for(i=1; i<=10; i++){
 printf("%d ", i*2);
}
```

How will you print the third row?

```
int i;

for(i=1; i<=10; i++){
 printf("%d ", i*3);
}
```

Solution:

```
int i, j;

for(j=1; j<=10; j++){
  for(i=1; i<=10; i++){
      printf(“%d ”, i*j);
  }
  printf(“\n”);
}
```

Example 2

- Print all prime numbers from 1 to 100.

Review:

How do you determine whether a number is prime?

Review: Prime or Composite?

```
int n, i;

scanf("%d", &n);

for(i=2; i<=n/2; i++){
   if(n%i == 0){
        break;
   }
}
if(i<=n/2){
   printf("Composite\n");
}
else{
   printf("Prime\n");
}
```

Solution

```
int n, i;

for(n=1; n<=100; n++){
    for(i=2; i<=n/2; i++){
            if(n%i == 0){
                    break;
            }
    }
    if(i<=n/2){
            printf("Composite\n");
    }
    else{
            printf("Prime\n");
    }
}
```

Solution

```
int n, i;

for(n=1; n<=100; n++){
    for(i=2; i<=n/2; i++){
        if(n%i == 0){
            break;
        }
    }
    if(!(i<=n/2)){
        printf("%d\n", n);
    }
}
```

Example 3: Patterns

```
*       *
  *   *
    *
  *   *
*       *
```

```
* * * * *
*       *
* * * * *
```

```
                #
              # # #
            # # # # #
              # # #
            # # # # #
          # # # # # # #
        # # # # # # # # #
            # # # # #
          # # # # # # #
        # # # # # # # # #
      # # # # # # # # # # #
    # # # # # # # # # # # # #
          # # # # # # #
        # # # # # # # # #
      # # # # # # # # # # #
    # # # # # # # # # # # # #
  # # # # # # # # # # # # # # #
# # # # # # # # # # # # # # # # #
```

Merry Christmas!

```
*
* *
* * *
* * * *
* * * * *
```

```
* * *
*   *
*   *
*   *
* * *
```

Example 3: Patterns

	0	1	2	3	4
0	*				*
1		*		*	
2			*		
3		*		*	
4	*				*

i = 0, j = 0	i = 0, j = 4
i = 1, j = 1	i = 1, j = 3
i = 2, j = 2	i = 2, j = 2
i = 3, j = 3	i = 3, j = 1
i = 4, j = 4	i = 4, j = 0

i == j OR i+j == size-1