CS1010E Lecture 2

Control Structures: Selection

Henry Chia (hchia@comp.nus.edu.sg)

Semester 1 2016 / 2017

Lecture Outline

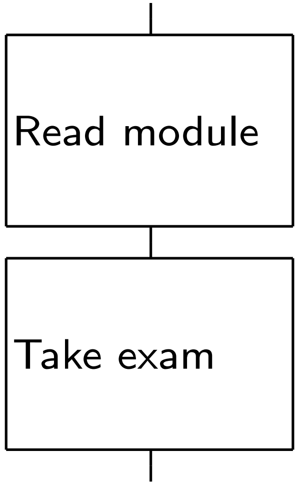
- Algorithmic problem solving
- Control structures
- Boolean values, variables and expressions
- Relational and Logical Operators
- Selection statements
- Nested selection statements

Algorithmic Problem Solving

- □ Algorithm – set of instructions that manipulates data to solve an algorithmic problem.
- □ Control structures (sequence, selection, and repetition) provide the flow of control in an algorithm
- □ Characteristics of an algorithm:
  - – Each step of an algorithm is **exact**
  - – An algorithm must **terminate**
  - – An algorithm must be **effective**
  - – An algorithm must be **general**

Control Structures – Sequence

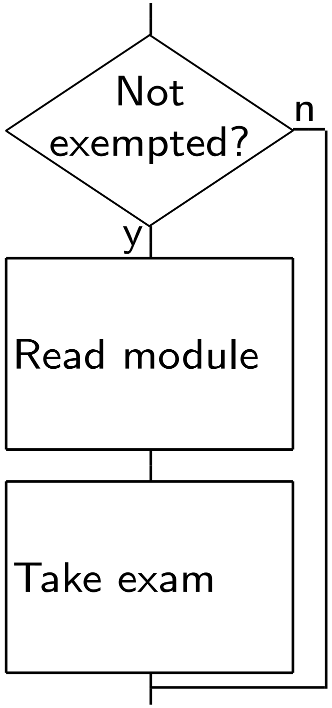
- □ A **sequence** structure contains steps that are performed one after another
- □ Example: To pass this course, you have to read the module, and then take the exam



Control Structures – Selection

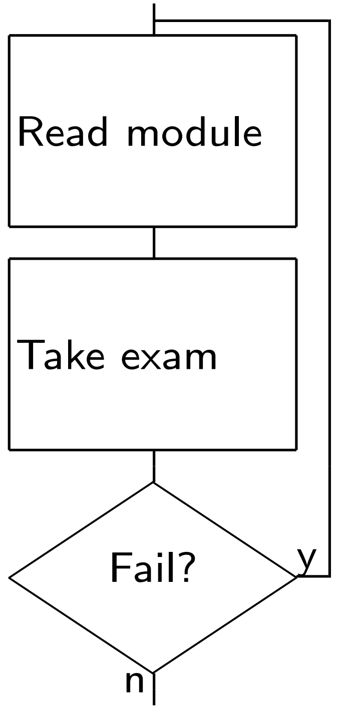
□ A **selection** structure contains one set of steps that is performed if a *condition* is true, and possibly another set of steps that is performed if a *condition* is false

□ Example: If you are not exempted from the module, then you need to read the module and take the exam



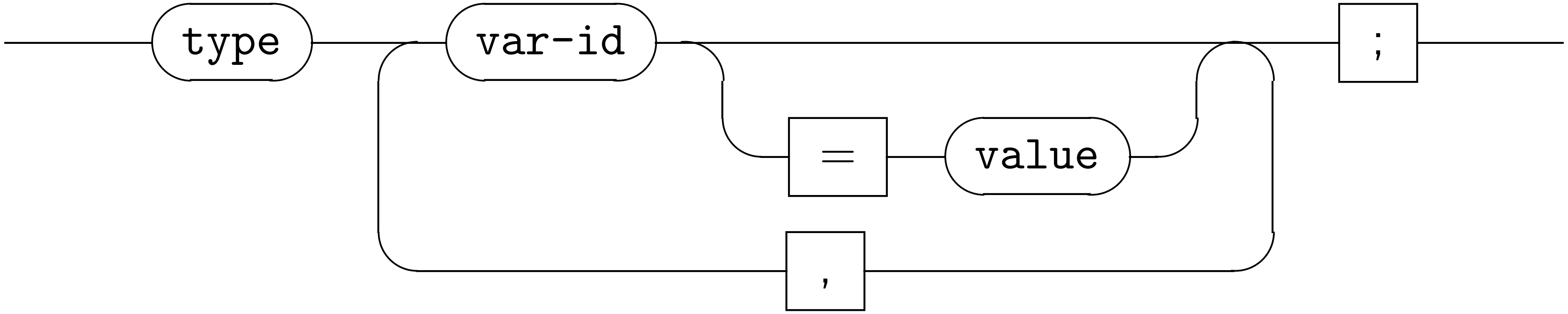
Control Structures – Repetition

- A **repetition** structure contains a set of steps that is repeated as long as a *condition* is true
- Example: One who reads the module, takes the exam but fail will need to repeat again



Declaring Boolean Variables

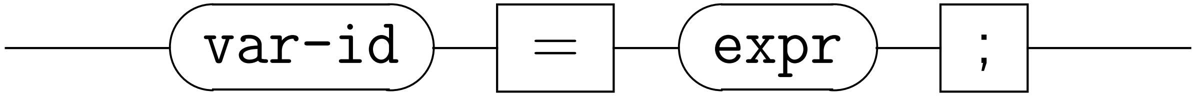
declaration



- □ **type**: `bool`
- □ **value**: `true`, `false`
  - – To specify `true` and `false` within the program, use `#include <stdbool.h>`
- □ Boolean variable identifiers should suggest a true/false outcome, e.g. `overWeight`, `underWeight`, but not `weight`
- □ Example: `bool overWeight=true;`

Boolean Assignment, Input and Output

assignmentStatement



- □ Example:

  ```
  overWeight = false;
  ```

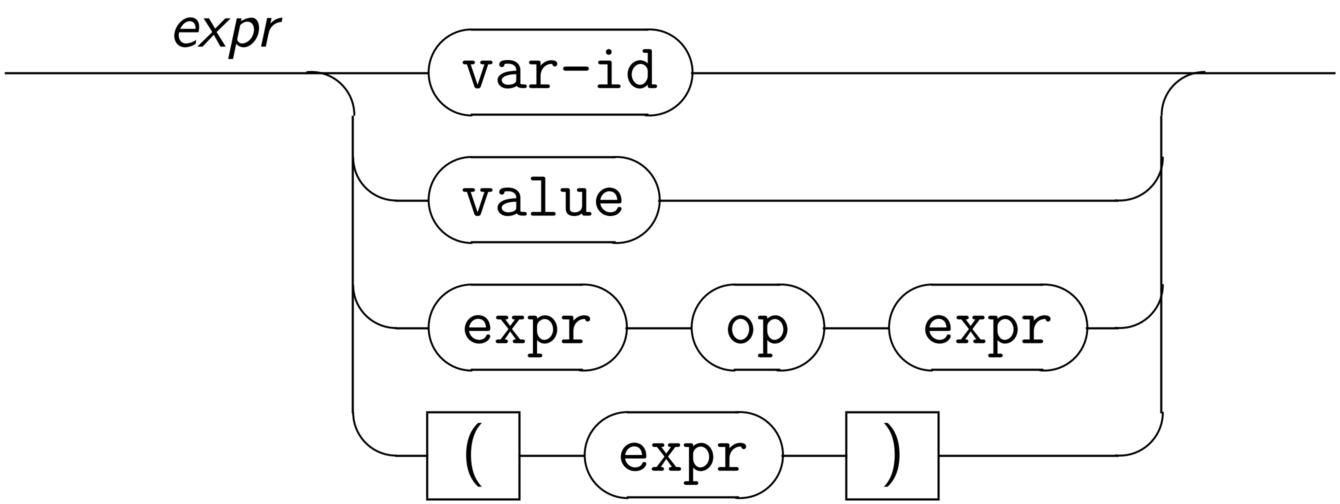
- □ C associates `true` with integer 1, and `false` with integer 0
  - – To output a boolean expression, use the %d for the placeholder; 0 or 1 is displayed

    ```
    printf("Is overweight? %d\n", overWeight);
    ```

  - – Likewise, input 0 or 1 when reading a boolean value

    ```
    scanf("%d", &overWeight);
    ```

Boolean Expression – Condition



- A condition is an expression that evaluates to `true`/`false`
- Two types of operations that evaluates to `true`/`false`
  - Relational operations that operates on two arithmetic expressions
  - Logical (Boolean) operations that operates on two conditions

Relational Operators

□ Relational operators compare two arithmetic expressions:

Relational Op	Interpretation
<	less than
<=	less than or equal to
>	greater than
>=	greater than or equal to
==	equal to
!=	not equal to

□ E.g. conditions using relational operators:

- `x == y`
- `(mass/(ht*ht)) > 24.9`

Logical Operators

- □ Logical (Boolean) operators compare conditions
- □ Three logical operators: **and** (`&&`), **or** (`||`), **not** (`!`)

A	B	A && B	A || B	!A	!B
false	false	false	false	true	true
false	true	false	true	true	false
true	false	false	true	false	true
true	true	true	true	false	false

  - `A && B` is true only if both `A` and `B` are true
  - `A || B` is false only if both `A` and `B` are false
  - `!A` is true only if `A` is false

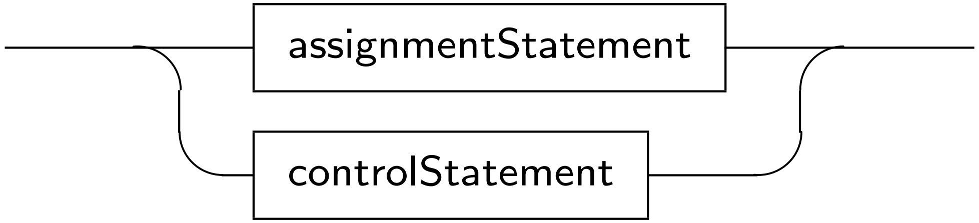
- □ Example: `(bmi >= 18.5) && (bmi <= 24.9)`
- □ How about this? `(18.5 <= bmi <= 24.9)`

Logical Operators: Short-Circuit

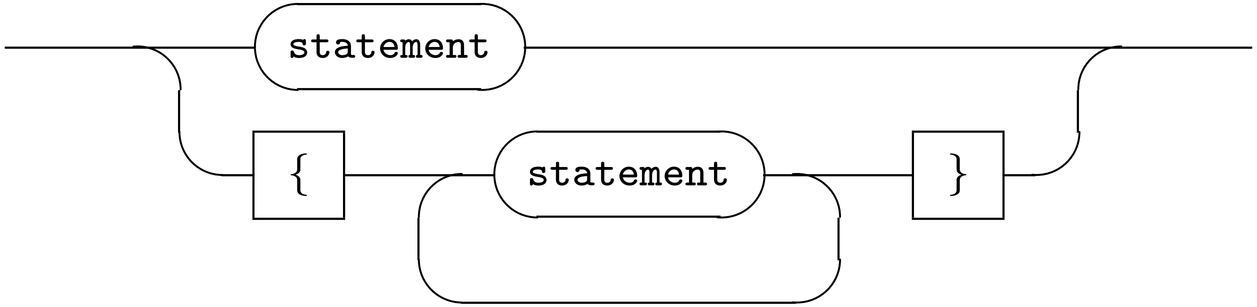
- □ When expressions with logical operators are executed, C will only evaluate as much of the expression as necessary to evaluate it
  - If `A` is false, then the expression `A && B` is also false, and there is no need to evaluate `B`
  - If `A` is true, then the expression `A || B` is also true, and there is no need to evaluate A

Statement and Statement Block

statement



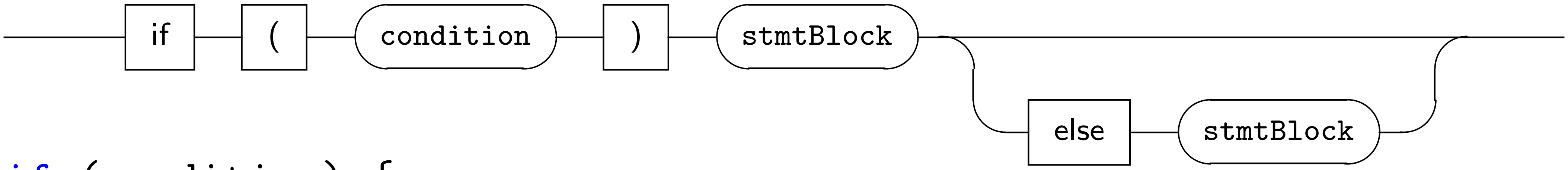
stmtBlock



- □ Control Statements
  - – Selection
    - ▷ `if..else`
  - – Repetition
    - ▷ `do..while`
    - ▷ `while`
    - ▷ `for`
- □ Statement Block – one statement or group of statements

Selection: `if..else` Statement

ifElseStatement



```
if (condition) {
   statement;
   ...
} else {
   statement;
   ...
}
```

□ Executes the *if* statement block when the *condition* is true; otherwise the *else* statement block is executed

□ *Else* statement block is optional

□ Curly braces may be omitted (but encouraged) for statement block consisting of one statement

Example: Maximum of two numbers

- □ Using one if..else construct
- □ Easier to understand

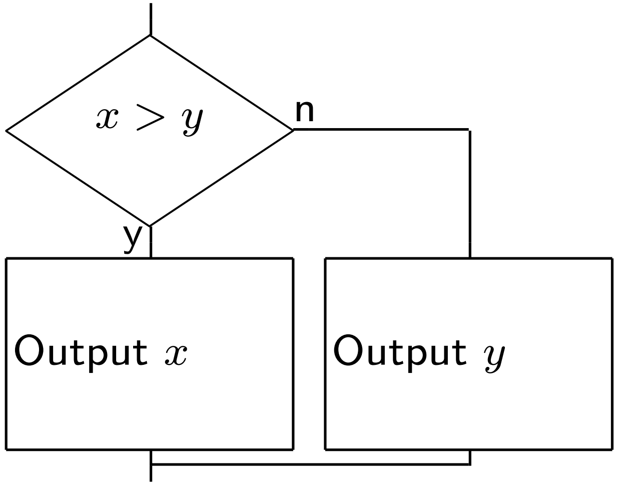
```
/*
   This program determines the maximum of two input numbers.
*/
#include <stdio.h>

int main(void) {
   int x=0, y=0;

   printf("Enter two numbers: ");
   scanf("%d%d", &x, &y);

   if (x > y) {
      printf("Maximum is %d\n", x);
   } else {
      printf("Maximum is %d\n", y);
   }

   return 0;
}
```



Example: Maximum of two numbers

- □ Using two if constructs
- □ Requires two conditions

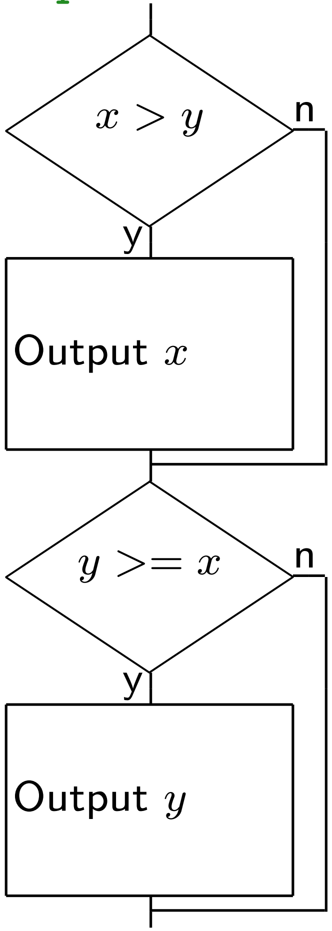
```
/*
   This program determines the maximum of two input numbers.
*/
#include <stdio.h>

int main(void) {
   int x=0, y=0;

   printf("Enter two numbers: ");
   scanf("%d%d", &x, &y);

   if (x > y) {
      printf("Maximum is %d\n", x);
   }
   if (y >= x) { /* if (y > x) ??? */
      printf("Maximum is %d\n", y);
   }

   return 0;
}
```



Exercise: Maximum of two numbers

□ Using only one if construct

```
/*
   This program determines the maximum of two input numbers.
*/
#include <stdio.h>

int main(void) {
   int x=0, y=0;

   printf("Enter two numbers: ");
   scanf("%d%d", &x, &y);
```

Example: BMI

□ Using the logical && operator in the condition

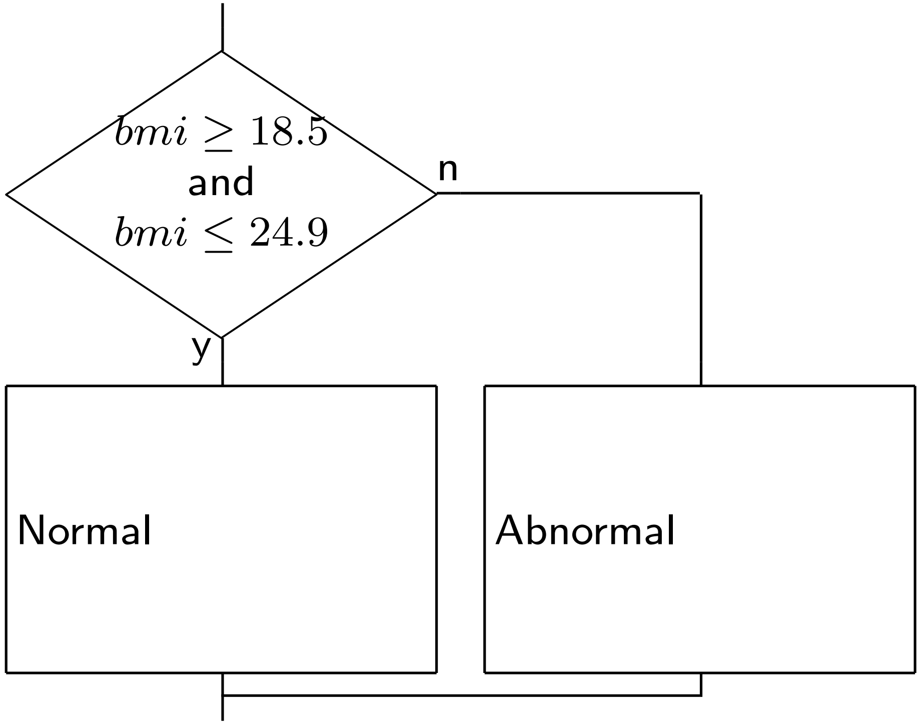
```
/*
   This program classifies a BMI input as Normal/Abnormal.
*/
#include <stdio.h>

int main(void) {
   double bmi=0.0;

   printf("Enter bmi: ");
   scanf("%lf", &bmi);

   if (bmi >= 18.5 && bmi <= 24.9) {
      printf("Normal\n");
   } else {
      printf("Abnormal\n");
   }

   return 0;
}
```



Example: BMI

□ Using the logical || operator in the condition

□ Condition is expressed in the opposite sense

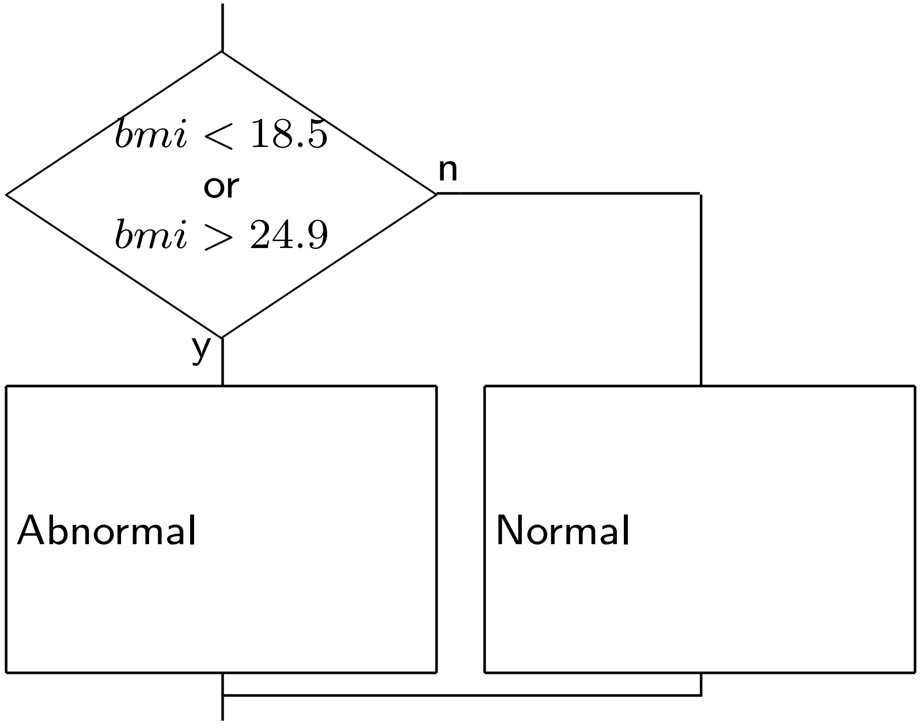
```
/*
   This program classifies a BMI input as Normal/Abnormal.
*/
#include <stdio.h>

int main(void) {
   double bmi=0.0;

   printf("Enter bmi: ");
   scanf("%lf", &bmi);

   if (bmi < 18.5 || bmi > 24.9) {
      printf("Abnormal\n");
   } else {
      printf("Normal\n");
   }

   return 0;
}
```



Nesting `if..else` Statements

- □ Nested `if`s represent &&

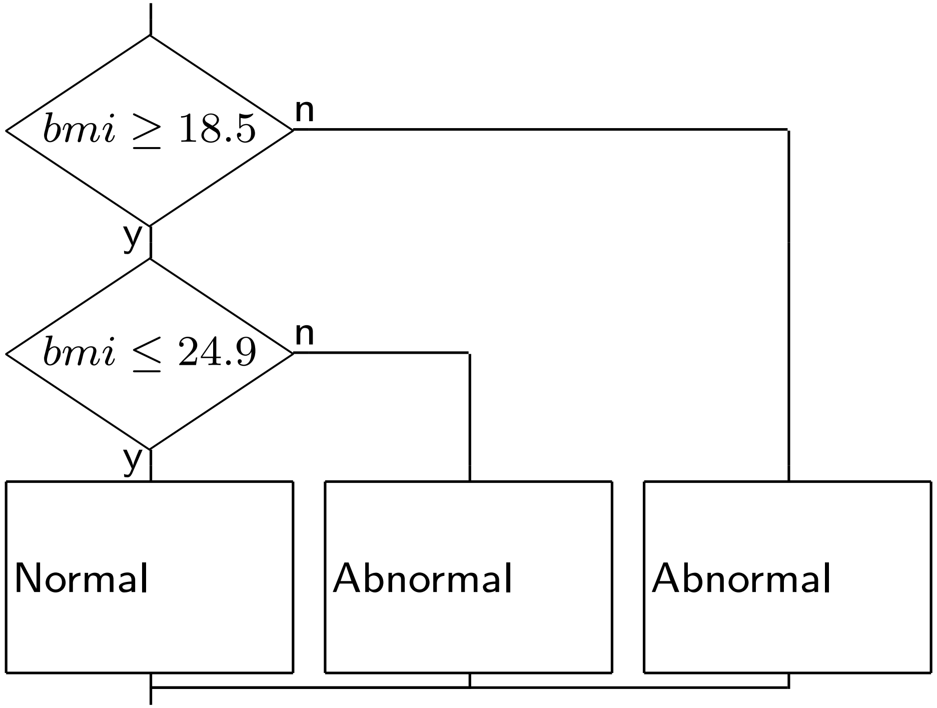
```c
/*
   This program classifies a BMI input as Normal/Abnormal.
*/
#include <stdio.h>

int main(void) {
   double bmi=0.0;

   printf("Enter bmi: ");
   scanf("%lf", &bmi);

   if (bmi >= 18.5) {
      if (bmi <= 24.9) {
         printf("Normal\n");
      } else {
         printf("Abnormal\n");
      }
   } else {
      printf("Abnormal\n");
   }

   return 0;
}
```



Nesting `if..else` Statements

□ Resolving case-by-case starting with the lowest BMI values

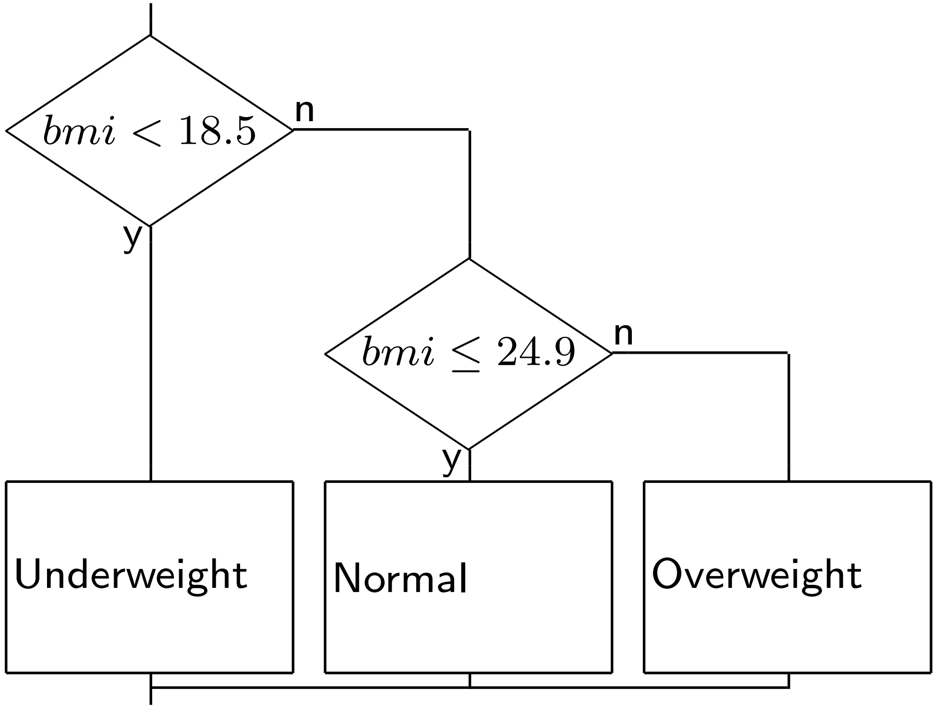
```
/*
   This program classifies a BMI input as Normal/Underweight/Overweight.
*/
#include <stdio.h>

int main(void) {
   double bmi=0.0;

   printf("Enter bmi: ");
   scanf("%lf", &bmi);

   if (bmi < 18.5) {
      printf("Underweight\n");
   } else {
      if (bmi <= 24.9) {
         printf("Normal\n");
      } else {
         printf("Overweight\n");
      }
   }

   return 0;
}
```



Nesting `if..else` Statements

□ Resolving case-by-case starting with the highest BMI values

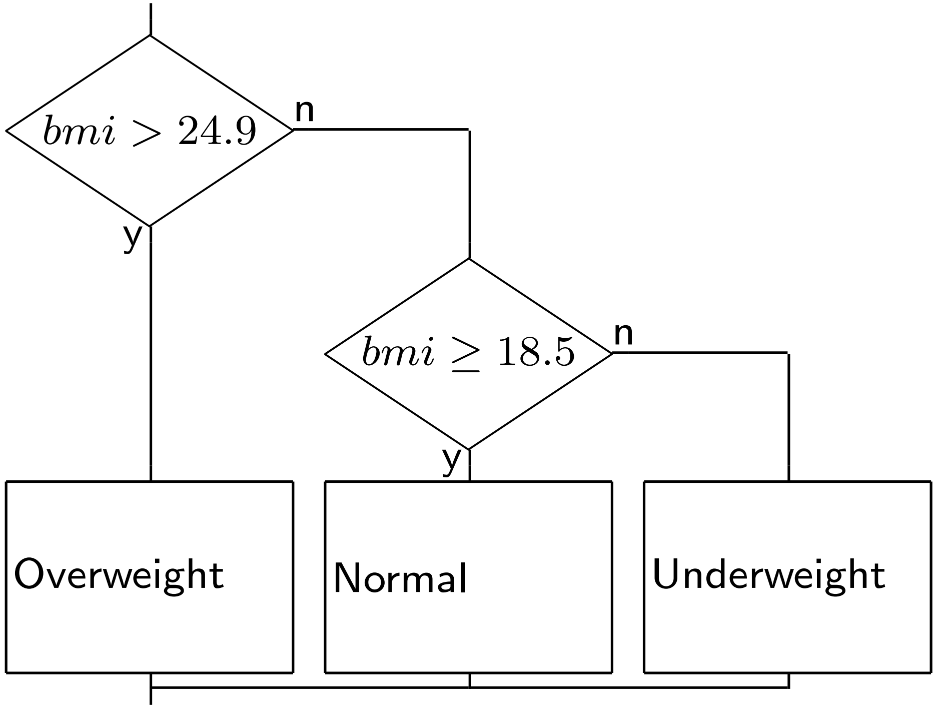
```
/*
   This program classifies a BMI input as Normal/Underweight/Overweight.
*/
#include <stdio.h>

int main(void) {
   double bmi=0.0;

   printf("Enter bmi: ");
   scanf("%lf", &bmi);

   if (bmi > 24.9) {
      printf("Overweight\n");
   } else {
      if (bmi >= 18.5) {
         printf("Normal\n");
      } else {
         printf("Underweight\n");
      }
   }

   return 0;
}
```



Nesting `if..else` Statements

- □ Still correct, but difficult to understand

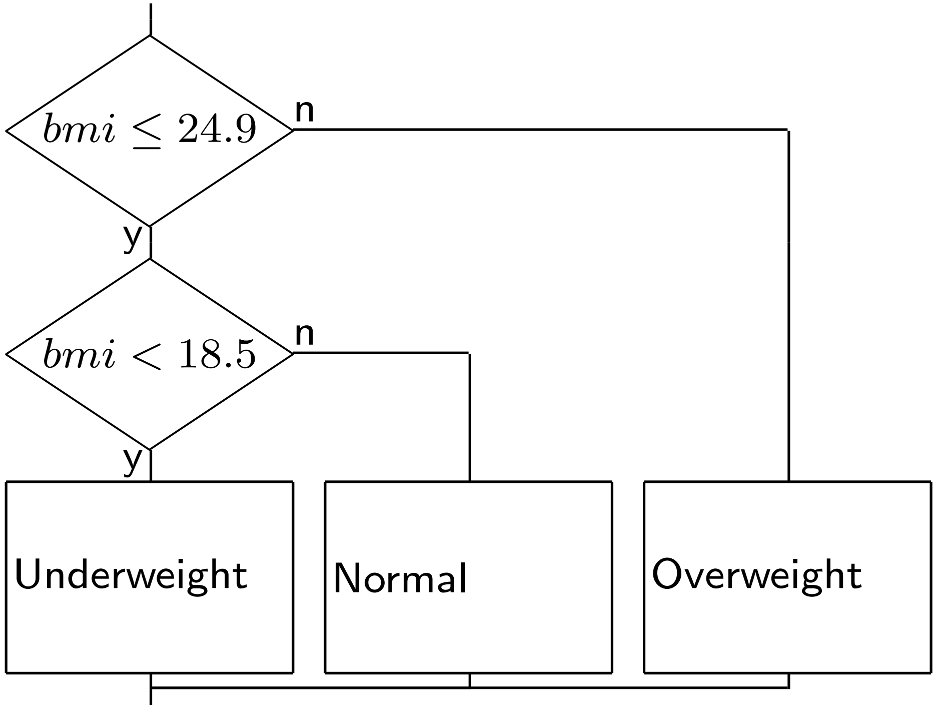
```
/*
   This program classifies a BMI input as Normal/Underweight/Overweight.
*/
#include <stdio.h>

int main(void) {
   double bmi=0.0;

   printf("Enter bmi: ");
   scanf("%lf", &bmi);

   if (bmi <= 24.9) {
      if (bmi < 18.5) {
         printf("Underweight\n");
      } else {
         printf("Normal\n");
      }
   } else {
      printf("Overweight\n");
   }

   return 0;
}
```



Lecture Summary

- □ Characteristics of an algorithm
- □ Control structures: sequence, selection and repetition
- □ Conditions involving relational and logical operators
- □ Selection
  - – `if..else` statement
  - – Nested `if..else` statements
- □ Lay out nested `if..else` constructs in an easy to understand fashion, typically resolving case-by-case starting from one end of the range of possible values, and working towards the other end