

**Programming Fundamentals for Engineers
0702113**

3. Selection Statements

MUNTASER ABULAFI
DR. LABIB ARAFEH
YACOUB SABATIN

1

Introduction

- Selection Statements: Sometimes in programming, we need to be able to choose between **two** or **more** alternatives.
- Selection statements include *if* and **switch** statements.
- Logical expressions need to be discussed as it is used in statements such as *if*
- **Logical Expressions:** A logical expression such as $a < b$ appears in statements such as '*if*' that:
 - Tests value of expression, and
 - Chooses a course of action based on whether the value is "**true**" **1** or "**false**" **0**;

2

- **Operators** used to build Logical Expressions:
 - Relational Operators (<, >, <=, >=);
 - Equality Operators (==, !=);
 - Logical Operators (!, &&, ||).
- **Relational Operators:**
 - <, >, <= and >=
 - Produce **0** or **1** when used in expressions.
 - Are Binary **infix** operators (Operators between operands).
 - **Lower precedence** than Arithmetic Operators.
 - **Left Associative (evaluated LtR)**.
- **Example:**
 - $20 < 30$ → produces **1**
 - $30 < 20$ → produces **0**
 - $a + b > c - 2$ → means $(a+b) > (c-2)$

3

C_Prog. Spring 2012

- $a < b < c$ does not test if **b** lies between **a** and **c**!
Instead, the value of $a < b$ (which can be **0** or **1**) is compared against **c**.
- **Equality Operators:**
 - ==, !=
 - Produce either **0** “False” or **1** “True”;
 - Are **binary Infix** operators;
 - **Lower precedence** than Relational Operators;
 - **Left Associative**.
- **Example:**
 - $a == b < c$ → equivalent to $a == (b < c)$
 - $a < b == b < c$ → equivalent to $(a < b) == (b < c)$
 - **TRUE: IF $a < b$ and $b < c$ are both True or False**
(Both relational operators are equal in priority)
 - $a == b < c$ → equivalent to $a == (b < c)$

4

C_Prog. Spring 2012

Hint: Equality Operators have lower precedence than Relational Operators;

• **Notes:**

– Compiler deals with 0's & 1's as logical operations' results

– '**bool**' keyword: Used to declare Boolean variables.

• **Logical Operators : ! (NOT), && (AND), || (OR)**

– Used to **combine** simple logical expressions into more complex ones;

– **! (Negation): Toggles** operand's value: **0** to **1**, or **1** to **0**

- Is **unary** prefix, **right associative (RtL)**;

5

C_Prog. Spring 2012

– **&& (AND), || (OR): Binary** infix, **left associative**;

- **&&**: Produces **1** if **both** its operands are **Non-Zero**, otherwise, produces **Zero**

- **||**: Produces **1** if **either** operands are **Non-Zero**

– Both **||** and **&&** have **left-to-right** order of evaluation:

- Left operand is evaluated **first**, and Right operand is evaluated ***IF Necessary!***

(3==4) && exp'n Yields (0) without evaluating exp'n.

(3==3) || exp'n Yields (1) without evaluating exp'n!

Examples:

age > 12 && age < 20

gpa > 3.5 || Twajihi_Score > 850

gpa > 3.5 || Twajihi_Score > 850 && age > 17

6

C_Prog. Spring 2012

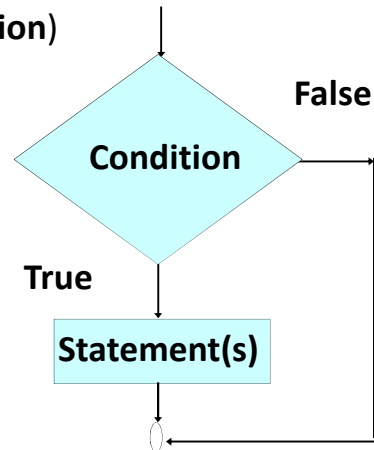
Making Decisions: *The if Statement*

- *if* statements allows branching (decision making) depending upon the value or state of variables.
- Allows statements to be **executed** or **skipped**, depending upon decisions.
- ***if first form*** (**One-Way Selection**)

–Syntax:

```
if (expression / condition)
    statement;
```

```
if (expression / condition)
{
    statement1;
    statement2;
    statementn;
}
```

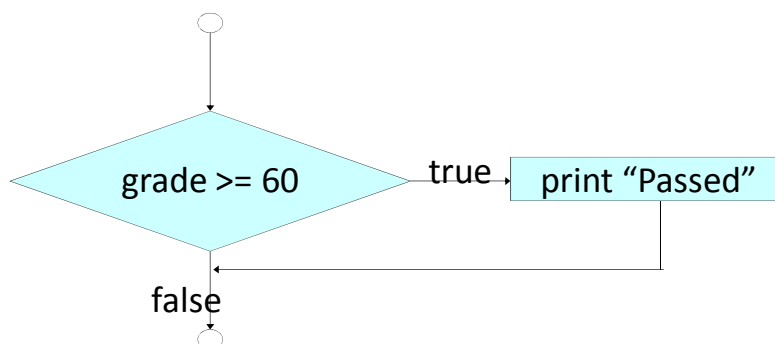


7

C_Prog. Spring 2012

Pseudocode:

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



Statement in C:

```
if ( grade >= 60 )
    printf( "Passed\n" );
```

8

C_Prog. Spring 2012

- **Examples:**

```

if (hours > 40)
    printf("Overtime!\n");
if (students < 65)
    ++student_count;
if (temperature < 0)
    printf("Frozen\n");
if (Cost > 30000)
{
    Tax = (Cost - 30000) * 0.1;
    printf("For a car costing %8.2f, a
    tax of %8.2f is due.", Cost, Tax);
}
printf("Thanks for helping the economy");

```

9

C_Prog. Spring 2012

- **if Second form (Two-Way Selection):**

Syntax:

```

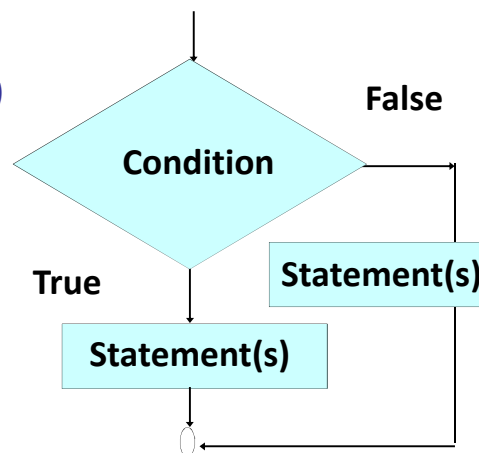
if (expression / condition)
    statement;
else
    statement;

```

```

if (expression / condition)
{ statement1;
  statement2;
  statementn;}
else
{ statement1;
  statement2;
  statementn;}

```



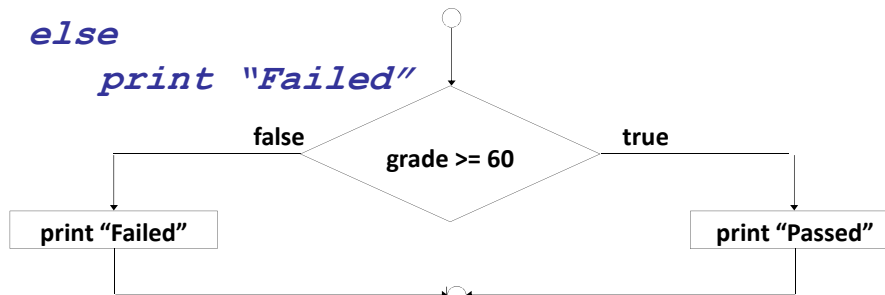
10

C_Prog. Spring 2012

- Psuedocode:

*If grade is greater than or equal to 60
print "Passed"*

*else
print "Failed"*



```

if ( grade >= 60 )
    printf( "Passed\n" );
else
    printf( "Failed\n" );
  
```

11

C_Prog. Spring 2012

Examples:

```

if (result >= 45)
    printf("Pass\n");
else
    printf("Fail\n");
//Tests whether a student has passed (a pass mark of 45)
if (hours <= Reg_Hours)
    Gross_Pay = Wage * Hours;
else
{
    Overtime_Hours = Hours - Reg_Hours;
    Overtime_Pay = Wage * Overtime_Hours *
        Overtime_Rate;
    Gorss_Pay=Wage*Reg_Hours+Overtime_Pay;
}
  
```

12

C_Prog. Spring 2012

- **Compound statements** has the form of:

- **{statements}**

→ Putting **braces** around group of statements will force compiler to treat it as single statement.

```
if (result >= 45)
{
    printf("Passed\n");
    printf("Congratulations\n");
}
else
{
    printf("Failed\n");
    printf("Good luck in the rests\n");
}
```

13

C_Prog. Spring 2012

Cascaded if example (ladder)

```
if (n<0)
    printf("n is less than 0\n");
else
    if (n==0)
        printf("n is equal to 0\n");
    else
        printf("n is greater than 0\n");
```

- The second **if** is nested inside the first.
- This leads to *ladder* shape
- **if third form** (*Multi-Way Selection / Ladder Structure*):

14

C_Prog. Spring 2012

- **(Multi-Way Selection / Ladder Structure) Syntax:**

```

if (expression1 / condition-1)
    {statement-1;}
else if (expression2 / condition-2)
    {statement-2;}
else if (expression3 / condition-3)
    {statement-3};
.....
else if (expression-n / condition-n);
    {statement-n;}
else
    {statement-n+1;}

```

15

C_Prog. Spring 2012

- Pseudocode for a nested if...else statement

```

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

```
- ```

if (result >= 75) printf("Passed: Grade A\n");
else if (result >= 60) printf("Passed: Grade B\n");
 else if (result >= 45) printf("Passed: Grade C\n");
 else printf("Failed\n");

```
- All comparisons test a single variable called **result**. In other cases, each test may involve a different variable or some combination of tests

16

C\_Prog. Spring 2012

Develop a C code to provide a user with a division of two input numbers?

- **Analysis:**

- **Output:** Division\_Result;
- **Inputs:** Number1, Number2;
- **Process:**  
if Number2 == 0 Print "Can't Divide";  
else Division\_Result = Number1/Number2;

- **Algorithm:**

1. Start;
2. Read Number1, Number2;
3. if (Number2 == 0) Display "Can't Divide";  
else { Division\_Result = Number1/Number2;  
Print (Division\_Result); }
4. End.

17

C\_Prog. Spring 2012

```

/* Code Demonstrates the Use of if else block:
 To provide a division of two input numbers. */
#include <stdio.h>
main()
{
 int Number1, Number2, Div_Result;
 printf("\nEnter first number ");
 scanf("%d",&Number1);
 printf("\nEnter second number ");
 scanf("%d",&Number2);
 if (Number2 ==0)
 printf("\n\n Can't divide by zero\n\n");
 else
 { Div_Result = Number1/Number2;
 printf("\nAnswer:%d\n\n",Div_Result);
 } return 0;} //end of main function

```

18

C\_Prog. Spring 2012

**Example: Calculating a broker's commission:**

- The broker's commission is calculated as follow:

| Transaction size          | Commission rate      |
|---------------------------|----------------------|
| <b>Under \$2500</b>       | <b>\$30 + 1.7%</b>   |
| <b>\$2500 – \$6250</b>    | <b>\$56 + 0.66%</b>  |
| <b>\$6250 - \$20000</b>   | <b>\$76 + 0.34%</b>  |
| <b>\$20000 - \$50000</b>  | <b>\$100 + 0.22%</b> |
| <b>\$50000 - \$500000</b> | <b>\$155 + 0.11%</b> |
| <b>Over \$500000</b>      | <b>\$255 + 0.09%</b> |

The minimum charge is \$39.

- The Algorithm to Calculate a broker's commission:

**1. Start;**

**2. Read value;**

**3. What's the value of the transaction?**

19

C\_Prog. Spring 2012

```

if (value < 2500.00) commission = 30.00 + .017 * value;
else if (value < 6250.00)
 commission = 56.00 + .0066 * value;
else if (value < 20000.00)
 commission = 76.00 + .0034 * value;
else if (value < 50000.00)
 commission = 100.00 + .0022 * value;
else if (value < 500000.00)
 commission = 155.00 + .0011 * value;
else commission = 255.00 + .0009 * value;

```

**4. Check for the minimum charge (\$39)**

```

if (commission < 39.00)
 commission = 39.00;

```

**5. Print Commission**

**6. End.**

20

C\_Prog. Spring 2012

```

#include <stdio.h>
main()
{float commission, value;
 printf("Enter value of trade: ");
 scanf("%f", &value);
 if (value < 2500.00)
 commission = 30.00 + .017 * value;
 else if (value < 6250.00)
 commission = 56.00 + .0066 * value;
 else if (value < 20000.00)
 commission = 76.00 + .0034 * value;
 else if (value < 50000.00)
 commission = 100.00 + .0022 * value;
 else if (value < 500000.00)
 commission = 155.00 + .0011 * value;
 else commission = 255.00 +.0009* value;
 if (commission< 39.00)commission=39.00;
 printf("Commission:$.2f\n",commission);
 return 0; }

```

21

C\_Prog. Spring 2012

**RULE:** *else* clause **belongs to the nearest *if*** statement that hasn't been paired with an *else*;

One way to make the *else* clause part of the outer *if*

Statement: Enclose the inner *if* statement with braces

```

int a = 2;
int b = 2;
if (a == 1)
 if (b == 2)
 {
 printf("a was 1 and b was 2\n");
 }
else
 printf("a wasn't 1\n");

```

22

C\_Prog. Spring 2012

## A Ternary Operator

### "Conditional Expressions"

- C has many **unary** operators, and plenty of binary ones!
- In addition, C has a **ternary operator** (one that takes three operands.)

***cond ? exp<sup>T</sup> : exp<sup>F</sup>*** Read as →

***"if cond then exp<sup>T</sup> else exp<sup>F</sup>"***

- **Conditional Expression:**

- Consists of Symbols: ***?*** and ***:***
- Must be together → ***cond ? exp<sup>T</sup> : exp<sup>F</sup>***
- Similar to → ***if (cond) exp<sup>T</sup>***  
***else exp<sup>F</sup>***

23

C\_Prog. Spring 2012

How it works? (***cond ? exp<sup>T</sup> : exp<sup>F</sup>*** )

→ The ***?*** operator evaluates ***cond***:

→ If it is **true**, it evaluates ***exp<sup>T</sup>*** and returns it as the **value of the *?* expression**

→ If it is **false**, it evaluates ***exp<sup>F</sup>*** and returns it as the **value of the expression.**

#### Example 1:

```
int i, j, k;
```

```
i = 1;
```

```
j = 2;
```

```
k = i > j ? i : j; // or max of both (k is 2 now)
```

```
k = (i >= 0 ? i : 0) + j; // k is now 3 i.e (1+2)
```

24

C\_Prog. Spring 2012

Similar to:

```
if (i>j)k=i;
else k=j;
```

**Example 2:**

```
int y = 7;
```

```
int x;
```

```
x = ((y > 5) ? 1 : 0); /* since y is greater than
 5, x is assigned a value of 1 */
```

- Conditional statements tend to make programs **shorter** BUT **harder** to understand! Occasionally they are tempting to use. Instead of writing:

```
if (y > 5)
 x=1;
else
 x=0;
return x;
```

We may write:

```
return (y > 5 ? 1 : 0);
```

Also → `printf("%d/n", y > 5 ? 1 : 0);` is valid

## Switch Statement

- **Switch statement:** A construct that is used to replace deeply **nested or chained (Multiple) *if else*** statements.
- **Example:** Suppose that an ice cream store has asked us to write a program that will automate taking of the orders?
  - We will need to present a **menu** and
  - Based on the customer's choice take an **appropriate action.**

```

#include <stdio.h>
main()
{
 int choice;
 printf("What flavor ice cream do you want?\n");
 printf("Enter 1 for chocolate\n");
 printf("Enter 2 for vanilla\n");
 printf("Enter 3 for strawberry\n");
 printf("Enter 4 for green tea flavor, yuck\n");
 printf("Enter you choice: ");
 scanf("%d",&choice);
 if (choice==1){
 printf("Chocolate, good choice\n");
 }
 else if (choice==2) {printf("Vanillarific\n");}
 else if (choice==3) {printf("Berry Good\n");}
 else if (choice==4) {printf("Big Mistake\n");}
 else {printf("We don't have any.\n");
 printf("Make another selection.\n");}
 return 0;}

```

27

C\_Prog. Spring 2012

- This program will works fine, but the *if else* block is **cumbersome**.
- It would be easy, particularly if there were more *choices* and maybe *sub choices* involving more *if else's* to end up with program that doesn't perform the actions intended.
- The general form of a **switch** statement is:

```

switch (variable)
{
 case expression1: Action1; break;
 case expression2: Action2; break;

 default: default_Action;
}

```

- Here's the same program re-written with a **switch structure**.

28

C\_Prog. Spring 2012

```

#include <stdio.h>
main()
{
 int choice;
 printf("What flavor ice cream do want?\n");
 printf("Enter 1 for chocolate\n");
 printf("Enter 2 for vanilla\n");
 printf("Enter 3 for strawberry\n");
 printf("Enter 4 for green tea flavor\n");
 printf("Enter you choice: \n");
 scanf("%d",&choice);
 switch (choice)
 {
 case 1: printf("Chocolate, good choice\n");
 break;
 case 2: printf("Vanillarific\n");
 break;
 case 3: printf("Berry Good\n");
 break;
 case 4: printf("Big Mistake\n");
 break;
 default: printf("We don't have any.\n");
 printf("Make another selection.\n");
 }
 return 0;
}

```

29

C\_Prog. Spring 2012

### Notes:

- Each expression must be a *constant*.
- The variable is compared for equality against each expression.
- When an expression is found equal to the tested variable, execution continues until a break statement is encountered
- No need to use `{ }` around statements.
- It is possible to have a case without a break
- This causes execution to *fall through* into the next case!!

30

C\_Prog. Spring 2012

```

#include <stdio.h> //Another example
main()
{
char Grade;
printf("What's your grade?\n");
scanf("%c", &Grade);
switch(Grade)
{
case 'A': printf("Excellent"); break;
case 'B': printf("Good"); break;
case 'C': printf("OK"); break;
case 'D': printf("Mmmmm...."); break;
case 'F': printf("Do better than this");break;
default:printf("What's your grade anyway?");
}
return 0;
}

```

31

C\_Prog. Spring 2012

```

#include <stdio.h> //Another example
main()
{
float mark1, mark2, sum, avg;
int tens;
scanf("%f", &mark1);
scanf("%f", &mark2);
sum=mark1+mark2;
avg=sum/2;
tens=avg/10; //ignoring the fractions ! Tens is int
switch(tens)
{
case 10:
case 9: printf("Excellent"); break;
case 8: printf("Very Good"); break;
case 7: printf("Good"); break;
case 6: printf("Mmmmm...."); break;
case 5: printf("FAIL!"); break;
default: printf("What is your grade anyway?");
}
return 0; }

```

32

C\_Prog. Spring 2012

**Analyze, develop an Algorithm, draw a Flowchart, write a C Code, and run the following questions?**

**(No Need to submit the answers!)**

1. **Develop an interactive and efficient (FAST) C program to display the date entered (mm/dd/yyyy): 02/28/1955 into the following format: This is the 28<sup>th</sup> Day of February 1955? Try it for the 21<sup>st</sup> century date!**

33

2. **Develop an interactive and efficient C program to select one item from the following menu by Either entering the number, OR the number \* 10, OR the first character:**
  1. **Word Processing**
  2. **Spreadsheet**
  3. **Database**
  4. **Access**
  5. **Internet Explorer**
  6. **Outlook**
  7. **PowerPoint**
  8. **Visio**
  9. **Quit**

**Word processing could be accessed by:**

**w, W, 1, or 10!**

34

C\_Prog. Spring 2012