

Conditional Statements

Prepared by: Lec Tasmiah Tamzid Anannya, CS Dept, AIUB

Multiway if-else statements

```
if(expression 1)
    statement 1;
else if(expression 2)
    statement 2;
.
.
.
else if(expression n)
    statement n;
else
    other statement;
```

Example- calculate.c

- ▶ Create a calculator. Ask the user to give two numbers and then ask what s/he wants to do (add/sub/mul/div). According to the choice print the appropriate result.
- ▶ Suppose, ask the user to-
 - ▶ Press 1 for addition
 - ▶ Press 2 for subtraction
 - ▶ Press 3 for multiplication
 - ▶ Press 4 for division

Example-substance.c

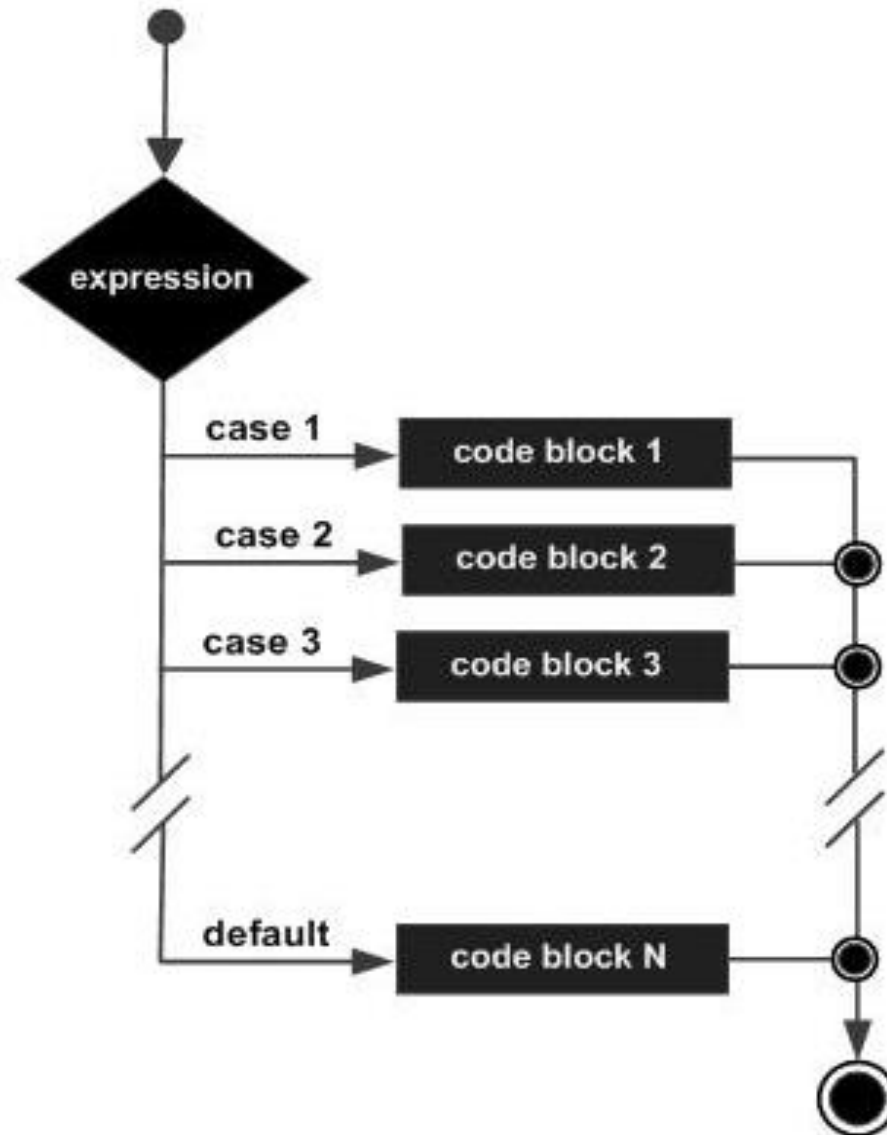
- ▶ Write a program that prompts the user to **input the boiling point** in degree Celsius.
- ▶ The program should **output the substance** corresponding to the boiling point listed in the table.
- ▶ The program should output the message “**substance unknown**” when it does not match any substance.

Substance	Boiling point
Water	100°C
Mercury	357°C
Copper	1187°C
Silver	2193°C
Gold	2660°C

switch statement

- ▶ **if** is good for choosing between two alternatives
- ▶ But it becomes cumbersome when several alternatives are needed.
- ▶ C's solution to the problem- **switch** statement
- ▶ A **switch** statement allows a variable to be tested for equality against a list of values.
- ▶ Each value is called a case, and the variable being switched on is checked for each **switch case**.

switch statement- flowchart



switch statement-syntax

switch (variable or an integer expression)

{

case constant1:

statement(s);

break;

case constant2:

statements(s);

break;

.

.

.

default: //optional

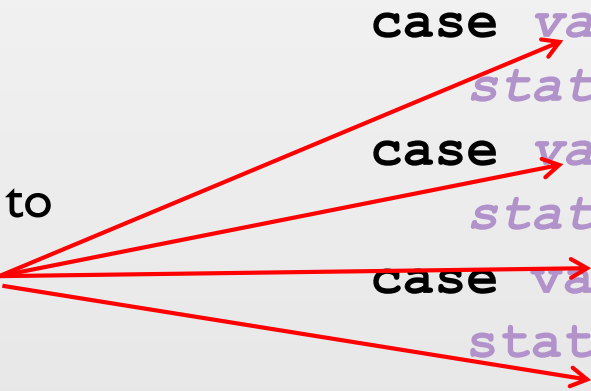
statement(s);

}

The Switch Statement

- ▶ The *switch statement* provides another way to decide which statement to execute next
- ▶ The `switch` statement evaluates an expression, then attempts to match the result to one of several possible cases
- ▶ The match must be an exact match.

```
switch ( expression ) {  
    case value1 :  
        statement-list1  
    case value2 :  
        statement-list2  
    case value3 :  
        statement-list3  
    case ...  
}
```



Switch - syntax

- The general syntax of a `switch` statement is:

`switch`
`and`
`case`
`are`
reserved
words

```
switch ( expression ) {  
    case value1 :  
        statement-list1  
    case value2 :  
        statement-list2  
    case value3 :  
        statement-list3  
    case ...  
}
```

If *expression*
matches *value3*,
control jumps
to here

The Switch Statement

- ▶ The *break* statement can be used as the last statement in each case's statement list
- ▶ A *break* statement causes control to transfer to the end of the `switch` statement
- ▶ If a *break* statement is not used, the flow of control will continue into the next case

```
switch ( expression ) {  
    case value1 :  
        statement-list1  
        break;  
    case value2 :  
        statement-list2  
        break;  
    case value3 :  
        statement-list3  
        break;  
    case ...  
}
```

switch statement-rules

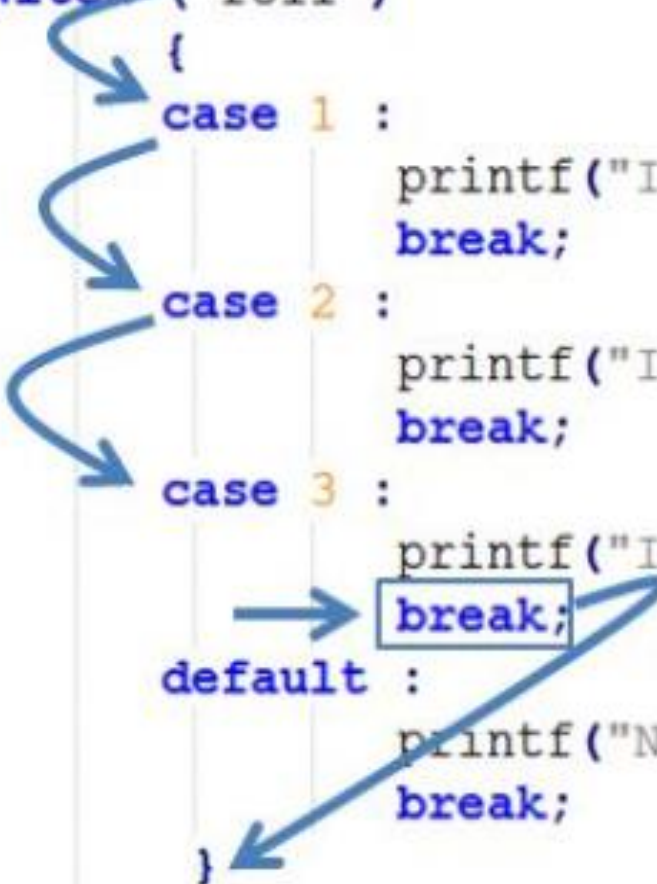
- ▶ You can have any number of case statements within a switch. Each case is followed by the value to be compared to and a colon.
- ▶ The value for a case must be the same data type as the variable in the switch.
- ▶ Duplicate case values are not allowed.
- ▶ When the variable being switched on is equal to a case, the statements following that case will execute until a **break** statement is reached.
- ▶ When a **break** statement is reached, the switch terminates, and the flow of control jumps to the next line following the switch statement.
- ▶ Not every case needs to contain a **break**. If no **break** appears, the flow of control will *fall through* to subsequent cases until a break is reached.

switch statement-rules

- ▶ A **switch** statement can have an optional **default** case, which must appear at the end of the switch. The default case can be used for performing a task when none of the cases is true. No **break** is needed in the default case.
- ▶ Nesting of switch statements are allowed, which means you can have switch statements inside another switch. However nested switch statements should be avoided as it makes program more complex and less readable.
- ▶ The expression of a **switch** statement must result in an **integral type**, meaning an integer (`byte, short, int, long`) or a `char`
- ▶ It **cannot** be a boolean value or a floating point value (`float` or `double`)

Example

```
int roll = 3 ;  
switch( roll )  
{  
    case 1 :  
        printf("I am Pankaj");  
        break;  
    case 2 :  
        printf("I am Nikhil");  
        break;  
    case 3 :  
        printf("I am John");  
        break;  
    default :  
        printf("No student found");  
        break;  
}
```



Guess the output?

```
#include <stdio.h>
int main()
{
    int i=2;
    switch (i)
    {
        case 1:
            printf("Case1 ");
        case 2:
            printf("Case2 ");
        case 3:
            printf("Case3 ");
        case 4:
            printf("Case4 ");
        default:
            printf("Default ");
    }
    return 0;
}
```

Guess the output?

```
#include <stdio.h>
int main()
{
    int i=2;
    switch (i)
    {
        case 1:
            printf("Case1 ");
        case 2:
            printf("Case2 ");
        case 3:
            printf("Case3 ");
        case 4:
            printf("Case4 ");
        default:
            printf("Default ");
    }
    return 0;
}
```

Case2 Case3 Case4 Default

How to avoid this situation?

- ▶ We can use break statement to break the flow of control after every case block.
- ▶ Break statements are useful when you want your program-flow to come out of the switch body.
- ▶ Whenever a break statement is encountered in the switch body, the control comes out of the switch case statement.

Solution

```
#include <stdio.h>
int main()
{
    int i=2;
    switch (i)
    {
        case 1:
            printf("Case1 ");
            break;
        case 2:
            printf("Case2 ");
            break;
        case 3:
            printf("Case3 ");
            break;
        case 4:
            printf("Case4 ");
            break;
        default:
            printf("Default ");
    }
    return 0;
}
```

Why didn't I use break statement after default?

- ▶ The control would itself come out of the switch after default so I didn't use it.
- ▶ However if you want to use the break after default then you can use it, there is no harm in doing that.

switch statement

- ▶ Case doesn't always need to have order 1, 2, 3 and so on. They can have any integer value after case keyword.
- ▶ Also, case doesn't need to be in an ascending order always, you can specify them in any order as per the need of the program.
- ▶ You can also use characters in switch case.

Example

```
char ch='b';
switch (ch)
{
    case 'd':
        printf("CaseD ");
        break;
    case 'b':
        printf("CaseB");
        break;
    default:
        printf("Default ");
}
```

Do it with switch case

- ▶ Create a calculator. Ask the user to give two numbers and then ask what s/he wants to do (add/sub/mul/div). According to the choice print the appropriate result.
- ▶ Suppose, ask the user to-
 - ▶ Press 1 for addition
 - ▶ Press 2 for subtraction
 - ▶ Press 3 for multiplication
 - ▶ Press 4 for division

Assignment1-restaurant.c

- ▶ Using Switch statement, write a program that displays the following menu for the food items available to take order from the customer:

- B= Burger
- F= French Fries
- P= Pizza
- S= Sandwiches

The program inputs the type of food and quantity. It finally displays the total charges for the order according to following criteria:

- Burger = Rs. 200
- French Fries= Rs. 50
- Pizza= Rs. 500
- Sandwiches= Rs. 150

Nested if-else

- ▶ When an if statement is target of another if or else, it is said to be nested within the outer if.

```
If(condition)  //outer if
    {statement(s);}
else
    {
        if(condition)
            {statement(s);}
        else
            {statement(s);}
    }
```

Example- maximum.c

- ▶ Write a c code to find the largest number among three numbers.