5. Selection

Agenda

- Introduction
- Logical Data and Operators
- Two-Way Selection
- Multi-Way Selection
- More Standard Functions

Introduction

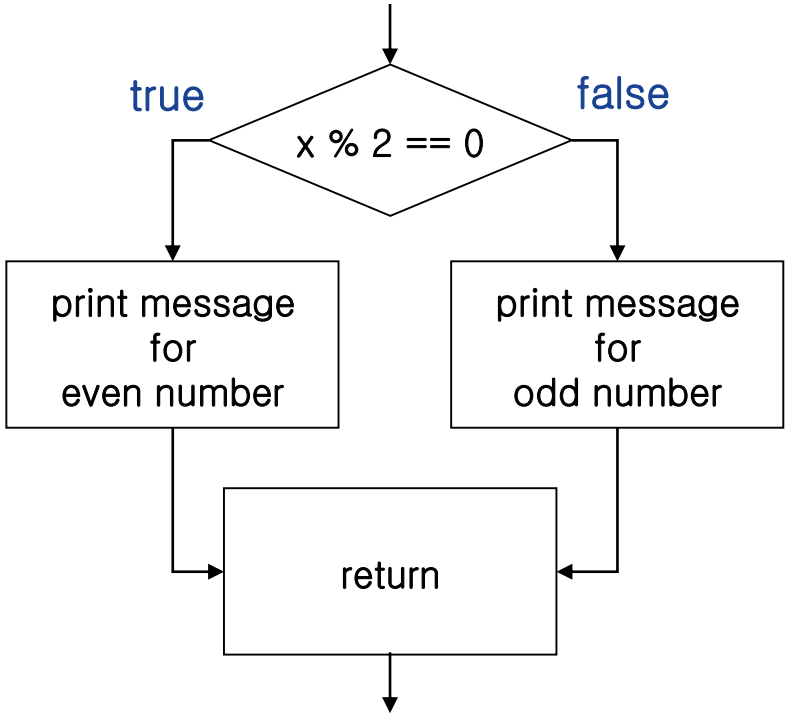
- Control statements: statements to control execution order
- Any program could be written with three constructs [Dijkstra]
  - Sequence
  - Selection
  - Loop (Repetition)

Introduction

- An example of selection
  - Deciding if a number is odd (1,3,5,…) or even (0,2,4,…)

```
#include <stdio.h>

int main()
{
   int x = 0;

   scanf("%d", &x);

   if( x % 2 == 0 )
      printf("%d is an even number₩n", x);
   else
      printf("%d is an odd number₩n", x);

   return 0;
}
```

(logic) expression



Comparative Operators

Type	Operator	Meaning	Precedence
Relational	<	less than	10
	<=	less than or equal	
	>	greater than	
	>=	greater than or equal	
Equality	==	equal	9
	!=	not equal	

Precedence and Associativity

Operators	Associativity
() [] -> .	left to right
! ~ ++ -- + - * & (type) sizeof	right to left
* / %	left to right
+ -	left to right
<< >>	left to right
< <= > >=	left to right
== !=	left to right
^	left to right
|	left to right
&&	left to right
||	left to right
?:	right to left
= += -= *= /= %= &= ^= |= <<= >>=	right to left
,	left to right

Combining Logic Expressions

- Comfortable temperature: 15 ~ 25.

```
#include <stdio.h>
int main()
{
   int temperature = 0;

   printf(“Enter current temperature : ”);
   scanf(“%d”, &temperature);

   if(                                                    )
      printf(“Good!₩n”, x);          // temperature is between 15 and 25
   else
      printf(“Bad!₩n”, x);           // otherwise

   return 0;
}
```

Logical Operators

- Logical operators
  - Not operator: !
  - And operator: &&
  - Or operator: ||

not

x	!x
false	true
true	false

and

x	y	x&&y
false	false	false
false	true	false
true	false	false
true	true	true

or

x	y	x||y
false	false	false
false	true	true
true	false	true
true	true	true

Precedence and Associativity

Operators	Associativity
() [] -> .	left to right
! ~ ++ -- + - * & (type) sizeof	right to left
* / %	left to right
+ -	left to right
<< >>	left to right
< <= > >=	left to right
== !=	left to right
^	left to right
|	left to right
&&	left to right
||	left to right
?:	right to left
= += -= *= /= %= &= ^= |= <<= >>=	right to left
,	left to right

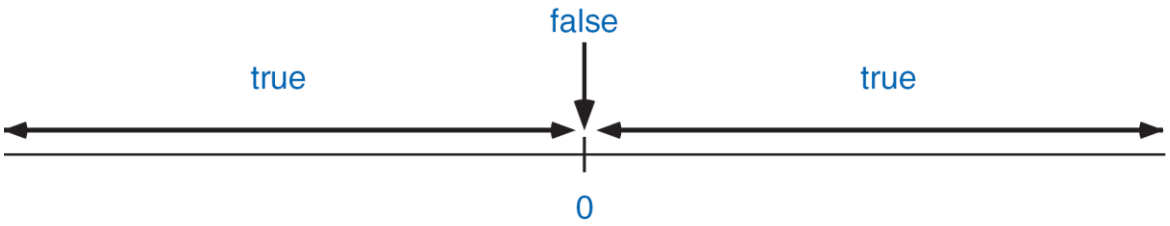
Example

- Decide whether current temperature is suitable.
  - If current temperature is below 15C, it's cold
  - If current temperature is between 15C and 25C, it's mild
  - If current temperature is higher than 25C, it's hot.

```
int tem = 0;

scanf("%d₩n", &tem);
if(tem < 15)
   printf("It's cold₩n");
if(tem >= 15 && tem <= 25)
   printf("It's mild₩n");
if(tem > 25)
   printf("It's hot₩n");
```

Logical Data in C

- Logical data: true / false
  - C99: bool type, { true, false }
  - Traditional C: int
    - False: 0
    - True: otherwise



Example

```
#include <stdio.h>

int IsGoodTemperature(int cur_temp);

main()
{
    int temp = 0;

    printf("Input current temperature : ");
    scanf("%d", &temp);

    if(IsGoodTemperature(temp)){
        printf("Good!₩n");
    } else {
        printf("Bad!₩n");
    }

}
```

```
int IsGoodTemperature(int cur_temp)
// if cur_temp is between 15 and 25
// return true, otherwise, return false
{
    if(cur_temp >= 15 && cur_temp <= 25)
        return   ;          // true
    else
        return   ;          // false
}
```

Example

```
#include <stdio.h>
#include <stdbool.h>

int IsGoodTemperature(int cur_temp);

main()
{
    int temp = 0;

    printf("Input current temperature : ");
    scanf("%d", &temp);

    if(IsGoodTemperature(temp)){
        printf("Good!₩n");
    } else {
        printf("Bad!₩n");
    }

}
```

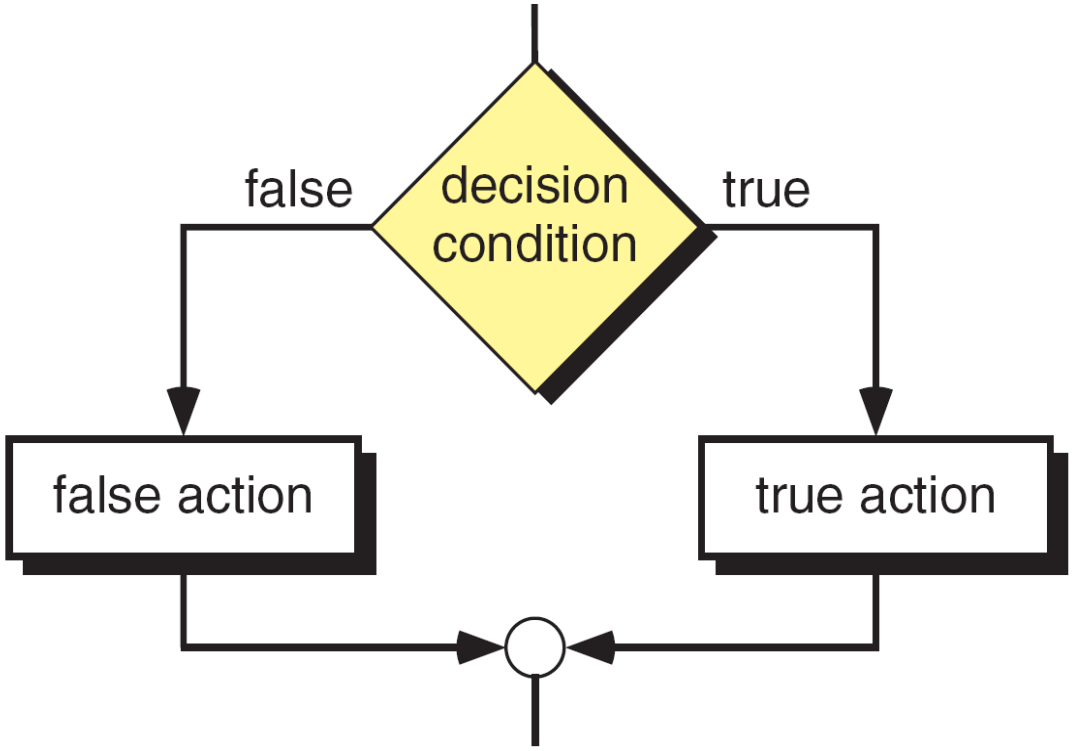
```
int IsGoodTemperature(int cur_temp)
// if cur_temp is between 15 and 25
// return true, otherwise, return false
{
    if(cur_temp >= 15 && cur_temp <= 25)
        return true;        // true
    else
        return false;       // false
}
```

Agenda

- Introduction
- Logical Data and Operators
- Two-Way Selection
- Multi-Way Selection
- More Standard Functions

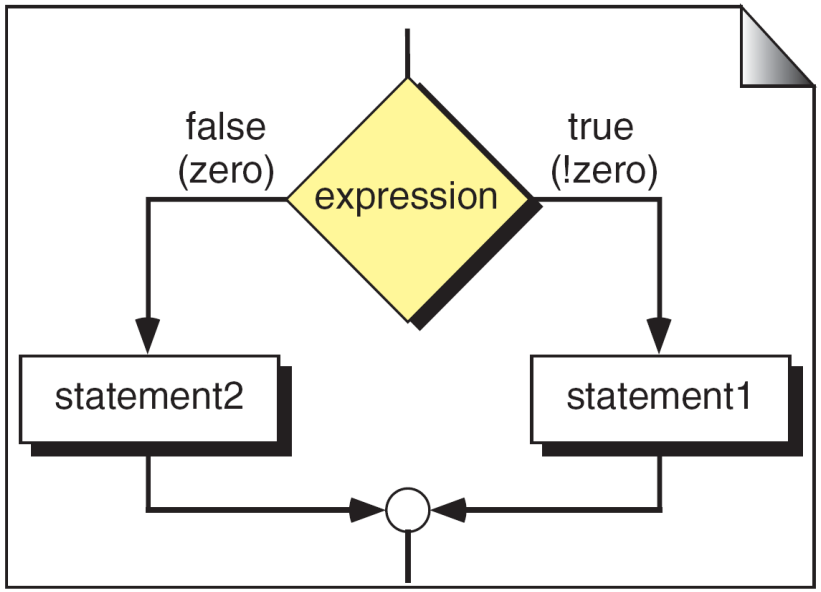
Two-Way Selection

- Control path is decided by decision condition



if … else

- If *expression* is true, *statement1* is executed. Otherwise *statement2* is executed



(a) Logical Flow

```
if (expression)
    statement1
else
    statement2
```

else part can be omitted!

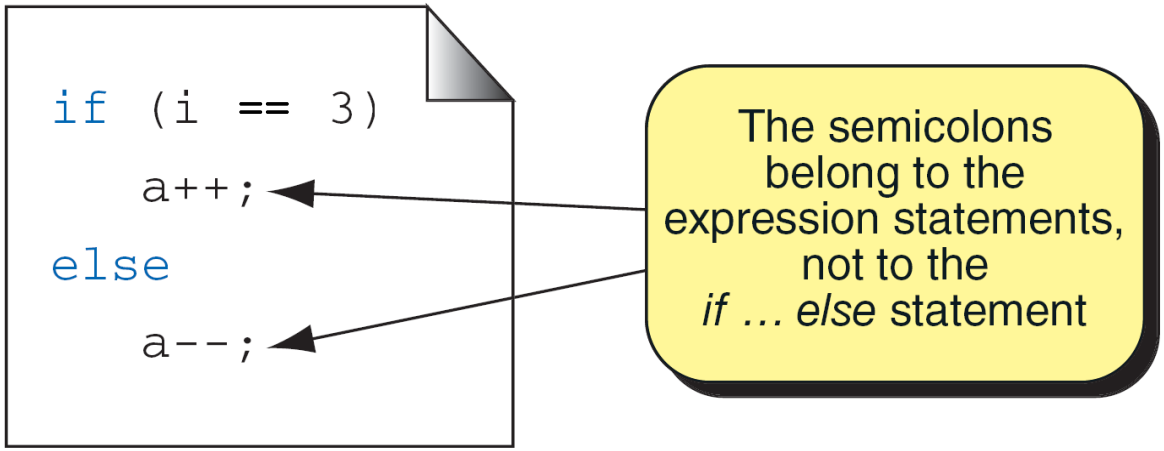
(b) Code

if ··· else

1. The expression must be enclosed in parentheses.
2. No semicolon (;) is needed for an *if…else* statement; statement 1 and statement 2 may have a semicolon as required by their types.
3. The expression can have a side effect.
4. Both the true and the false statements can be any statement (even another *if…else* statement) or they can be a null statement.
5. Both statement 1 and statement 2 must be one and only one statement. Remember, however, that multiple statements can be combined into a compound statement through the use of braces.
6. We can swap the position of statement 1 and statement 2 if we use the complement of the original expression.

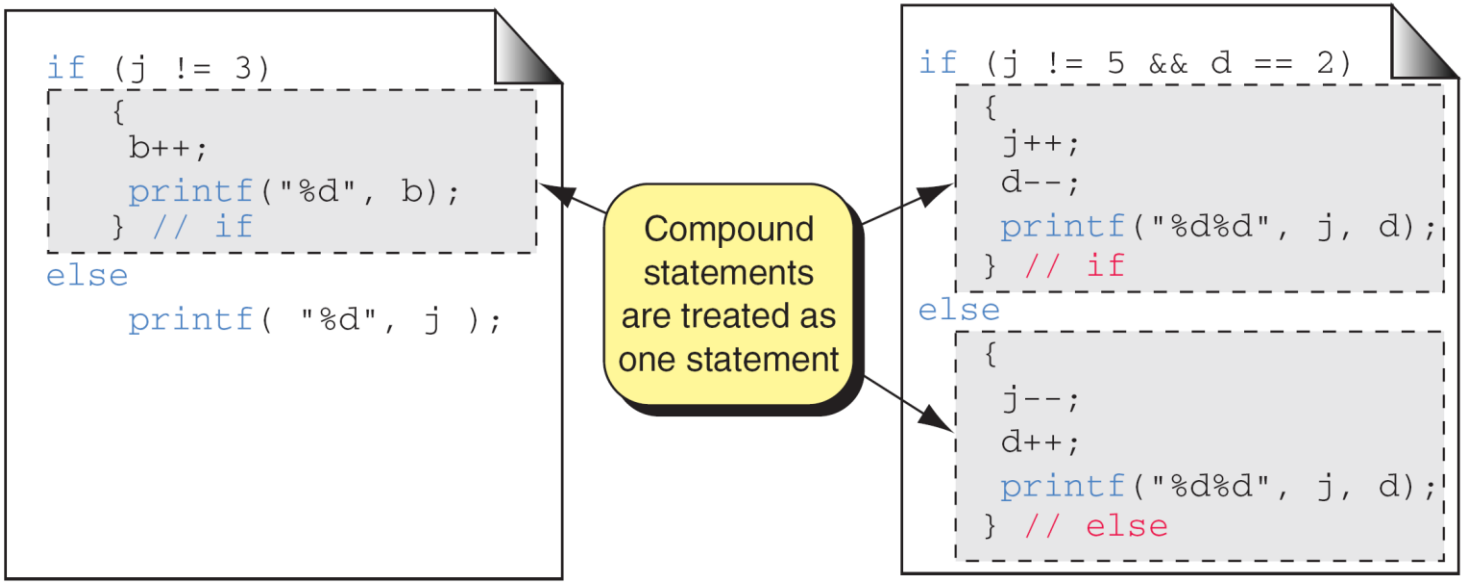
if … else

- Semicolon is not necessary for if statement



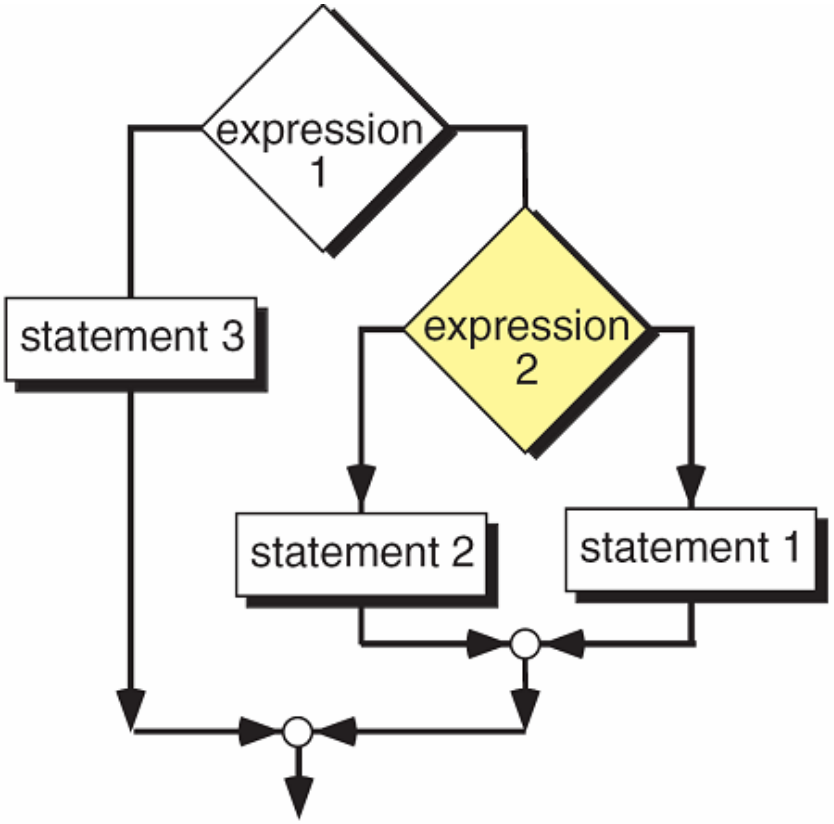
if … else

- Statement1 and statement2 can be compound statements



Nested if Statements

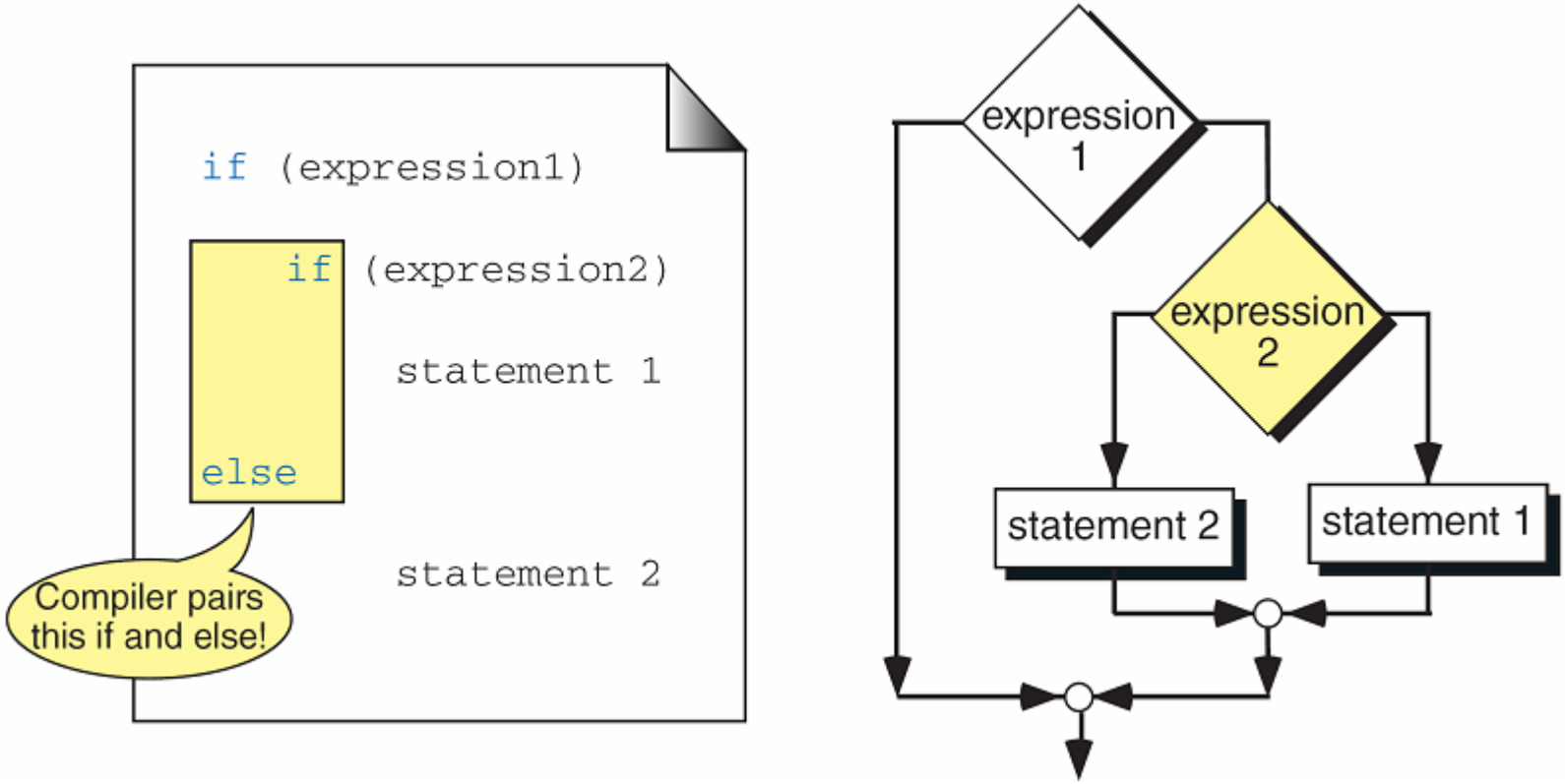
- *if* statement can include another *if* statement




```
if (expression 1)
    if (expression 2)
        statement 1
    else
        statement 2
else
    statement 3
```

Dangling else Problem

- *else* is always paired with the most recent unpaired *if*.



Recommendations

- Make code easy to read by indentation
- Clarify meaning of *if else* statement by braces even if it is not really necessary.

necessary

```
if(expression1){
   if(expression2)
      statement1
}
else
      statement2
```

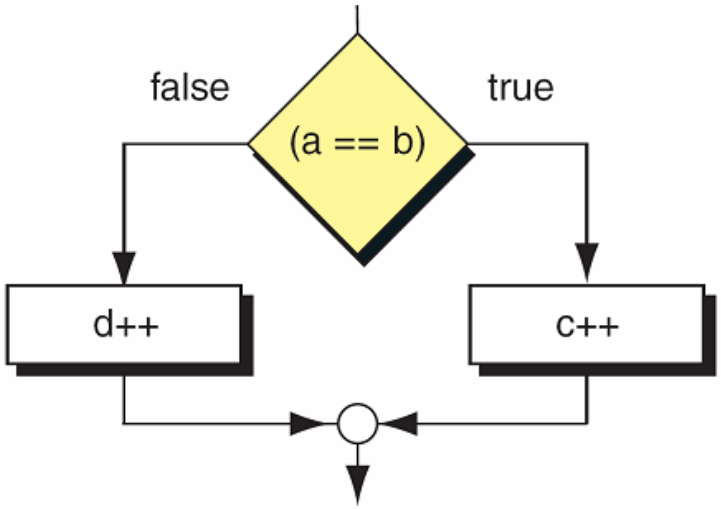
not necessary,
but recommendable

```
if(expression1){
   if(expression2)
      statement1
   else
      statement2
}
```

Conditional Expression

- Conditional expression
  - Syntax: logic_expression ? expression1 : expression2

    Ex) a == b ? c++ : d++;



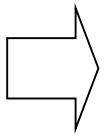

```
a == b ? c++ : d++;
```

Conditional Expression

- Conditional expression simplifies program

complex

```
int IsEvenNumber(int x)
{
   if(x % 2 == 0)
      return true;
   else
      return false;
}
```



simple

```
int IsEvenNumber(int x)
{
   return x % 2 == 0 ? true :  false;
}
```

even better

```
int IsEvenNumber(int x)
{
   return (x % 2 == 0) ? true : false;
}
```

Precedence and Associativity

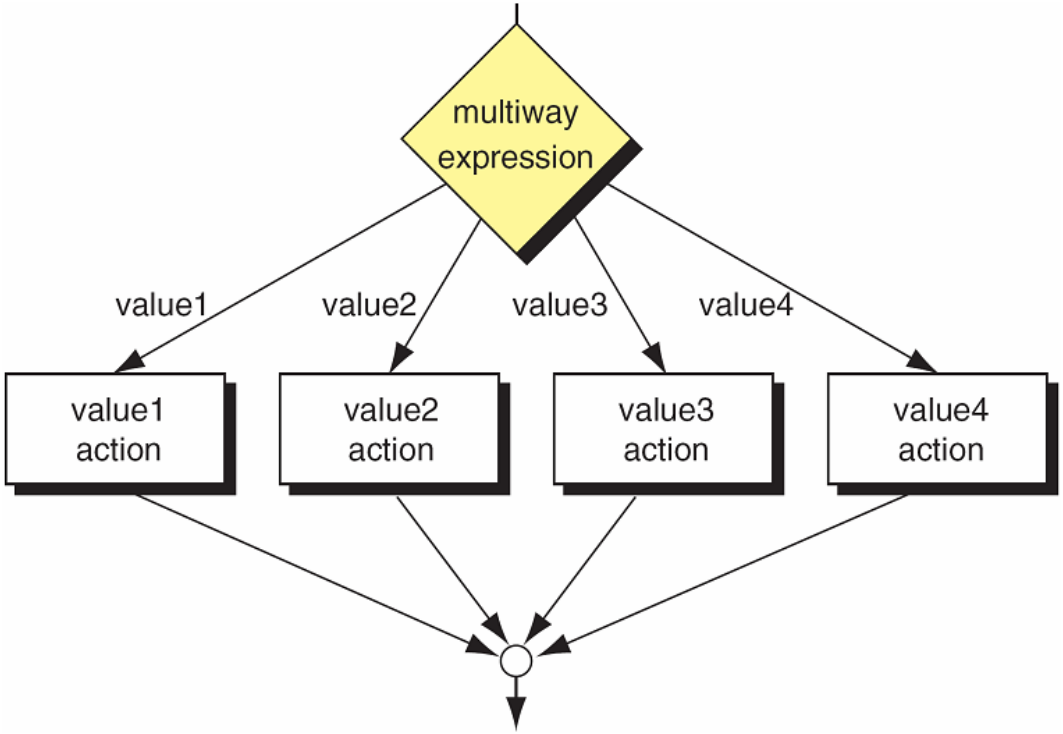
Operators	Associativity
() [] -> .	left to right
! ~ ++ -- + - * & (type) sizeof	right to left
* / %	left to right
+ -	left to right
<< >>	left to right
< <= > >=	left to right
== !=	left to right
^	left to right
|	left to right
&&	left to right
||	left to right
?:	right to left
= += -= *= /= %= &= ^= |= <<= >>=	right to left
,	left to right

Agenda

- Introduction
- Logical Data and Operators
- Two-Way Selection
- **Multi-Way Selection**
- More Standard Functions

Multiway Selection

- Multiway selection: selection among several alternatives



Multiway Selection

- Example: Calculator

  Perform addition, subtraction, multiplication, division according to operator

```
if(op == '+')
   result = num1 + num2;
else if(op == '-')
   result = num1 - num2;
else if(op == '*')
   result = num1 * num2;
else if(op == '/')
   result = num1 / num2;
else
   printf("Operator you entered is not valid. Try again₩n");
```

```
if(op == '+')
   result = num1 + num2;
if(op == '-')
   result = num1 - num2;
if(op == '*')
   result = num1 * num2;
if(op == '/')
   result = num1 / num2;
```

Multiway Selection

- Example: Calculator

  Perform addition, subtraction, multiplication, division according to operator

```
if(op == '+')
   result = num1 + num2;
else if(op == '-')
   result = num1 - num2;
else if(op == '*')
   result = num1 * num2;
else if(op == '/')
   result = num1 / num2;
```

```
switch(op){
case '+':
      result = num1 + num2;
      break;
case '-':
      result = num1 - num2;
      break;
case '*':
      result = num1 * num2;
      break;
case '/':
      result = num1 / num2;
      break;
}
```

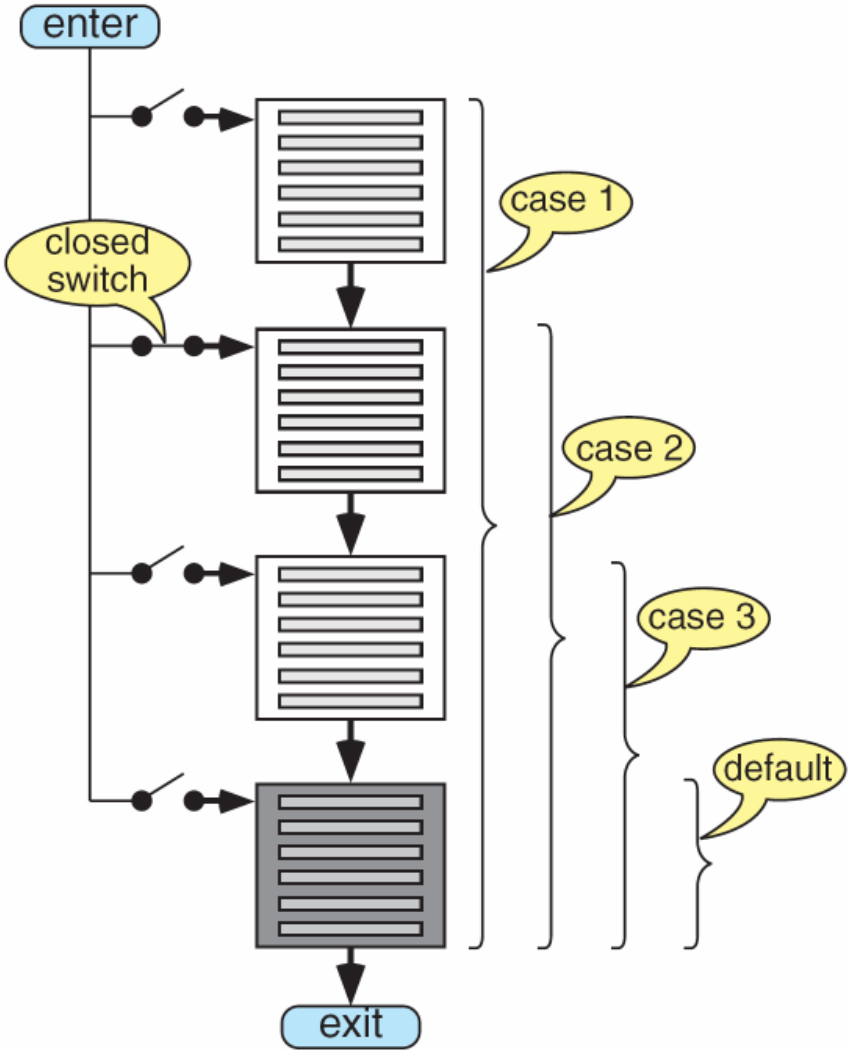
switch Statement

- Switch: composite statement for multiway selection
  - Syntax:

```
switch (expression)
  {
   case constant-1: statement
                    ⋮
                    statement
   case constant-2: statement
                    ⋮
                    statement
   case constant-n: statement
                    ⋮
                    statement
   default        : statement
                    ⋮
                    statement
  } // end switch
```

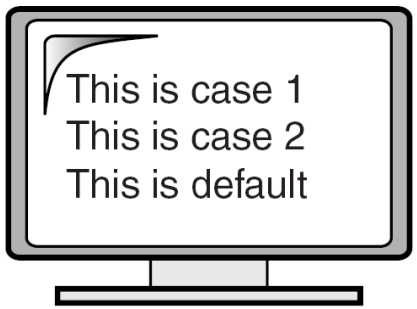
switch Statement

- *expression* in switch statement: any expression that reduces to an integral value
- Selection alternatives are specified by *case* label
- If there is no case values matches the value, *default* label is selected
  - *default* label can be omitted

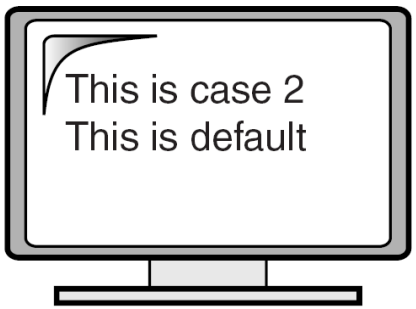


switch Statement

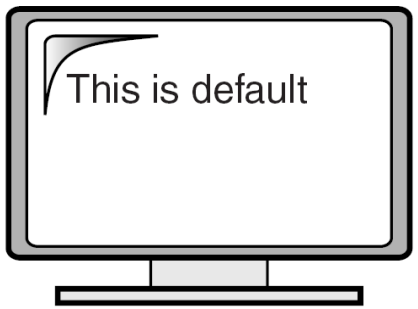
```
1  // Program fragment to demonstrate switch
2     switch (printFlag)
3        {
4         case 1:  printf("This is case 1\n");
5
6         case 2:  printf("This is case 2\n");
7
8         default: printf("This is default\n");
9        } // switch
```



(a) printFlag is 1



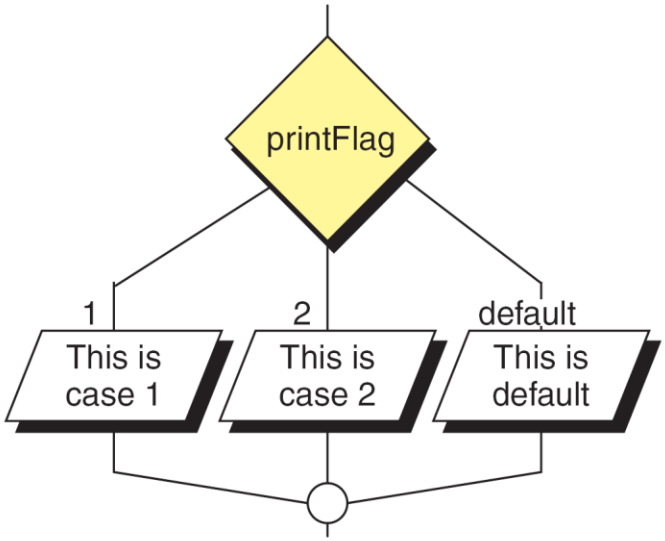
(b) printFlag is 2



(c) printFlag is not 1 or 2

switch Statement

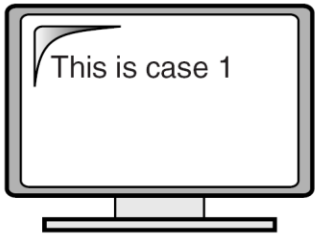
- *break* statement causes the program to jump out of switch statement

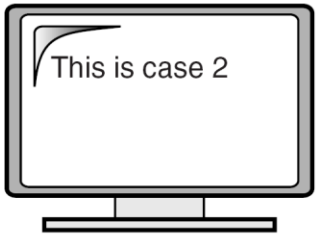


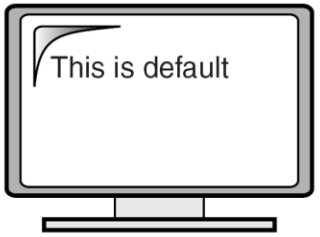
(a) Logic Flow

```
switch (printFlag)
  {
   case 1:
            printf
              ("This is case 1");
            break;
   case 2:
            printf
              ("This is case 2");
            break;
   default:
            printf
              ("This is default");
            break;
  }  // switch
```

(b) Code







Multiway Selection

- Example: Calculator

  Perform addition, subtraction, multiplication, division according to operator

```
if(op == '+')
   result = num1 + num2;
if(op == '-')
   result = num1 - num2;
if(op == '*')
   result = num1 * num2;
if(op == '/')
   result = num1 / num2;
```

```
switch(op){
case '+':
      result = num1 + num2;

case '-':
      result = num1 - num2;

case '*':
      result = num1 * num2;

case '/':
      result = num1 / num2;

}
```

switch Statement

- Two or more case labels may be associated with the same set of actions

```
1  /* Program fragment that demonstrates multiple
2     cases for one set of statements
3  */
4  switch (printFlag)
5     {
6      case 1:
7      case 3:  printf("Good Day\n");
8               printf("Odds have it!\n");
9               break;
10     case 2:
11     case 4:  printf("Good Day\n");
12              printf("Evens have it!\n");
13              break;
14     default: printf("Good Day, I'm confused!\n");
15              printf("Bye!\n");
16              break;
17    } // switch
```

Summary of switch Statement

1. The control expression that follows the keyword *switch* must be an integral type.
2. Each *case* label is the keyword *case* followed by a constant expression.
3. No two *case* labels can have the same constant expression value.
4. But two *case* labels can be associated with the same set of actions.
5. The *default* label is not required. If the value of the expression does not match with any labeled constant expression, the control transfers outside of the *switch* statement. However, we recommend that all *switch* statements have a *default* label.
6. The *switch* statement can include at most one *default* label. The *default* label may be coded anywhere, but it is traditionally coded last.

Example: Student Grading

- Convert a numeric score to a letter grade
  - score >= 90: A
  - score >= 80: B
  - score >= 70: C
  - score >= 60: D
  - score < 60: F

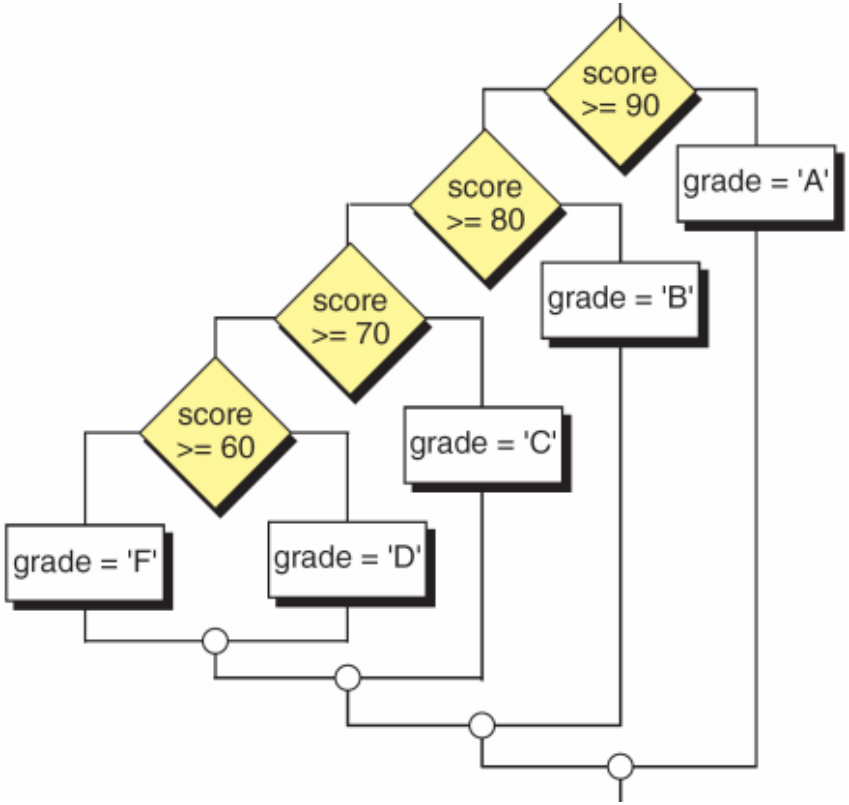
Example: Student Grading

```
char ScoreToGrade(int score)
{
   char grade = 0;
   switch(              ){









}
   return grade;
}
```

else if Statement

- Alternative multiway selection: *else if*
  - Concatenation of selection
  - Non-integral expression is allowed

```
if(score >= 90)
   grade = ‘A’;
else if(score >= 80)
   grade = ‘B’;
else if(score >= 70)
   grade = ‘C’;
else if(score >= 60)
   grade = ‘D’;
else
   grade = ‘F’;
```

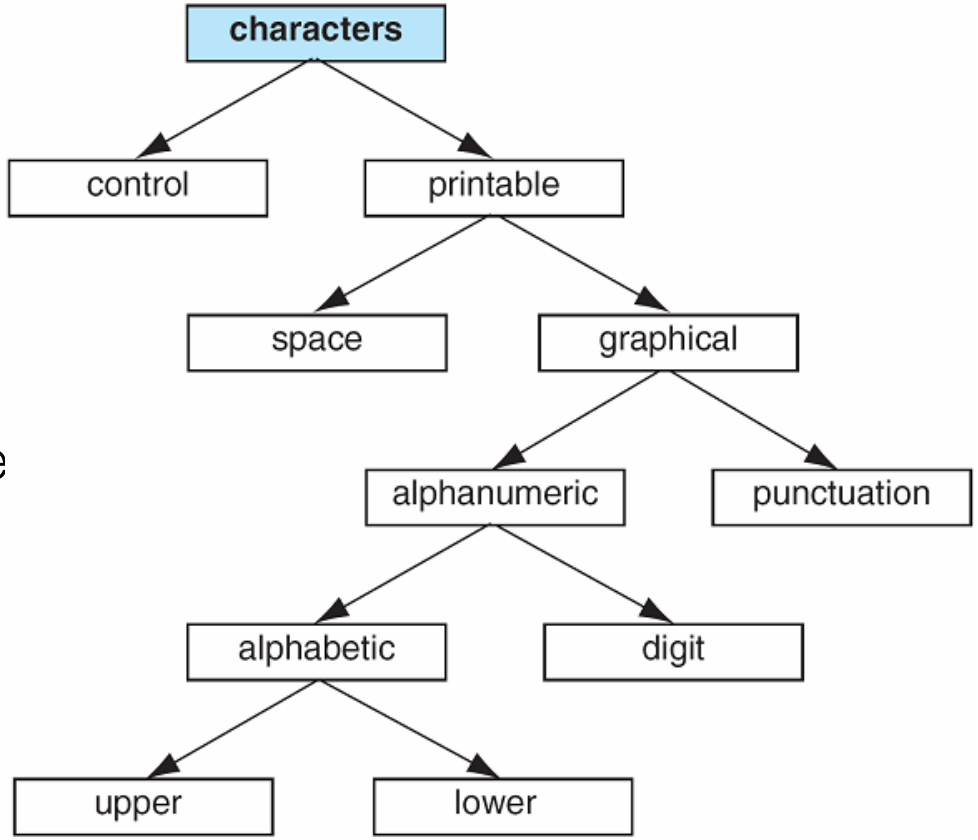


Agenda

- Introduction
- Logical Data and Operators
- Two-Way Selection
- Multi-Way Selection
- More Standard Functions

Character Library

- Character library
  - Classifying functions
  - Converting functions
- Include files
  - ctype.h – char type
  - wctype.h – wchar_t type



Classifying Functions

- Classifying functions: examine a character and tell if it belongs to a given classification
  - General format
    - int is…(int testChar); // for char type
    - int isw…(int testChar); // for wchar_t type

    Ex) isdigit, isupper, isgraph, …

Classifying Functions

Function	Description
iscntrl	Control characters
isprint	Printable character, that is character with an assigned graphic
isspace	Whitespace character: space character (32), horizontal tab (9), line feed (10), vertical tab (11), form feed (12), and carriage return (13)
isgraph	Character with printable graphic; all printable characters except space
isalnum	Alphanumeric: any alphabetic or numeric character
ispunct	Any graphic character that is not alphanumeric
isalpha	Any alphabetic character, upper- or lowercase
isupper	Only uppercase alphabetic
islower	Only lowercase alphabetic
isdigit	Decimal digits (0...9)
isxdigit	Hexadecimal digits (0...9, a...f, A...F)
isodigit	Octal digits (0...7)

Conversion Functions

- Conversion functions: convert a character from one case to another
  - General format
    - int to…(int oldChar); // for char type
    - int tow…(int oldChar); // for wchar_t type

Function	Description
toupper	Converts lower- to uppercase. If not lowercase, returns it unchanged.
tolower	Converts upper- to lowercase. If not uppercase, returns it unchanged.

Ex) tolower('A') = 97 // == 'a'

Program Termination

- Termination functions
  - Normal termination: terminates execution regardless of where it is executed
    - void exit(int terminationStatus);
  - Abnormal termination (error case): immediately terminates execution and send a signal of accident to OS.
    - void abort(void);
- Include file: stdlib.h