

Lecture 3

Control Structures

Selection: `else/if` and `switch`

2.1 Introduction

- Before writing a program
 - Have a thorough understanding of problem
 - Carefully plan your approach for solving it
- While writing a program
 - Know what “building blocks” are available
 - Use good programming principles

2.2 Algorithms

- Computing problems
 - Solved by executing a series of actions in a specific order
- Algorithm is a procedure determining
 - Actions to be executed
 - Order to be executed
 - Example: recipe
- Program control
 - Specifies the order in which statements are executed

2.3 Pseudocode

- Pseudocode
 - Artificial, informal language used to develop algorithms
 - Similar to everyday English
- Not executed on computers
 - Used to think out program before coding
 - Easy to convert into C++ program
 - Only executable statements
 - No need to declare variables

2.4 Control Structures

- Sequential execution
 - Statements executed in order
- Transfer of control
 - Next statement executed *not* next one in sequence
- 3 control structures (Bohm and Jacopini)
 - Sequence structure
 - Programs executed sequentially by default
 - Selection structures
 - `if`, `if/else`, `switch`
 - Repetition structures
 - `while`, `do/while`, `for`

2.4 Control Structures

■ C++ keywords

- Cannot be used as identifiers or variable names

C++ Keywords

*Keywords common to the
C and C++ programming
languages*

auto	break	case	char	const
continue	default	do	double	else
enum	extern	float	for	goto
if	int	long	register	return
short	signed	sizeof	static	struct
switch	typedef	union	unsigned	void
volatile	while			

C++ only keywords

asm	bool	catch	class	const_cast
delete	dynamic_cast	explicit	false	friend
inline	mutable	namespace	new	operator
private	protected	public	reinterpret_cast	
static_cast	template	this	throw	true
try	typeid	typename	using	virtual
wchar_t				

2.4 Control Structures

■ Flowchart

- Graphical representation of an algorithm
- Special-purpose symbols connected by arrows (flowlines)
- Rectangle symbol (action symbol)
 - Any type of action
- Oval symbol
 - Beginning or end of a program, or a section of code (circles)

■ Single-entry/single-exit control structures

- Connect exit point of one to entry point of the next
- Control structure stacking

2.5 if Selection Structure

- Selection structure
 - Choose among alternative courses of action
 - Pseudocode example:
 - If student's grade is greater than or equal to 60*
 - Print "Passed"*
 - If the condition is **true**
 - Print statement executed, program continues to next statement
 - If the condition is **false**
 - Print statement ignored, program continues
 - Indenting makes programs easier to read
 - C++ ignores whitespace characters (tabs, spaces, etc.)

2.5 if Selection Structure

- Translation into C++

If student's grade is greater than or equal to 60

Print "Passed"

```
if ( grade >= 60 )  
    cout << "Passed";
```

- Diamond symbol (decision symbol)

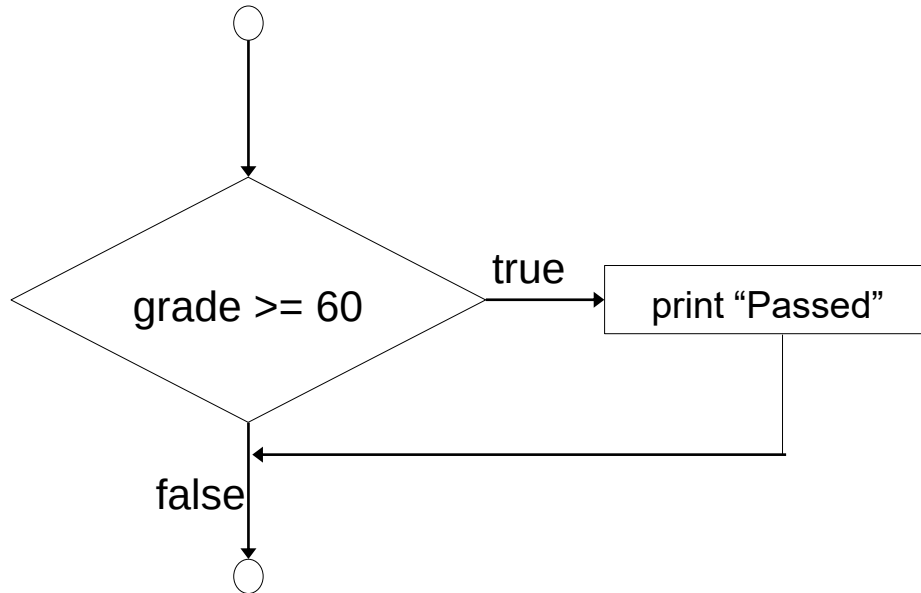
- Indicates decision is to be made
- Contains an expression that can be true or false
 - Test condition, follow path

- **if** structure

- Single-entry/single-exit

2.5 if Selection Structure

■ Flowchart of pseudocode statement



A decision can be made on any expression.

zero - **false**

nonzero - **true**

Example:

3 - 4 is true

2.6 if/else Selection Structure

- **if**

- Performs action if condition true

- **if/else**

- Different actions if conditions true or false

- Pseudocode

*if student's grade is greater than or equal to 60
 print "Passed"*

else

print "Failed"

- C++ code

```
if ( grade >= 60 )  
    cout << "Passed";  
else  
    cout << "Failed";
```

2.6 if/else Selection Structure

- Nested **if/else** structures

- One inside another, test for multiple cases
- Once condition met, other statements skipped

if student's grade is greater than or equal to 90

Print "A"

else

if student's grade is greater than or equal to 80

Print "B"

else

if student's grade is greater than or equal to 70

Print "C"

else

if student's grade is greater than or equal to 60

Print "D"

else

Print "F"

2.6 if/else Selection Structure

■ Example

```
if ( grade >= 90 )           // 90 and above
    cout << "A";
else if ( grade >= 80 )      // 80-89
    cout << "B";
else if ( grade >= 70 )      // 70-79
    cout << "C";
else if ( grade >= 60 )      // 60-69
    cout << "D";
else                          // less than 60
    cout << "F";
```

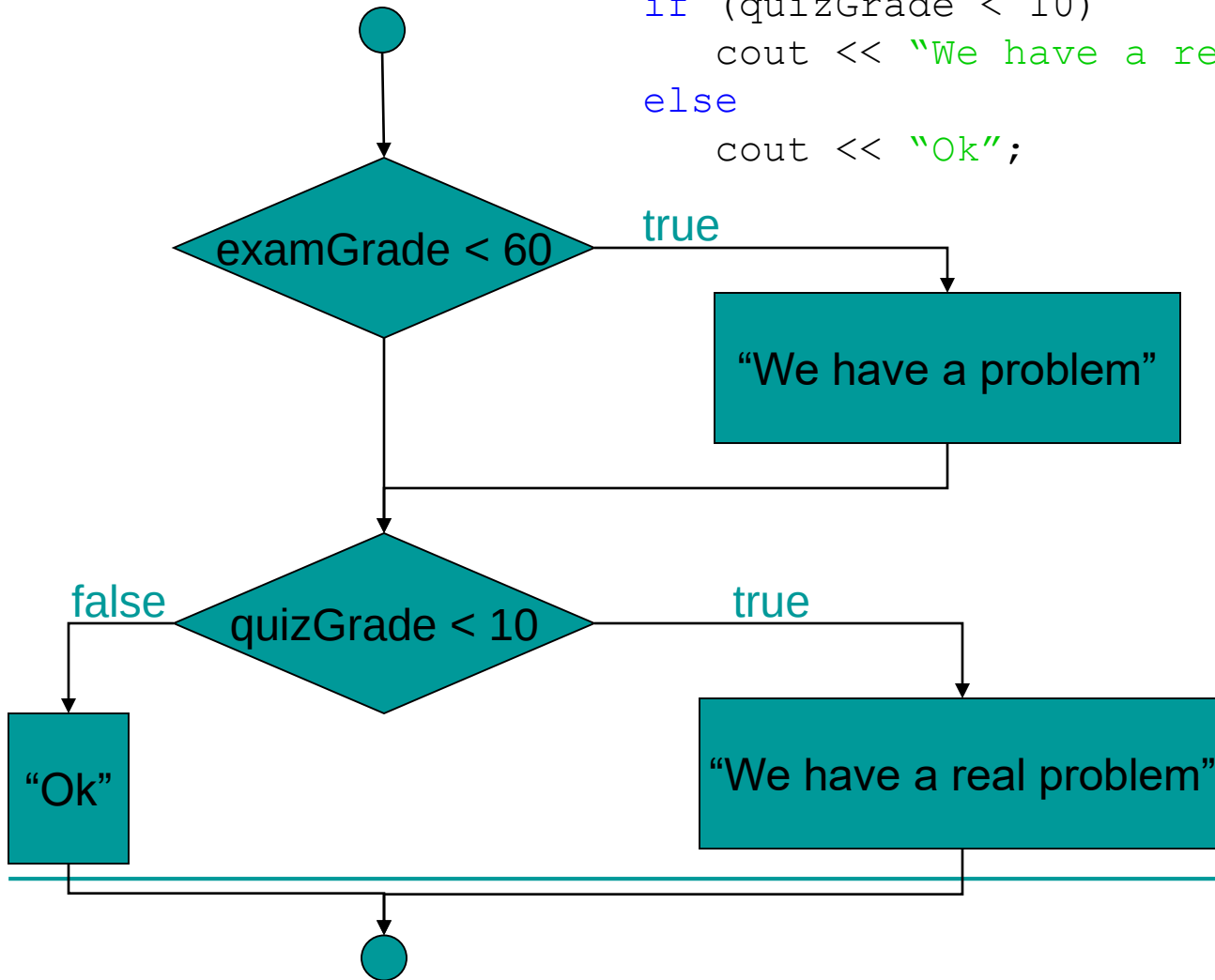
Importance of Curly Braces

- Print “We have a problem” if examGrade < 60
- Print “We have a real problem” if examGrade < 60 and quizGrade < 10
- Print “Ok” if examGrade >= 60

```
int examGrade, quizGrade;  
if (examGrade < 60)  
    cout << "We have a problem" << endl;  
if (quizGrade < 10)  
    cout << "We have a real problem" << endl;  
else  
    cout << "Ok";
```

Exam Grade Flowchart

```
int examGrade, quizGrade;  
if (examGrade < 60)  
    cout << "We have a problem" << endl;  
if (quizGrade < 10)  
    cout << "We have a real problem" << endl;  
else  
    cout << "Ok";
```



Writing Cases

- Print “We have a problem” if examGrade < 60
- Print “We have a real problem” if examGrade < 60 and quizGrade < 10
- Print “Ok” if examGrade >= 60

	examGrade < 60	quizGrade < 10	Action
Case 1	true	false	“We have a problem”
Case 2	true	true	“We have a problem” and “We have a real problem”
Case 3	false	true/false	“Ok”

Putting it all together

	examGrade < 60	quizGrade < 10	Action
Case 1	true	false	"We have a problem"
Case 2	true	true	"We have a problem" and "We have a real problem"
Case 3	false	true/false	"Ok"

```
int examGrade, quizGrade;
if (examGrade < 60) {
    cout << "We have a problem" << endl;
    if (quizGrade < 10)
        cout << "We have a real problem" << endl;
}
else
    cout << "Ok";
```

boolean Operators

- Combines multiple boolean expressions
- If person's age greater than or equal to 13 and less than 17, he can go to G and PG-13 rated movies, but not R rated movies
- ```
if (age >= 13 && age < 17)
 cout << "You can go to G and PG-13"
 << " rated movies, but not R" +
 << " rated movies.") << endl;
```
- boolean operators
  - ❑ and - && (true if all conditions are true)
  - ❑ or - || (true if one condition is true)
  - ❑ not - ! (true if conditions is false)

# Expression Combinations

## The && (and) operator

| operand1 | operand2 | operand1 && operand2 |
|----------|----------|----------------------|
| true     | true     | true                 |
| true     | false    | false                |
| false    | true     | false                |
| false    | false    | false                |

Let age = 17

Let age = 16

Let age = 12

```
if (age >= 13 && age < 17)
 cout << "You can go to G and PG-13"
 << " rated movies, but not R" +
 << " rated movies." << endl;
```

# Expression Combinations

## The || (or) operator

| operand1 | operand2 | operand1    operand2 |
|----------|----------|----------------------|
| true     | true     | true                 |
| true     | false    | true                 |
| false    | true     | true                 |
| false    | false    | false                |

## The ! (not) operator

| operand | !operand |
|---------|----------|
| true    | false    |
| false   | true     |

### Example

```
if (!(grade == sentinelValue))
 cout << "The next grade is "
 << grade << endl;
```

### Alternative:

```
if (grade != sentinelValue)
 cout << "The next grade is "
 << grade << endl;
```

# Playing Cards

- Exercise with playing cards
  - Numbers represent the rank and suit of cards

```
// Codes for suits
```

```
const int SPADES = 0;
```

```
const int HEARTS = 1;
```

```
const int DIAMONDS = 2;
```

```
const int CLUBS = 3;
```

```
// Codes for nonnumeric ranks
```

```
const int ACE = 1;
```

```
const int JACK = 11;
```

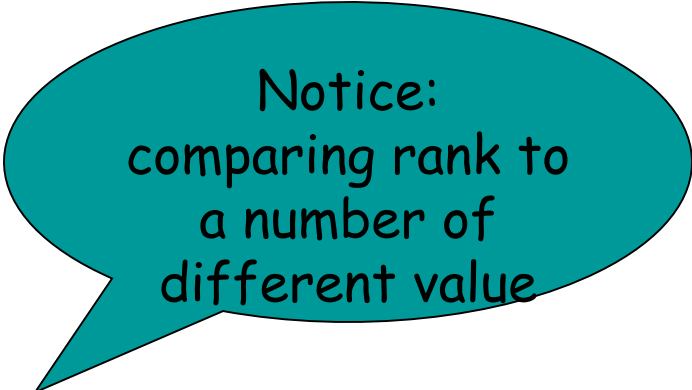
```
const int QUEEN = 12;
```

```
const int KING = 13;
```

# Prints a Card Name

- Print “rank of suit”
- Consider just the rank part

```
if (rank == JACK)
 cout << "Jack";
else if (rank == QUEEN)
 cout << "Queen";
else if (rank == KING)
 cout << "King";
else if (rank == ACE)
 cout << "Ace";
else
 cout << rank;
```



Notice:  
comparing rank to  
a number of  
different value

# 2.16 switch Multiple-Selection

## Structure

### ■ **switch**

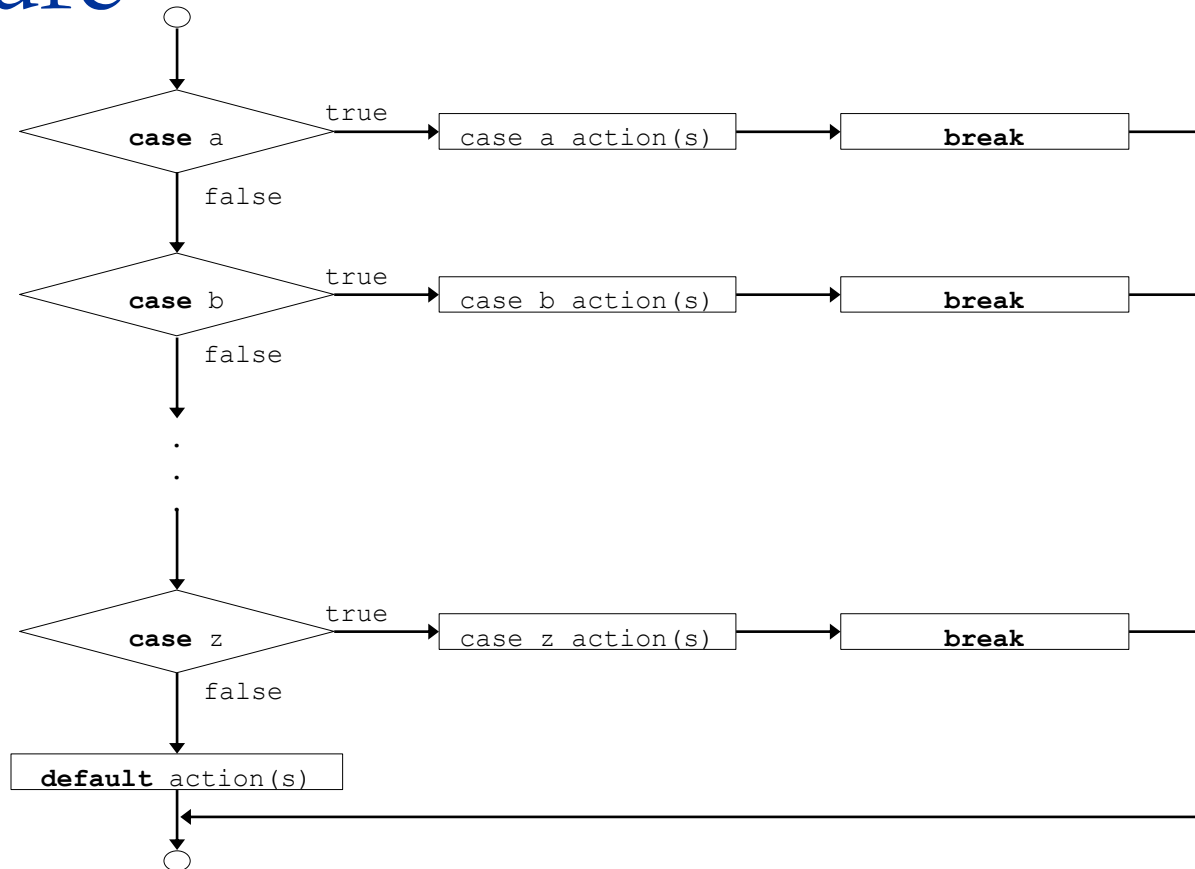
- ❑ Test variable for multiple values
- ❑ Series of **case** labels and optional **default** case

```
switch (variable) {
 case value1: // taken if variable == value1
 statements
 break; // necessary to exit switch

 case value2:
 case value3: // taken if variable == value2 or == value3
 statements
 break;

 default: // taken if variable matches no other cases
 statements
 break;
}
```

## 2.16 switch Multiple-Selection Structure



# Converting if/else to a switch

```
if (rank == JACK)
 cout << "Jack";

else if (rank == QUEEN)
 cout << "Queen";

else if (rank == KING;
 cout << "King";

else if (rank == ACE)
 cout << "Ace";

else
 cout << rank;
```

```
switch (rank)
{
 case JACK:
 cout << "Jack";
 break;
 case QUEEN:
 cout << "Queen";
 break;
 case KING:
 cout << "King";
 break;
 case ACE:
 cout << "Ace";
 break;
 default:
 cout << rank;
}
```