# Chapter 3

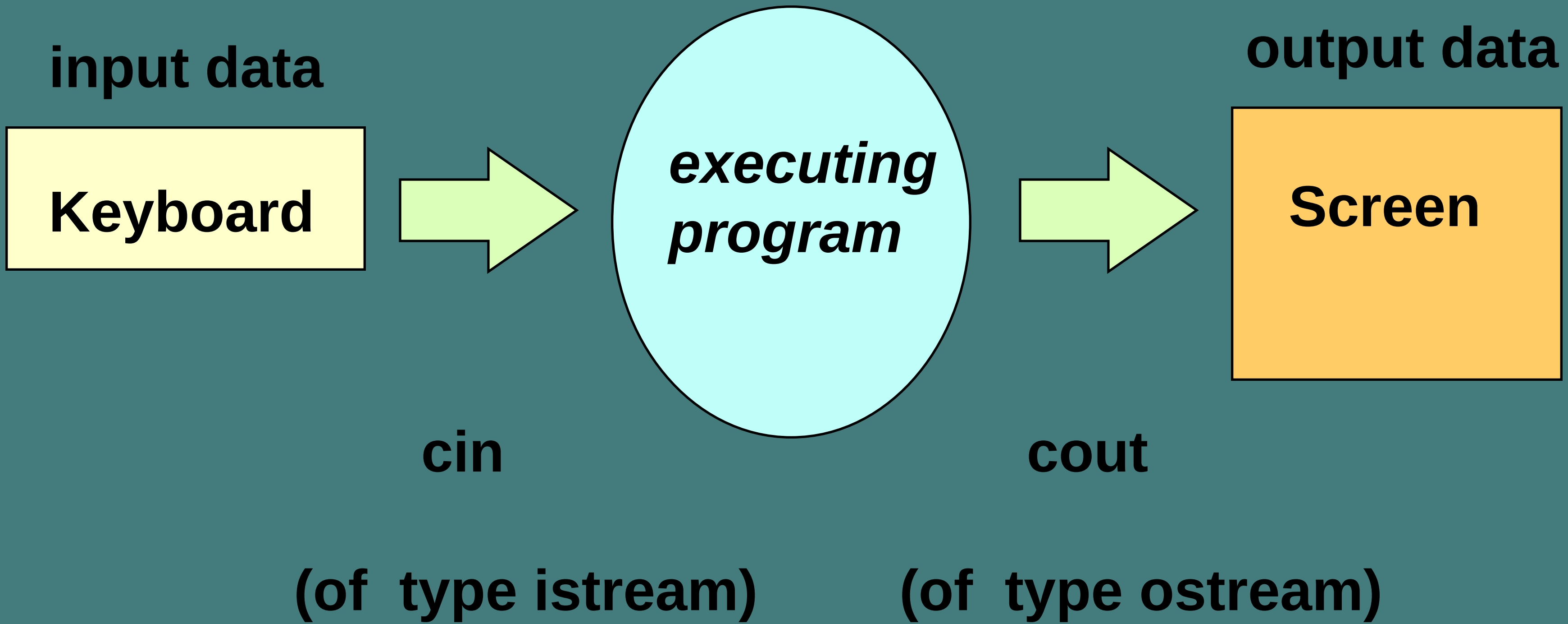
# Introduction to Programming

# Chapter 3 Topics

- ❖ Simple Input and Output Operations
  - ❖ Input Statements
  - ❖ Output Statements
  - ❖ Manipulators
- ❖ Control Structures
  - ❖ Basic Control Structures
  - ❖ Selection Control Structures
  - ❖ Loop Control Structures

## Keyboard and Screen I/O

#include <iostream>



## Input Statements

### SYNTAX

```
cin  >> Variable  >> Variable . . . ;
```

These examples yield the same result.

```
cin  >>  length ;
cin  >>  width ;
```

```
cin  >> length  >> width ;
```

## Output Statements

### SYNTAX

```
cout  <<  Expression   << Expression  . . . ;
```

### These examples yield the same output:

```
cout  <<  "The answer is " ;
cout  <<  3 * 4 ;
```

```
cout  <<  "The answer is "  <<  3 * 4 ;
```

## Manipulators

- manipulators are used only in input and output statements
- endl, fixed, setprecision, and setw are manipulators that can be used to control output format
- endl is used to terminate the current output line, and create blank lines in output

## Using Manipulator fixed

```
cout  <<   fixed;
```

- specify that (for output sent to the cout stream) **decimal format** (not scientific notation) be used, and that a **decimal point** be included (even for floating values with 0 as fractional part)

## setprecision(n)

- requires #include <iomanip>
- if fixed has already been specified, argument n determines the number of places displayed after the decimal point for floating point values
- remains in effect until explicitly changed by another call to setprecision

## What is exact output?

```
#include  <iomanip>                             // for setprecision( )
#include  <iostream>

using  namespace  std;

int main (  )
{
   float   myNumber  =  123.4587 ;

   cout  <<  fixed;          // use decimal format


   cout  <<  “Number is ”  <<  setprecision ( 3 )
           <<  myNumber     <<  endl ;

   return  0 ;
}
```

## OUTPUT

```
Number is 123.459
```

value is rounded if necessary to be displayed with exactly 3 places after the decimal point

## setw(n)

- requires #include <iomanip>
- argument n is called the field width specification, and determines the number of character positions in which to display a right-justified number or string (not char data). The number of positions used is expanded if n is too narrow
- “set width” affects only the very next item displayed

## What is exact output?

```
#include  <iomanip>                          // for setw( )
#include  <iostream>

using  namespace  std;

int  main (  )
{
   int  myNumber    =  123 ;
   int  yourNumber  =  5 ;

   cout  <<  setw ( 10 )    <<   “Mine”
          <<   setw ( 10 )    <<  “Yours”           <<   endl;
    cout <<   setw ( 10 )    <<  myNumber
          <<   setw ( 10 )    <<  yourNumber   <<   endl ;

   return 0 ;
}
```

## OUTPUT

```
position    12345678901234567890

                 Mine          Yours
                  123              5
```

## What is exact output?

```
#include  <iomanip>
#include  <iostream>

using  namespace  std;

int  main (  )
{
   float myNumber     = 3.14159;
   float yourNumber  = 123. 45678 ;

   cout  <<  fixed;


   cout  <<  “Numbers are: ”  <<  setprecision ( 3 ) <<  endl
            <<   setw ( 6 )            <<  myNumber      <<   endl
            <<   setw ( 6 )            <<  yourNumber   <<   endl ;

   return 0 ;
}
```

## OUTPUT

```
position    123456123456
```

```
Numbers are:
 3.142
123.457
```

| HEADER FILE | MANIPULATOR | ARGUMENT TYPE | EFFECT |
|---|---|---|---|
| <iostream> | endl | none | terminates output line |
| <iostream> | fixed | none | sets decimal point |
| <iomanip> | setw(n) | int | sets field width to n positions |
| <iomanip> | setprecision(n) | int | sets precision to n digits |

# Chapter 3 Topics

- Simple Input and Output Operations
  - Input Statements
  - Output Statements
  - Manipulators
- Control Structures
  - Basic Control Structures
  - Selection Control Structures
  - Loop Control Structures

## Basic Control Structures

- a **sequence** is a series of statements that execute one after another
- **selection (branch)** is used to execute different statements depending on certain conditions
- **looping (repetition)** is used to repeat statements while certain conditions are met

## Control structures

**use logical expressions which may include:**

***6 Relational Operators***

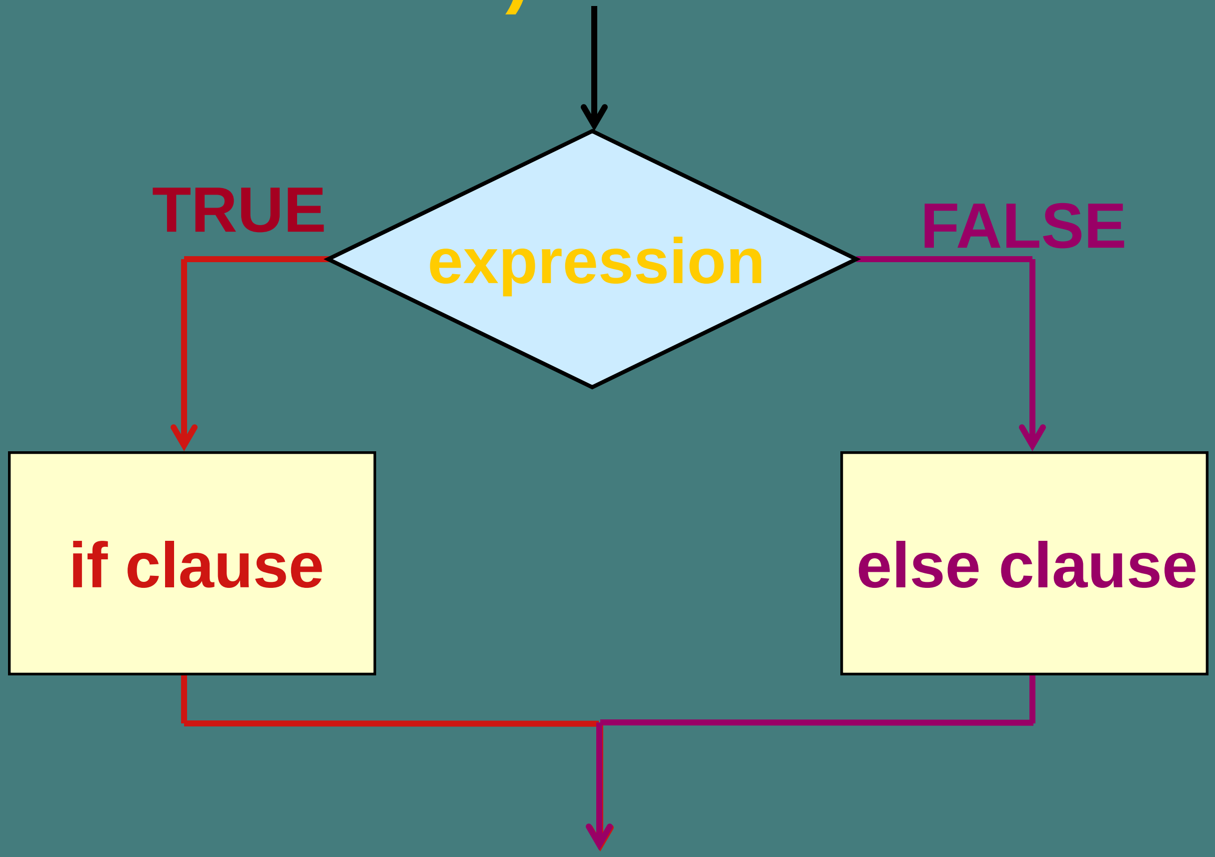
| | | | | | |
|---|---|---|---|---|---|
| < | <= | > | >= | == | != |

***3 Logical Operators***

| | | |
|---|---|---|
| ! | && | \|\| |

## *If-Then-Else* provides two-way selection

between executing one of 2 clauses (the if clause or the else clause)



## Example

```
int     carDoors,  driverAge ;
float  premium,  monthlyPayment ;
.   .   .
if  ( (carDoors == 4 )  &&  (driverAge > 24) )
{
            premium = 650.00 ;
            cout << " LOW RISK " ;
}
else
{
            premium = 1200.00 ;
            cout << " HIGH RISK " ;
}

monthlyPayment = premium / 12.0 + 5.00 ;
```

"if clause"

"else clause"

## *Nested if*

**Is a selection control structure for multi-way branching.**

**SYNTAX**

```
if  ( Expression1  )
                Statement1

else  if  ( Expression2  )
                Statement2
                    .
                    .
                    .
else if  ( ExpressionN  )
                StatementN
else
                Statement N+1
```

## Example

```
//Multi-way Branching
if  ( creditsEarned >= 90  )
          cout  <<  "SENIOR STATUS ";
else if  ( creditsEarned >= 60  )
          cout  <<  "JUNIOR STATUS ";
else if  ( creditsEarned >= 30  )
          cout  <<  "SOPHOMORE STATUS ";
else
          cout  <<  "FRESHMAN STATUS ";
```

## Two Types of Loops

**count controlled loops**

**repeat a specified number of times**

**event-controlled loops**

**some condition within the loop body changes and this causes the repeating to stop**

## Example

```
//Count-controlled Loop
int   count ;
count  =  4;                          // initialize loop variable
while (count > 0)                     // test expression
{
        cout  << count  << endl ;     // repeated action

        count  - -  ;                 // update loop variable
}
cout  << “Done” << endl ;
```

## Example

```
//Event-controlled Loop
 do{
        cin  >>  response ;

        if  ( ( response  !=  ‘y’ )  &&  ( response  !=  ‘n’ ) )
            cout  << “Please type  y or n : “ ;

 } while ( ( response  !=  ‘y’ )  &&  ( response  !=  ‘n’ ) ) ;
```