**How to draw a flowchart**

1. **definition**

A flowchart is used to describe the “flow of a program”.

1. **Symbols:**

An arrow is used to show the direction of the program flow

Using a circle with a letter or a number inside to represent a connector

This represents a program module, such as a function

Using a diamond to represent a yes/no question

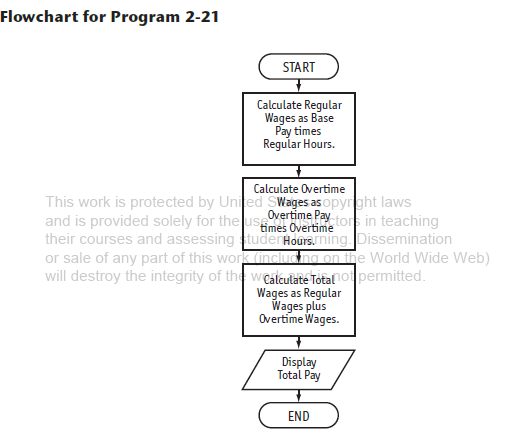
Using a rectangle to demonstrate a process, eg: calculations and a variable assignment

Using parallelograms to demonstrate input/ouput operations

Using rounded rectangles to represent terminal points

**3. Flowchart structures**

**1) Sequence**



// This program calculates hourly wages, including overtime.

#include <iostream>

using namespace std;

int main()

{

double regularWages, // To hold regular wages

basePayRate = 18.25, // Base pay rate

regularHours = 40.0, // Hours worked less overtime

overtimeWages, // To hold overtime wages

overtimePayRate = 27.78, // Overtime pay rate

overtimeHours = 10, // Overtime hours worked

totalWages; // To hold total wages

// Calculate the regular wages.

regularWages = basePayRate \* regularHours;

// Calculate the overtime wages.

overtimeWages = overtimePayRate \* overtimeHours;

// Calculate the total wages.

totalWages = regularWages + overtimeWages;

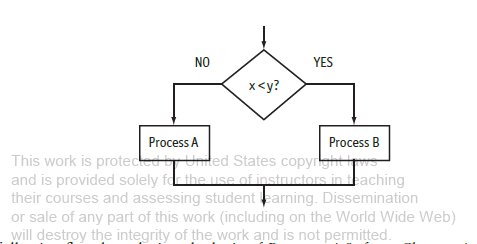
// Display the total wages.

cout << "Wages for this week are $" << totalWages << endl;

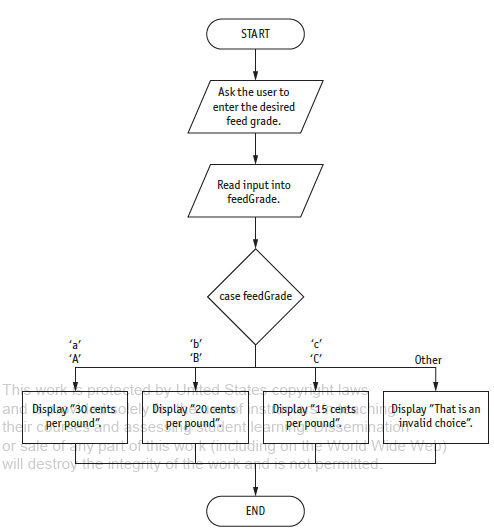
return 0;

}

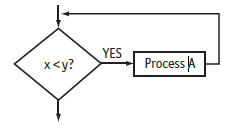
**2) Decision**



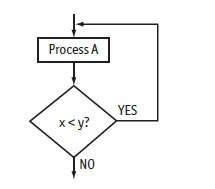
**3) Case**

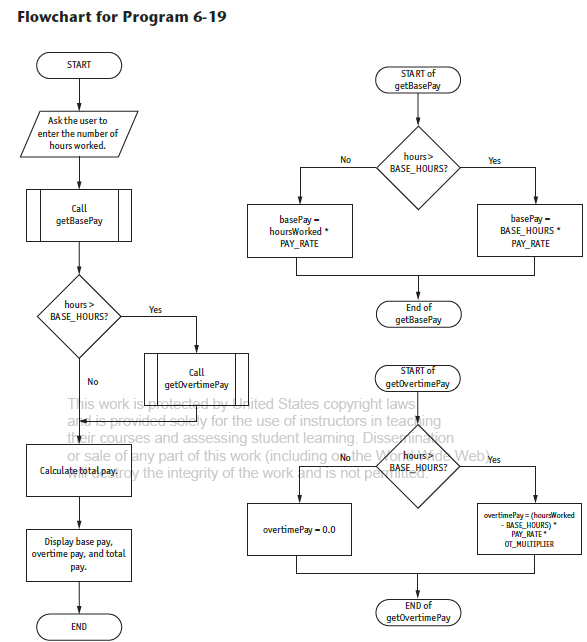


**4) Repetition (loop)**

**A. Pre-test:**

**B. Post-test:**





// This program calculates gross pay.

#include <iostream>

#include <iomanip>

using namespace std;

// Global constants

const double PAY\_RATE = 22.55; // Hourly pay rate

const double BASE\_HOURS = 40.0; // Max non-overtime hours

const double OT\_MULTIPLIER = 1.5; // Overtime multiplier

// Function prototypes

double getBasePay(double);

double getOvertimePay(double);

int main()

{

double hours, // Hours worked

basePay, // Base pay

overtime = 0.0, // Overtime pay

totalPay; // Total pay

// Get the number of hours worked.

cout << "How many hours did you work? ";

cin >> hours;

// Get the amount of base pay.

basePay = getBasePay(hours);

// Get overtime pay, if any.

if (hours > BASE\_HOURS)

overtime = getOvertimePay(hours);

// Calculate the total pay.

totalPay = basePay + overtime;

// Set up numeric output formatting.

cout << setprecision() << fixed << showpoint;

// Display the pay.

cout << "Base pay: $" << basePay << endl

<< "Overtime pay $" << overtime << endl

<< "Total pay $" << totalPay << endl;

return 0;

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// The getBasePay function accepts the number of \*

// hours worked as an argument and returns the \*

// employee's pay for non-overtime hours. \*

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

double getBasePay(double hoursWorked)

{

double basePay; // To hold base pay

// Determine base pay.

if (hoursWorked > BASE\_HOURS)

basePay = BASE\_HOURS \* PAY\_RATE;

else

basePay = hoursWorked \* PAY\_RATE;

return basePay;

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// The getOvertimePay function accepts the number \*

// of hours worked as an argument and returns the \*

// employee's overtime pay. \*

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

double getOvertimePay(double hoursWorked)

{

double overtimePay; // To hold overtime pay

// Determine overtime pay.

if (hoursWorked > BASE\_HOURS)

{

overtimePay = (hoursWorked - BASE\_HOURS) \*

PAY\_RATE \* OT\_MULTIPLIER;

}

else

overtimePay = 0.0;

return overtimePay;

}

**Citation**

2018 Pearson Education, Inc., Hoboken, NJ. All rights reserved.