**Express Invoice 2.0 Code Walk Through**

**This C++ application calculates invoice discounts based on customer type and subtotal. Let's go through the code step by step.**

**Include necessary libraries:**

#include <iostream>

#include <string>

#include <cmath>

These lines include the necessary libraries for the program. <iostream> is used for input and output streams, <string> provides string manipulation functions, and <cmath> includes mathematical functions.

**Declare the namespace:**

using namespace std;

This line lets you use the std namespace without typing std:: before every standard library function.

**Define the main function:**

int main()

The main function is the entry point of the program.

**Display the welcome message:**

cout << "Welcome to the Invoice Calculator Program version 2.0" << endl;

This line uses cout to print the welcome message to the screen, and endl to start a new line.

**Get the customer type from the user:**

char customerType;

cout << "Enter the customer type (r or w or c): ";

cin >> customerType;

Declare a char variable called customerType to store the customer type input. Prompt the user to enter the customer type, and use cin to read the input.

**Validate the customer type input:**

while (customerType != 'r' && customerType != 'w' && customerType != 'c')

{

cout << "invalid entry" << endl;

cout << "Enter the customer type (r or w or c): ";

cin >> customerType;

}

Use a while loop to keep asking for input until the user enters a valid customer type ('r', 'w', or 'c').

**A detailed look at what the while loop is and does:**

The while() loop is a control flow statement that executes a block of code repeatedly as long as the condition specified in the while() statement evaluates to true. Here are the steps to use the while() loop in C++:

* Declare any variables that you need to use inside the while() loop.
* Start the while() loop with the keyword "while", followed by an open parenthesis "(".
* Inside the parenthesis, write the condition that will be evaluated each time the loop is executed.
* This condition should evaluate as either true or false.
* Close the parenthesis ")" and open a set of curly braces "{ }" to define the block of code that will be executed as long as the condition evaluates to true.
* Write the code you want to execute inside the while() loop.
* At the end of the code block, you should modify the variables used in the condition, so that the condition will eventually become false and the loop will terminate.

**Here's an example of a while() loop in C++ that prints the numbers from 1 to 5:**

int i = 1;

while (i <= 5) {

cout << i << endl;

i++;

}

* In this example, the variable "i" is declared and initialized to 1.
  + The while() loop condition checks whether "i" is less than or equal to 5.
  + If it is, then the code inside the loop is executed, which prints the value of "i" and increments it by 1.
  + This process continues until "i" becomes more significant than 5, when the condition becomes false, and the loop terminates.

The while() loop in C# is similar to the while() loop in C++, with a few minor differences.

**Here are the steps to use the while() loop in C#:**

* Declare any variables that you need to use inside the while() loop.
* Start the while() loop with the keyword "while", followed by an open parenthesis "(".
* Inside the parenthesis, write the condition that will be evaluated each time the loop is executed.
  + This condition should evaluate as either true or false.
* Close the parenthesis ")" and open a set of curly braces "{ }" to define the block of code that will be executed as long as the condition evaluates to true.
* Write the code you want to execute inside the while() loop.
* At the end of the code block, you should modify the variables used in the condition, so that the condition will eventually become false and the loop will terminate.

**Here's an example of a while() loop in C# that prints the numbers from 1 to 5:**

int i = 1;

while (i <= 5) {

Console.WriteLine(i);

i++;

}

* In this example, the variable "i" is declared and initialized to 1.
  + The while() loop condition checks whether "i" is less than or equal to 5.
  + If it is, then the code inside the loop is executed, which prints the value of "i" and increments it by 1.
  + This process continues until "i" becomes more significant than 5, when the condition becomes false, and the loop terminates.

That's it! Both C++ and C# use the while() loop similarly to execute a block of code repeatedly as long as a specific condition is true.

**Get the subtotal from the user:**

cout << "Enter subtotal" << endl;

double subtotal;

cin >> subtotal;

Prompt the user to enter the subtotal, declare a double variable called subtotal to store the input, and use cin to read the input.

**Validate the subtotal input:**

while (subtotal < 0)

{

cout << "Invalid subtotal" << endl;

cout << "Enter valid subtotal" << endl;

cin >> subtotal;

}

Use a while loop to keep asking for input until the user enters a valid subtotal (a non-negative number).

**Calculate the discount percent:**

double discountPercent;

Declare a double variable called discountPercent to store the calculated discount percentage.

**Use a switch statement to calculate the discount percentage based on the customer type:**

switch (customerType)

{

// Retail

case 'r':

case 'R':

...

break;

// Wholesale

case 'w':

case 'W':

...

break;

// College

case 'c':

case 'C':

...

break;

// Other

default:

...

break;

}

The switch statement evaluates the customerType variable and runs the corresponding code block depending on the value. The discount percentage is calculated based on the customer type and subtotal.

**A detailed look at the switch statement:**

The switch statement is a control flow statement that allows you to test the value of an expression against a series of possible cases. Depending on the value of the expression, the switch statement can execute different blocks of code.

**Here's how to use the switch statement in C++ and C#:**

**The switch statement in C++ has the following syntax:**

switch(expression) {

case value1:

// code to be executed if expression == value1

break;

case value2:

// code to be executed if expression == value2

break;

case value3:

// code to be executed if expression == value3

break;

// and so on...

default:

// code to be executed if none of the cases match the expression

break;

}

**Here's a breakdown of how the switch statement works in C++:**

* An expression follows the switch keyword in parentheses.
  + This expression must be evaluated to an integral type (int, char, or enum).
  + The code inside the switch statement is executed in order until one of the case statements matches the value of the expression.
  + If a case statement matches the value of the expression, the code inside that case statement is executed.
    - If there is no break statement at the end of the case statement, execution will "fall through" to the following case statement.
  + If none of the case statements match the value of the expression, the code inside the default statement is executed.

**Here's an example of a switch statement in C++ that tests the value of a variable "grade":**

switch (grade) {

case 'A':

cout << "Excellent!" << endl;

break;

case 'B':

cout << "Good job!" << endl;

break;

case 'C':

cout << "You passed." << endl;

break;

case 'D':

case 'F':

cout << "Better luck next time." << endl;

break;

default:

cout << "Invalid grade." << endl;

break;

}

* In this example, the variable "grade" is tested against different cases using the switch statement.
  + If the value of "grade" is 'A,’ the code inside the first case statement is executed, which prints "Excellent!" to the console.
  + If the value of "grade" is 'B,’ the code inside the second case statement is executed, which prints "Good job!" to the console. And so on.

**The switch statement in C# has a similar syntax to the switch statement in C++. Here's how it looks in C#:**

switch(expression) {

case value1:

// code to be executed if expression == value1

break;

case value2:

// code to be executed if expression == value2

break;

case value3:

// code to be executed if expression == value3

break;

// and so on...

default:

// code to be executed if none of the cases match the expression

break;

}

* The switch statement works the same way in C# as in C++.
  + The expression is evaluated, and the code inside the switch statement is executed based on which case statement matches the value of the expression.
  + If none of the cases match, the code inside the default statement is executed.

**Here's an example of a switch statement in C# that tests the value of a variable "day":**

switch (day) {

case "Monday":

case "Tuesday":

case "Wednesday":

case "Thursday":

Console.WriteLine("It's a weekday.");

break;

case "Friday":

Console.WriteLine("Thank God it's Friday!");

break;

case "Saturday":

case "Sunday":

Console.WriteLine("It's the weekend!");

break;

default:

Console.WriteLine("Invalid day.");

break;

}

* In this example, the variable "day" is tested against different cases using the switch statement.
  + If the value of "day" is "Monday,” "Tuesday,” "Wednesday", or "Thursday", the code inside the first case statement is executed, which prints "It's a weekday." to the console.
  + If the value of "day" is "Friday", the code inside the second case statement is executed, which prints "Thank God it's Friday!" to the console.
  + And so on.

**When to use the switch statement:**

The switch statement applies when you have a value you want to compare against a series of possible cases. It's often used as an alternative to a long chain of if-else statements. The switch statement can make your code easier to read and understand, especially if you have many cases to test. However, suppose you have a large number of cases. In that case, using a different data structure, such as a dictionary or a hash table, may be more efficient in storing the possible values and their corresponding actions.

**Calculate the discount amount and total:**

double discountAmount = subtotal \* discountPercent;

double total = subtotal - discountAmount;

Declare two double variables, discountAmount and total, to store the calculated discount amount and capacity, respectively.

cout << "Discount percent is: " << discountPercent << endl;

cout << "Discount amount is: " << discountAmount << endl;

cout << "The total is: " << total << endl;

Use cout to print the discount percent, discount amount, and total to the screen.

**Display the goodbye message:**

cout << "Goodbye from the Invoice Calculator Program" << endl;

This line uses cout to print the goodbye message to the screen.

**Return from the main function:**

return 0;

This line indicates the end of the main function and successful program execution.

To learn from this application, you can study each part of the code, understand the logic, and try modifying it to add new features or handle additional edge cases. You can also experiment with different methods for input validation or discount calculation to deepen your understanding of C++ programming.