Summer Job Income Calculator

Introduction

Imagine you've landed an exciting summer job that pays well, but as with any income, it's not just about earning money. It's about managing it wisely. That's where our program comes in! The purpose of this program is quite straightforward: to help you calculate your income, understand how expenses impact your finances, and plan your savings from your summer job. By the end of this lesson, you'll have a tool to demystify the financial aspect of your summer earnings.

Problem set

You found and exciting summer job for five weeks. It pays 35.50 php per hour. Suppose that the total tax you pay on you summer job income is 12%. After paying taxes, you spend 10% of your net income to buy new clothes and other accessories and 1% to buy school supplies. After buying clothes and school supplies, you use 25% for savings.

Write a program that prompt the user to enter the total hours worked. Then the program will display the following outputs:

a. Your income before and after taxes are taken from your summer job
b. The amount of money you spend on clothes and other accessories
c. The money you spend on school supplies
d. The money you spend for your savings
e. Your remaining money

Objectives

1. Learn: Understand Income Components

    - Objective: By the end of this lesson, learners should grasp the components of their summer job income, including the hourly wage and total hours worked, and how these contribute to the total earnings.

    - Importance: This objective ensures learners can accurately evaluate their earnings, a fundamental aspect of financial awareness.

2. Apply: Calculate Gross and Net Income

    - Objective: Upon completing this lesson, students should be able to apply the provided formula to calculate both gross income (before taxes) and net income (after taxes) from their summer job.

- Importance: This skill empowers students to perform essential financial calculations, aiding them in understanding the real-world implications of taxes on their income.

3. Analyze: Breakdown of Expenses

   - Objective: By the end of this lesson, participants should be able to analyze and calculate their expenses for clothing and school supplies as percentages of their net income.
   - Importance: This objective helps students recognize the importance of allocating their earnings toward expenses, fostering responsible spending habits.

4. Plan: Determine Savings Allocation

   - Objective: After completing this lesson, learners should be capable of calculating their savings by allocating a specific percentage of their net income.
   - Importance: Teaching the concept of savings instills financial responsibility, ensuring that learners put aside money for future goals or unexpected expenses.

5. Apply Knowledge: Calculate Remaining Money

   - Objective: By the end of this lesson, students should be able to compute the remaining amount of money after accounting for expenses and savings.
   - Importance: This objective helps learners understand how much money they have available for discretionary spending or additional savings.

These objectives are structured to guide learners in comprehending the financial aspects of their summer job income and making informed decisions regarding their finances. Understanding income, expenses, and savings is vital for financial literacy and responsible money management.

Source code example

```
using System;

namespace SummerJobCalculator
{
    class Program
    {
        static void Main(string[] args)
        {
            Console.WriteLine("iNetTutor.com");
            Console.WriteLine("Summer Job Income Calculator");
            Console.WriteLine("----------------------------");

            // Constants
            const double HourlyRate = 35.50;
            const double TaxRate = 0.12;
            const double ClothesExpenseRate = 0.10;
            const double SchoolSuppliesExpenseRate = 0.01;
            const double SavingsRate = 0.25;

            // Ask the user for total hours worked
            Console.Write("Enter the total hours worked: ");
            if (!double.TryParse(Console.ReadLine(), out double totalHours) || totalHours < 0)
            {
                Console.WriteLine("Invalid input. Please enter a valid non-negative number of hours.");
                return;
            }

            // Calculate income before taxes
            double incomeBeforeTaxes = totalHours * HourlyRate;

            // Calculate income after taxes
            double incomeAfterTaxes = incomeBeforeTaxes * (1 - TaxRate);

            // Calculate expenses
            double clothesExpense = incomeAfterTaxes * ClothesExpenseRate;
            double schoolSuppliesExpense = incomeAfterTaxes * SchoolSuppliesExpenseRate;
            double savings = incomeAfterTaxes * SavingsRate;

            // Calculate remaining money
            double remainingMoney = incomeAfterTaxes - (clothesExpense + schoolSuppliesExpense + savings);

            // Display the results
            Console.WriteLine($"Income before taxes: {incomeBeforeTaxes:C}");
            Console.WriteLine($"Income after taxes: {incomeAfterTaxes:C}");
            Console.WriteLine($"Money spent on clothes and accessories: {clothesExpense:C}");
            Console.WriteLine($"Money spent on school supplies: {schoolSuppliesExpense:C}");
            Console.WriteLine($"Money saved: {savings:C}");
            Console.WriteLine($"Remaining money: {remainingMoney:C}");

            Console.WriteLine("Press any key to exit.");
            Console.ReadKey();
        }
    }
}
```

Explanation

A. Variables and Constants

- Explanation: Variables are placeholders used to store data that can vary or change, while constants are used for values that remain fixed throughout the program's execution. In this lesson, variables are used to store values like hourly wage, total hours worked, and percentages for expenses and savings. Constants, on the other hand, might be used for fixed tax rates or other unchanging values.

B. User Input

- Explanation: User input is essential for this program as it allows individuals to provide data such as the total hours worked. Console.ReadLine() is used to capture this input from the user. It's crucial to validate user input to ensure it meets the program's requirements, such as being a valid number.

C. Calculations

- Explanation: Mathematical calculations are a fundamental aspect of this program. Learners apply basic arithmetic operations, such as multiplication (calculating income) and subtraction (deducting expenses and savings), to compute various financial aspects of their summer job.

D. Percentage Calculations

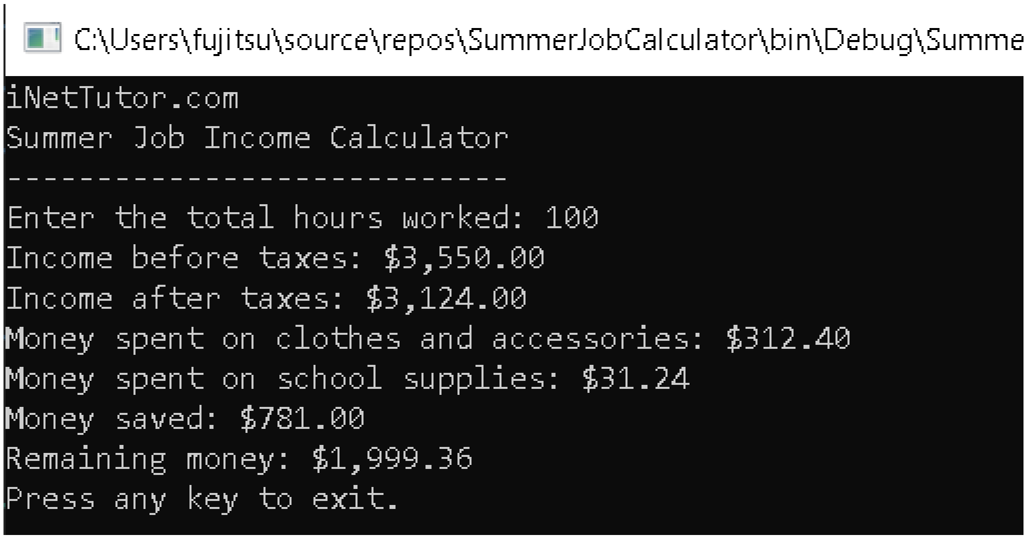
- Explanation: This concept involves determining specific percentages of an amount. In this lesson, learners calculate percentages for taxes, expenses (clothing and school supplies), and savings from their net income. Understanding how to calculate percentages is crucial for budgeting and financial planning.

E. Displaying Output

- Explanation: After performing calculations, it's important to present the results clearly to the user. Console.WriteLine() and string formatting are employed to display output in a readable format, providing users with information about their income, expenses, savings, and remaining funds.

These key concepts collectively form the foundation for learners to comprehend and apply financial principles through the Summer Job Income Calculator program. By mastering these concepts, users can better manage their finances and make informed decisions regarding their summer job income.

Output


C:\Users\fujitsu\source\repos\SummerJobCalculator\bin\Debug\Summe

```
iNetTutor.com
Summer Job Income Calculator
----------------------------
Enter the total hours worked: 100
Income before taxes: $3,550.00
Income after taxes: $3,124.00
Money spent on clothes and accessories: $312.40
Money spent on school supplies: $31.24
Money saved: $781.00
Remaining money: $1,999.36
Press any key to exit.
```

Summary

Throughout this lesson, you've gained valuable insights into creating a C# console program designed to calculate various financial aspects associated with a summer job. These encompass income, taxes, expenses, savings, and the remaining available funds. Your exploration started with an understanding of how to employ variables, essential for storing and manipulating data, alongside constants, which are utilized for unchanging values such as tax rates. You've delved into the mechanics of user input collection using Console.ReadLine(), an essential component in obtaining details like the total hours worked, with a particular emphasis on input validation to ensure data accuracy. Mathematical calculations, primarily based on multiplication to compute income and subtraction for deducting expenses and savings, were central to the program's functionality.

Furthermore, this lesson introduced the concept of calculating percentages, a crucial skill for determining tax amounts, allocating expenses for clothing and school supplies, and identifying savings from the net income. The lesson's practicality was significantly enhanced by instructing how to effectively display output through Console.WriteLine(), thereby offering users a comprehensive breakdown of their financial situation. These foundational concepts have armed you with the knowledge and skills necessary to engage in financial calculations using C#, a valuable skill applicable not only in this specific scenario but also in numerous other programming and financial contexts.

Exercises and Assessment

Exercise 1: Additional Expense Categories Challenge the learners to expand the program by allowing users to input additional expense categories and corresponding percentage allocations (e.g., entertainment, transportation). Then, modify the program to calculate and display these expenses separately, providing a more detailed breakdown of spending.

Exercise 2: Income Increase Encourage learners to enhance the program's realism by adding a feature that allows users to input pay rate increases over time. This would involve collecting information on pay rate changes for different weeks and adjusting the income calculations accordingly.

Assessment: Scenario Modification Present a modified scenario where the taxation rate changes (e.g., 10% for the first two weeks and 15% for the remaining weeks) and ask the learners to adapt the program to accommodate this change while maintaining accuracy in tax calculations.

These exercises and assessment will challenge learners to extend and adapt the program, promoting a deeper understanding of financial calculations and programming flexibility.

Quiz

1. What is the purpose of the Summer Job Income Calculator program?
   a) To calculate income from a summer job
   b) To calculate taxes on summer job income
   c) To calculate expenses for school supplies
   d) All of the above

2. How is user input collected in the program?
   a) Using Console.Write()
   b) Using Console.Read()
   c) Using Console.ReadLine()
   d) Using Console.Input()

3. Which data type is typically used to store income and expense values in the program?
   a) string
   b) int
   c) double
   d) bool

4. What is the purpose of constants in the program?
   a) To store user input
   b) To store fixed values like tax rates
   c) To store changing values

d) To store calculations

5. What key concept allows you to calculate percentages for taxes, expenses, and savings?
   a) Variables and Constants
   b) User Input
   c) Calculations
   d) Percentage Calculations