Самостійна робота 5. Розробка консольних програм з використанням функцій

Самостійна робота 8 присвячена поглибленому вивченню матеріалу теми 6 «Функції». ». Основна увага при поглибленому вивчені цієї теми приділялась питанню розроблення консольної програми, яка реалізує згідно індивідуального варіанту обробку масивів різноманітних даних з застосуванням певних функцій.

Мета роботи: отримання знань та навичок щодо розробки функцій користувача та їх застосування для обробки різноманітних даних.

У результаті виконання самостійної роботи у студента формуються такі **компетентності:** здатність розробляти програми з застосування стандартних функцій та функцій користувача; знання механізми опису і виклику функцій та способів обміну інформацією з функцією.

Результатом виконання самостійної роботи є налаштований консольний додаток та демонстрація його працездатності.

Завдання для самостійної роботи

Виконувати дану самостійну роботу доцільно перед виконанням лабораторної роботи «Використання функцій».

У ході виконання роботи необхідно:

1. Проаналізувати зміст індивідуального завдання і виділити ті пункти завдання, які можуть бути реалізовані у вигляді окремих процедур - функцій.

2. Підготувати чисельні контрольні приклади початкового масиву та результати його обробки згідно з індивідуальним варіантом.

3. Для кожної функції надати словесний опис дії алгоритму, який лежить в основі її роботи, та на цій основі розробити загальну графічну схему (блок-схему) опису алгоритму для усього додатку.

4. Обґрунтувати вибір операторів мови C#, які найкращим чином співвідносяться зі графічними схемами для кожної з функцій, та написати відповідний код опису та виклику функцій.

5. В середовище Visual Studio.NET створити шаблон консольного додатку та набрати у вбудований в нього редактор тексту розроблену програму.

5. Відкомпілювати текст програми, усуваючи у разі необхідності помилки.

6. Дослідити роботу програми, аналізуючи виконання контрольних чисельних прикладів.

Контрольні запитання для самодіагностики

1. Дайте визначення функції. Чому функція може слугувати прикладом коду, який повторно виконується?

2. Перерахуйте основні етапи виконання функції.

3. Який синтаксис запису значень, що повертаються з функції?

4. Охарактеризуйте параметри функцій. У чому полягає відповідність параметрів?

5. У чому полягають основні ідеї реалізації процесу обміну інформацією з функцією.

6. Що таке область дії змінних? Охарактеризуйте параметри і значення, що повертаються, за порівнянням із глобальними даними.

7. У чому полягають особливості передачі параметрів за посиланням і за значенням.

10. Які особливості використання функції і масиву?

Методичні рекомендації до теми

У якості прикладу розглянемо завдання де потрібно знайти в вихідному масиві цілих чисел суму елементів, значення яких менш нуля. Самостійна робота повинна виконуватися в наступній послідовності.

1. Аналіз змісту індивідуального завдання і виділення пунктів завдання, які можуть бути реалізовані у вигляді окремих процедур - функцій.

Як правило, обробці масиву випереджає його контрольний друк, також ця процедура вельми доцільна, якщо масив повинен обробляться і значення його елементів змінюються, або змінюється їх порядок (наприклад, після сортування). Друга процедура, яка витікає з завдання – це формування суми елементів, значення яких менш нуля. Надаємо цім процедурам відповідні ім’я: pech_mas( ) та sum_negativ( ).

2. Підготовка відповідного завданню чисельного початкового прикладу масиву та результатів його обробки.

У якості початкового масиву потрібно надати масив, який включає в себе три або більш нульових елементів.

3. Словесний опис і побудова відповідних графічних схем виконується аналогічно пункту 2 методичних вказівок к самостійний роботи 3. Крім двох блок-схем опису функцій, також необхідно навести загальну блок-схему, де кожний функції відповідає її стисле графічне зображення у вигляді прямокутника.

4. Розробка коду відповідних функцій.

На цьому етапі необхідно записати сигнатуру функції і код її тіла. Сигнатура містить ім'я функції, після якого в круглих дужках вказуються параметри функції, а також поперед ім'ям - значення, що повертається. Наприклад, визначення функцій **pech_mas** та **sum_negativ** , може мати такий вид:

```
static void pech_mas(int[ ] m)
        {
            for (int i = 0; i < m.Length;  i++)
                Console.Write("{0} ",m[i]);
            Console.WriteLine(); Console.WriteLine();
        }

static void sum_negativ[ ] m)
        {
            double sum_otr = 0.0;
            for(int i=0; i<m.Length; i++)
                if(m[i]<0)
                    sum_ negativ += m[i];
            Console.WriteLine( "sum_ negativ = {0}", sum_ negativ);
        }
```

Ключове слово **static** - означає що функція статична і до цієї функції можна звернутися з будь-якого місця без створення екземпляра об'єкту класу).

5. В середовище Visual Studio.NET створити шаблон консольного додатку та набрати у вбудований в нього редактор тексту розроблену програму. Наприклад, таким чином:

```
using System;
    class Class1
    {
        // місце для визначення функцій pech_mas  та sum_negativ

        static void Main(string[ ] args)
```

```
        {
            int[ ] mas = {-1,5,2,3,5,0,3,9,2,0,1,6,0,-3,2};
            pech_mas(mas);
            sum_ negativ (mas);
            Console.WriteLine();
        }
}
```

Слід звернути увагу на оформлення виклику функцій. У розглянутому прикладі обидві функції мають значення, що повертаються типу **void** і, отже, немає необхідності в явному вигляді (за допомогою операції привласнення) запам'ятовувати результати їх виконання.

6. Відкомпілювати текст програми, усуваючи у разі необхідності помилки, і дослідити її роботу, аналізуючи виконання контрольних чисельних прикладів.

Література: [1; 4; 5].