# **Лабораторная работа № 9**

**Тема:** Разработка и тестирование программ на языке С# с использованием функций пользователя.

**Цель работы:**

1. формирование навыков разработки и тестирования программ на языке С# с функциями пользователя;
2. формирование навыков работы в среде Visual Studio.Net;

. **Оборудование:**

1. Ноутбук core i5
2. Программное обеспечение: ОС Windows, среда Visual Studio.Net

**Контрольные вопросы:**

1. Метод в языке C# - это блок кода, который объединяет схожие операции или действия и может быть вызван в других частях программы для выполнения определенной функциональности. Он выполняет определенную задачу или возвращает значение, и может иметь параметры и тип возвращаемого значения.

2. Методы в языке C# могут быть разделены на несколько категорий:

- Статические (static) методы: они связываются с типом, а не с экземпляром объекта, и могут быть вызваны напрямую из класса без создания объекта этого класса.

- Экземплярные (instance) методы: они связываются с конкретным экземпляром объекта и могут быть вызваны только после создания экземпляра этого класса.

- Виртуальные (virtual) методы: они позволяют классам-наследникам переопределить методы базового класса с помощью ключевого слова "override".

- Абстрактные (abstract) методы: они не имеют определения в базовом классе и должны быть реализованы в классах-наследниках.

3. Метод-процедура - это метод, который не возвращает никакое значение, а только выполняет определенные действия или операции. Он может использоваться для модификации состояния программы или выполнения задач без вывода результата. Метод-функция, напротив, возвращает значение после выполнения определенных операций.

4. Метод-функция представляет собой метод, который не только выполняет действия, но также возвращает значение после завершения этих действий. Значение, возвращаемое методом, может быть использовано в других частях программы для дальнейших вычислений или операций.

5. Метод определяется с использованием ключевых слов "public", "private" (или других модификаторов доступности), за которыми следует тип возвращаемого значения (или ключевое слово "void" для методов-процедур), за которым следует имя метода, а затем фигурные скобки, в которых содержится тело метода (блок кода с операциями и инструкциями). Метод может иметь параметры, которые указываются в скобках после имени метода и перед закрывающей скобкой.

6. Метод можно вызвать путем использования его имени и передачи необходимых аргументов в круглых скобках, например:

МетодНазвание(аргумент1, аргумент2);

7. В C# существуют следующие типы параметров:

- Параметры значений (value parameters): передаются по значению и не изменяют оригинальное значение. Обозначаются ключевым словом "ref" или "out", либо не указываются, поскольку по умолчанию параметры являются параметрами значений.

- Параметры ссылок (reference parameters): передаются по ссылке и могут изменять значение оригинальной переменной. Обозначаются ключевым словом "ref" или "out".

- Параметры вывода (output parameters): используются для возвращения более одного значения из метода. Обозначаются ключевым словом "out".

8. Необязательные параметры (optional parameters) - это параметры, которые имеют значения по умолчанию и могут быть опущены при вызове метода. Они указываются в сигнатуре метода в круглых скобках с указанием значения по умолчанию, например:

public void МетодНазвание(int параметр1 = 0) { }

9. Именованные параметры (named parameters) - это параметры, которым можно явно присвоить значения при вызове метода по их именам, вместо порядка следования. Это удобно, когда у метода много параметров или не все параметры используются. Например:

МетодНазвание(параметр2: значение2, параметр1: значение1);

10. Параметры могут передаваться в метод путем передачи значений напрямую или по ссылке с использованием ключевых слов "ref" или "out".

11. При передаче параметров по значению в C# копия значения параметра передается в метод. Изменения, внесенные внутри метода, не влияют на исходное значение переменной, из которой был вызван метод. При передаче данных по ссылке в C# сама ссылка на объект передается в метод, а не его копия. Это означает, что изменения, внесенные внутри метода, будут отражены на исходной переменной.

12. Для передачи данных по ссылке в C# используется модификатор ref. Он указывается перед параметром метода и гарантирует, что все изменения, внесенные внутри метода, будут видны в вызывающем коде.

13. Модификатор out в C# используется для передачи параметра как выходного значения. Он аналогичен модификатору ref, но в отличие от него, out не требует, чтобы переменная была инициализирована перед передачей в метод. Метод, принимающий параметр out, должен присвоить значение этому параметру до того, как он завершит свою работу.

14. Модификатор params в C# позволяет методу принять переменное количество аргументов заданного типа. Он указывается перед последним параметром метода, который должен быть массивом. Объявление параметра с модификатором params позволяет вызывать метод с переменным количеством аргументов, без необходимости передавать массив.

15. Для передачи массива в метод в C# имя массива указывается в качестве аргумента при вызове метода. В методе параметр должен быть указан соответствующего типа массива. Метод сможет работать с переданным массивом и вносить изменения в его элементы, если это необходимо.

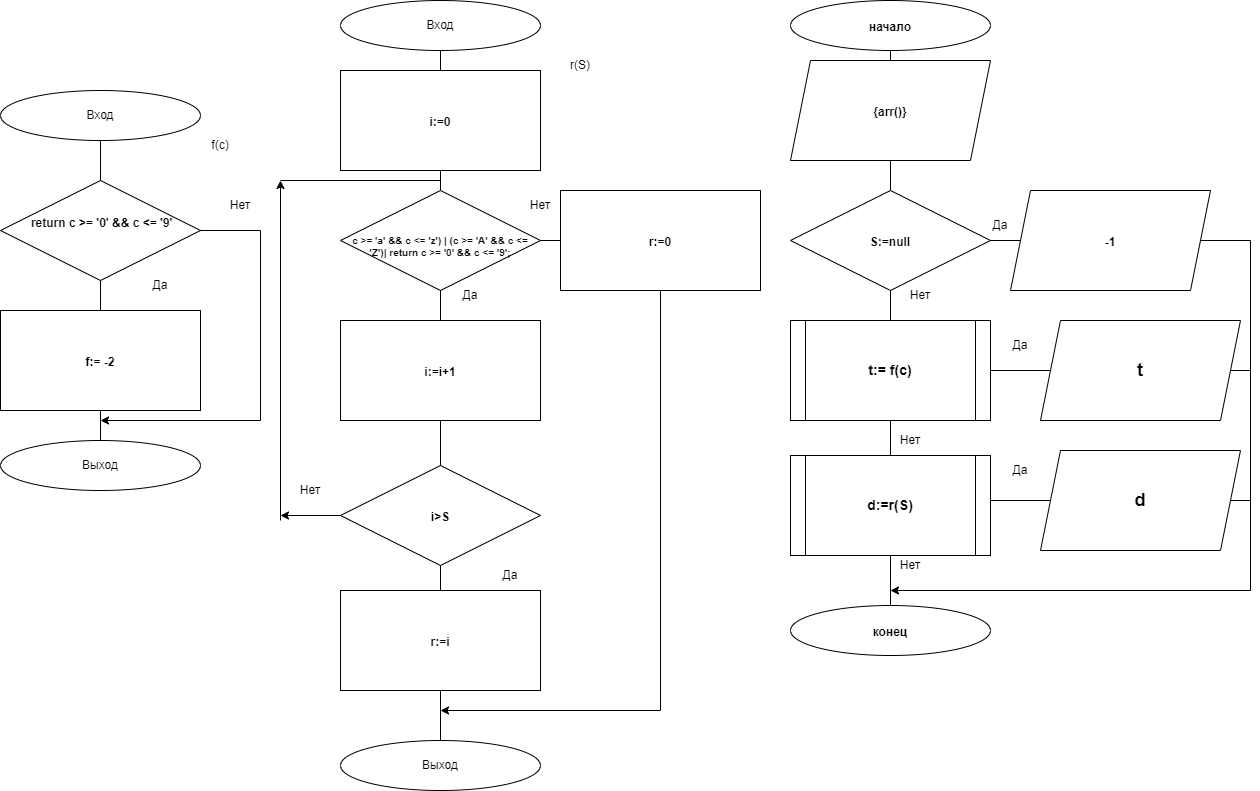
**Ход работы:**

1. Разработать алгоритм задачи и представить его в виде схемы программы
2. Используя Visual Studio.Net создать файл с программой
3. Выполнить тестирование и отладку программу
4. Результаты представить в виде отчета
5. Сделать вывод о проделанной работе

**Содержание отчета:**

**Задание на лабораторную работу:** Описать функцию IsIdent(S) целого типа, проверяющую, является ли строка S допустимым идентификатором в С#. При утвердительном ответе возвращается 0. Если S является пустой строкой, то возвращается -1, если строка начинается с цифры, то возвращается -2. Если S содержит недопустимые символы, то возвращается номер первого недопустимого символа. Проверить с помощью этой функции пять данных строк.

**Схема программы**



**Листинг с исходным кодом**

class Program

{

static void Main(string[] args)

{

string[] arr = { "shegjbesjz12", "", "5534hbsks", "uwgaew\_ka", "dzbk@segeuf" };

foreach (string str in arr)

{

int res = IsIdent(str);

Console.WriteLine($"Строка \"{str}\" результат: {res}");

}

Console.ReadKey();

}

static int IsIdent(string S)

{

if (S == "")

{

return -1;

}

else if (IsDigit(S[0]))

{

return -2;

}

else

{

for (int i = 0; i < S.Length; i++)

{

if (!IsLetterOrDigit(S[i]) && S[i] != '\_')// Проверяет, является ли текущий символ S[i]

// не буквой или цифрой (!IsLetterOrDigit(S[i]))

// и при этом символ не равен знаку подчеркивания (S[i] != '\_').

{

return i;

}

}

}

return 0;

}

static bool IsDigit(char c)//проверяет, является ли символ c цифрой.

//Он возвращает true, если символ находится в диапазоне от '0' до '9', и false в противном случае.

{

return c >= '0' && c <= '9';

}

static bool IsLetterOrDigit(char c)//проверяет, является ли символ c буквой или цифрой.

//Он возвращает true, если символ находится в одном из следующих диапазонов: от 'a' до 'z', от 'A' до 'Z'

//или если символ является цифрой, используя вызов метода IsDigit(c).

//В противном случае, метод возвращает false.

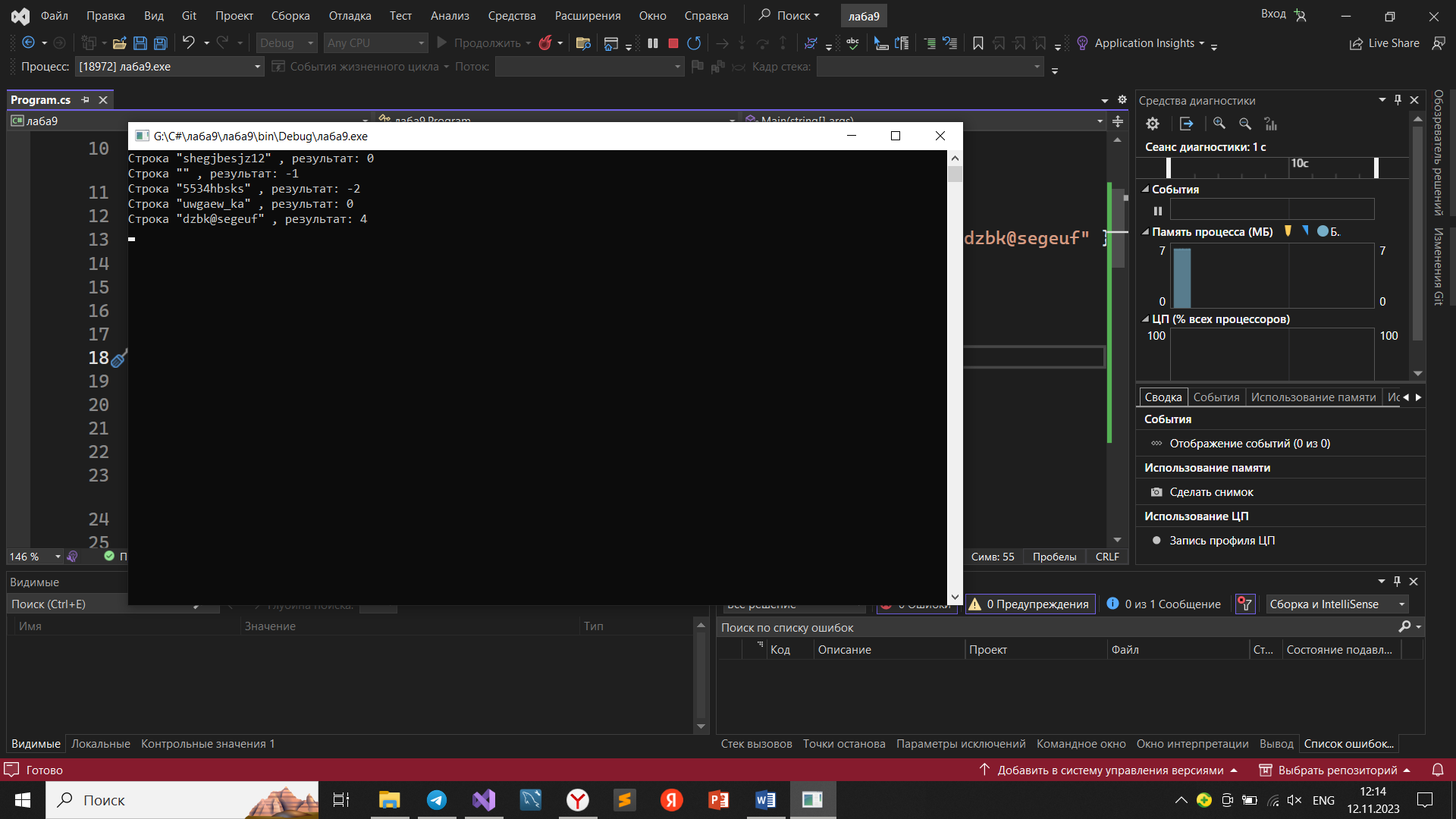
{

return (c >= 'a' && c <= 'z') || (c >= 'A' && c <= 'Z') || IsDigit(c);

}

}

**Результаты тестирования**



**Вывод по работе.** Работа с функциями упрощает код и делает его понятнее и короче.