Министерство образования Республики Беларусь
Учреждения образования
БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

КАФЕДРА ИНФОРМАТИКИ

Отчет по лабораторной работе №5

По теме «Интерпретация исходного кода»

Выполнил:
студент гр. 053502
Песоцкий В.А.

Проверил:
Гриценко Н. Ю.

Минск 2023

СОДЕРЖАНИЕ

1 ЦЕЛЬ РАБОТЫ

На основе результатов анализа лабораторных работ 1-4 выполнить интерпретацию программы.

2 РЕЗУЛЬТАТЫ ВЫПОЛНЕНИЯ

Демонстрируется интерпретация кода программ, на различный языковой синтаксис и обработку.

На рисунке 2.1 представлен код первой программы:

```
csharp_code = '''
using System;

public class Program {
    public static void Main() {
        int[] arr = {5, 2, 4, 6, 1, 3};
        Console.WriteLine("Before sorting:");
        Console.WriteLine(string.Join(", ", arr));

        Sort(arr);

        Console.WriteLine("After sorting:");
        Console.WriteLine(string.Join(", ", arr));
    }

    public static void Sort(int[] arr) {
        int n = arr.Length;
        for (int i = 0; i < n - 1; i++) {
            for (int j = 0; j < n - i - 1; j++) {
                if (arr[j] > arr[j + 1]) {
                    int temp = arr[j];
                    arr[j] = arr[j + 1];
                    arr[j + 1] = temp;
                }
            }
        }
    }
}
'''
```

Рисунок 2.1 – Код сортировки пузырьком

Результат интерпретации программы представлен ниже на рисунке 2.2. Программа ожидает когда пользователь введет длину массива, а затем соответсвующее количество значений, после чего сортирует его пузырьком и выводит результат на экран.

```
Before sorting:
3, 1, 6, 5, 4
After sorting:
1, 3, 4, 5, 6
```

Рисунок 2.2 – Результат выполнения

На рисунке 2.3 представлен код программы, проверяющий работу с байт-элементами:

```
byte_code = '''

using System;
using System.Text;

public class Program {
    public static void Main() {
        byte[] bytes = {72, 101, 108, 108, 111, 32, 87, 111, 114, 108, 100};
        string result = ConvertBytesToString(bytes);
        Console.WriteLine(result);
    }

    public static string ConvertBytesToString(byte[] bytes) {
        return Encoding.ASCII.GetString(bytes);
    }
}
```

Рисунок 2.3 – Байт-конвертер

Результат интерпретации второй программы представлен ниже на рисунке 2.4. Программа принимает на вход массив из байт, после чего согласно кодировке ASCII, преобразует массив байт в набор из букв латинского алфавита.

```
Введите строку:
ByteCode Kryto
Массив байт:
66, 121, 116, 101, 67, 111, 100, 101, 32, 75, 114, 121, 116, 111
```

Рисунок 2.4 – Результат выполнения

Далее рассмотрим код программы, проверяющий работу конкатенации строк и работу условных операторов. Данная программа принимает массив элементов, элемент который требуется найти и используя алгоритм бинарного поиска проводит поиск элемента, если элемент найден, то выводит его и его индекс в массиве, если же не найден, то выводит результат отсутствия элемента, чем проверяется условные выражения. Внутри методов библиотеки System, выводящих строки, используется присоединение двух строк, чем

проверяется работа методов внутри строк, различение операторов от самой строки. Код представлен на рисунке 2.5:

```
bnr_srch = '''
using System;

public class Program {
    public static void Main() {
        int[] arr = {1, 2, 3, 4, 5, 6};
        int searchValue = 4;
        Console.WriteLine("Array: {0}", string.Join(", ", arr));
        Console.WriteLine("Searching for: {0}", searchValue);

        int result = BinarySearch(arr, searchValue);

        if (result == -1) {
            Console.WriteLine("Value not found");
        } else {
            Console.WriteLine("Value found at index: {0}", result);
        }
    }

    public static int BinarySearch(int[] arr, int value) {
        int left = 0;
        int right = arr.Length - 1;
        while (left <= right) {
            int mid = (left + right) / 2;
            if (arr[mid] == value) {
                return mid;
            } else if (arr[mid] < value) {
                left = mid + 1;
            } else {
                right = mid - 1;
            }
        }
        return -1;
    }
}
'''
```

Рисунок 2.5 – Бинарный поиск

Результат интерпретации заключительной программы представлен ниже на рисунке 2.6.

```
Array: 1, 2, 3, 4, 5, 6
Searching for: 4
Value found at index: 3
```

Рисунок 2.6 – Бинарный поиск

ПРИЛОЖЕНИЕ А. ТЕКСТ ПРОГРАММ

```
import clr
import System
import os



import lexical


import synttest


pathDLLSort = os.getcwd() + "\\pythonCompile.dll"

pathDLLByte = os.getcwd() + "\\Bytes.dll"



clr.AddReference(pathDLLSort)

clr.AddReference(pathDLLByte)


# Bubble Sort Started


from pythonCompile import BubbleSort


import System


#print(BubbleSort)
```

```
print('\n')

print('\n')

print('\n')



arr = [5, 2, 9, 1, 5, 6]

cs_arr = System.Array.CreateInstance(System.Int32, len(arr))

for i, x in enumerate(arr):

    cs_arr[i] = x

BubbleSort.Sort(cs_arr)

for i in range(len(arr)):

    arr[i] = cs_arr[i]



#print(arr)



#print(clr.GetClrType(System.Object).Assembly.ImageRuntimeVersion)

# Bubble Sort Ended



#Byte Code Strted



import Bytes
```

```
#print(Bytes)


from Bytes import ByteCode


bytes_c = [104, 101, 108, 108, 111]
string = ByteCode.ConvertBytesToString(bytes_c)
#print(string)  # "hello"


bytes_b = [77, 121, 32, 78, 97, 109, 101, 32, 105, 115, 32, 86, 108, 97, 100]
string_2 = ByteCode.ConvertBytesToString(bytes_b)


#print(string_2) #My Name is Vlad ...



# пример кода на C#
csharp_code = '''
using System;


public class Program {
    public static void Main() {
        Console.WriteLine("Enter array length:");
        int n = Convert.ToInt32(Console.ReadLine());
        int[] arr = new int[n];
```

```
    Console.WriteLine("Enter array elements:");
    for (int i = 0; i < n; i++) {
        arr[i] = Convert.ToInt32(Console.ReadLine());
    }

    Console.WriteLine("Before sorting:");
    Console.WriteLine(string.Join(", ", arr));

    Sort(arr);

    Console.WriteLine("After sorting:");
    Console.WriteLine(string.Join(", ", arr));
}

public static void Sort(int[] arr) {
    int n = arr.Length;
    for (int i = 0; i < n - 1; i++) {
        for (int j = 0; j < n - i - 1; j++) {
            if (arr[j] > arr[j + 1]) {
                int temp = arr[j];
                arr[j] = arr[j + 1];
                arr[j + 1] = temp;
            }
        }
```

```
        }
    }
}
'''



byte_code = '''

using System;
using System.Text;

public class Program {
    public static void Main() {
        byte[] bytes = {72, 101, 108, 108, 111, 32, 87, 111, 114, 108, 100};
        string result = ConvertBytesToString(bytes);
        Console.WriteLine(result);
    }

    public static string ConvertBytesToString(byte[] bytes) {
        return Encoding.ASCII.GetString(bytes);
    }
}

'''
```

```
byte_code_user='''
using System;
using System.Text;

public class Program {
    public static void Main() {
        Console.WriteLine("Введите строку: ");
        string inputString = Console.ReadLine();
        byte[] bytes = Encoding.ASCII.GetBytes(inputString);
        Console.WriteLine("Массив байт:");
        Console.WriteLine(string.Join(", ", bytes));
    }

    public static string ConvertBytesToString(byte[] bytes) {
        return Encoding.ASCII.GetString(bytes);
    }
}
'''


bnr_srch = '''
using System;
```

```
public class Program {
    public static void Main() {
        int[] arr = {1, 2, 3, 4, 5, 6};
        int searchValue = 4;
        Console.WriteLine("Array: {0}", string.Join(", ", arr));
        Console.WriteLine("Searching for: {0}", searchValue);

        int result = BinarySearch(arr, searchValue);

        if (result == -1) {
            Console.WriteLine("Value not found");
        } else {
            Console.WriteLine("Value found at index: {0}", result);
        }
    }

    public static int BinarySearch(int[] arr, int value) {
        int left = 0;
        int right = arr.Length - 1;
        while (left <= right) {
            int mid = (left + right) / 2;
            if (arr[mid] == value) {
                return mid;
```

```
            } else if (arr[mid] < value) {
                left = mid + 1;
            } else {
                right = mid - 1;
            }
        }
        return -1;
    }
}
'''
```

```
def check_undeclared_variables(code):
    # регулярное выражение для поиска необъявленных переменных
    pattern = r'\b([a-zA-Z_]\w*)\b'
    variables = set(re.findall(pattern, code)) # получение списка всех переменных в коде
    declared_variables = {'int', 'float', 'double', 'char', 'string', 'bool'} # список зарезервированных переменных
    undeclared_variables = variables - declared_variables # нахождение необъявленных переменных

    if len(undeclared_variables) > 0:
        raise ValueError('Найдены необъявленные переменные: {}'.format(', '.join(undeclared_variables)))
```

```
def check_variable_types(code):
    # регулярное выражение для поиска типов переменных
    pattern = r'\b(int|float|double|char|string|bool)\b\s+([a-zA-Z_]\w*)\s*=?\s*([a-zA-Z_]\w*)?'
    variable_matches = re.findall(pattern, code)

    for match in variable_matches:
        variable_type, variable_name, variable_value = match

        # проверка типа переменной
        if variable_value and variable_type not in variable_value:
            raise TypeError('Переменная {} должна иметь тип {}, а получен тип {}'.format(variable_name, variable_type, type(variable_value).__name__))

def check_unused_variables(code):
    # регулярное выражение для поиска используемых переменных
    pattern = r'\b([a-zA-Z_]\w*)\b'
    used_variables = set(re.findall(pattern, code))

    # регулярное выражение для поиска объявленных переменных
    pattern = r'\b(int|float|double|char|string|bool)\b\s+([a-zA-Z_]\w*)\s*=?'
    declared_variables = set([match[1] for match in re.findall(pattern, code)])
```

```
    unused_variables = declared_variables - used_variables

    if len(unused_variables) > 0:
        raise ValueError('Обнаружены неиспользуемые переменные: {}'.format(', '.join(unused_variables)))
```

```
import operator

# Доступные операторы и соответствующие им функции
OPS = {
    "+": operator.add,
    "-": operator.sub,
    "*": operator.mul,
    "/": operator.truediv,
}

def evaluate_expression(expr):
    # Разбиваем строку на список токенов
    tokens = expr.split()

    # Стек для хранения операндов
```

```
    stack = []

    for token in tokens:
        if token.isdigit():
            # Если токен является числом, добавляем его на стек
            stack.append(int(token))
        elif token in OPS:
            # Если токен является оператором, извлекаем два операнда из стека,
            # применяем к ним оператор и добавляем результат на стек
            op2, op1 = stack.pop(), stack.pop()
            result = OPS[token](op1, op2)
            stack.append(result)
        else:
            # Если токен не является ни числом, ни оператором, возбуждаем исключение
            raise ValueError(f"Неизвестный токен: {token}")

    # После обработки всех токенов результат находится на вершине стека
    if len(stack) != 1:
        raise ValueError("Неправильный формат выражения")
    return stack.pop()
```

```
from anytree import NodeMixin, RenderTree


class AstNode(NodeMixin):
    def __init__(self, name, parent=None, children=None):
        super(AstNode, self).__init__()
        self.name = name
        self.parent = parent
        if children:
            self.children = children

    def __str__(self):
        return self.name

def ast_to_tree(ast):
    if isinstance(ast, tuple):
        node = AstNode(ast[0])
        for sub_ast in ast[1:]:
            child = ast_to_tree(sub_ast)
            child.parent = node
        return node
    elif isinstance(ast, list):
        node = AstNode("list")
        for sub_ast in ast:
```

```
            child = ast_to_tree(sub_ast)
            child.parent = node
        return node
    else:
        return AstNode(str(ast))



def check_unimplemented_methods(code):
    # регулярное выражение для поиска объявления интерфейсов
    pattern = r'\binterface\s+(\w+)\s*{([^}]*)}'
    interface_matches = re.findall(pattern, code)

    for interface_match in interface_matches:
        interface_name, interface_body = interface_match



# компиляция и выполнение кода на C#
def run_csharp_code(code: str) -> None:
    # добавление ссылки на mscorlib
    clr.AddReference('mscorlib')

    # компиляция исходного кода на C#
```

```
        provider = System.CodeDom.Compiler.CodeDomProvider.CreateProvider('CSharp')
        parameters = System.CodeDom.Compiler.CompilerParameters(['mscorlib.dll'])
        result = provider.CompileAssemblyFromSource(parameters, code)

        # проверка на наличие ошибок при компиляции
        if result.Errors.HasErrors:
            raise ValueError('Ошибка компиляции кода на C#!')

        # получение экземпляра метода Main()
        main_method = result.CompiledAssembly.GetType('Program').GetMethod('Main')

        # вызов метода Main() и получение результата его выполнения
        main_method.Invoke(None, None)

    # запуск кода на C#
    run_csharp_code(bnr_srch)
```