Задание 2

Деменчук Георгий ПИ19-4

Выполнить графовое представление и программную реализацию с помощью бинарного дерева следующие вычисления:

1. 2+2
2. (2+3)*4
3. (7+8)*(2-1)
4. (7+8)*(2-1)+7
5. (7+8)*(5-2)/(2-1)

Основная программная реализация

In [9]:

```
import operator
from tree_module import BinaryTree
```

In [10]:

```
class Str2Tree:
    """ Класс для перевода строки в бинарное дерево"""

    def __init__(self, exp):
        self.exp = ["("]+list(exp.replace(" ", ""))+[")"]
        self.tree = BinaryTree()
        self.processing()

    def digital_checker(self, number):
        """Проверка значения на число"""
        try:
            int(number)
            return True
        except ValueError:
            return False

    def processing(self):

        tree = self.tree
        current = tree

        for token in self.exp:
            if token == "(":
                # добавляем новый узел в качестве левого узла
                current = current.insert_left()

            elif token in ["+", "-", "/", "*"]:
                # устанавливаем значение в текущем узле
                current.set_root_val(token)
                # добавляем узел в качестве правого узла и спускаемся в него.
                current = current.insert_right()

            elif self.digital_checker(token):
                # устанавливаем значение в текущем узле
                current.set_root_val(int(token))
                # переходим к родительскому узлу.
                current = current.get_parent()

            elif token == ")":
                # переходим к родителю текущего узла.
                current = current.get_parent()

            else:
                print(token)
                raise ValueError("Я не понимаю что это")

        self.tree = tree


class Tree2Result:
    """Вычисление выражений на основе бинарного дерева"""

    def __init__(self, binarytree_obj):
        if not isinstance(binarytree_obj, BinaryTree):
            raise ValueError("Объект не является объектом класса BinaryTree!")

        self.result = self.processing(binarytree_obj)

    def processing(self, token):
```

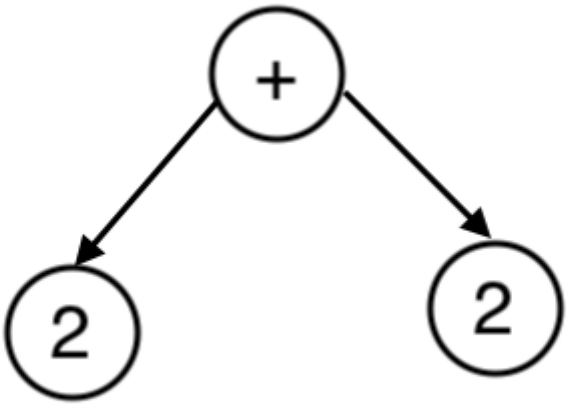
```
        """Обход бинарного дерева в обратном порядке"""
        operators = {'+': operator.add, '-': operator.sub,
                     '*': operator.mul, '/': operator.truediv}

        # Левые/правые значения
        left_value = token.get_left_child()
        right_value = token.get_right_child()

        # Если оба есть - падаем ниже
        if left_value and right_value:
            fn = operators[token.get_root_val()]
            # Отдаем результат выражения
            return fn(self.processing(left_value), self.processing(right_value))
        else:
            # Иначе, если уже некуда падать, отдаем текущее значение
            return token.get_root_val()
```

Выражение 2+2

Графовое представление



Программная реализация

In [11]:

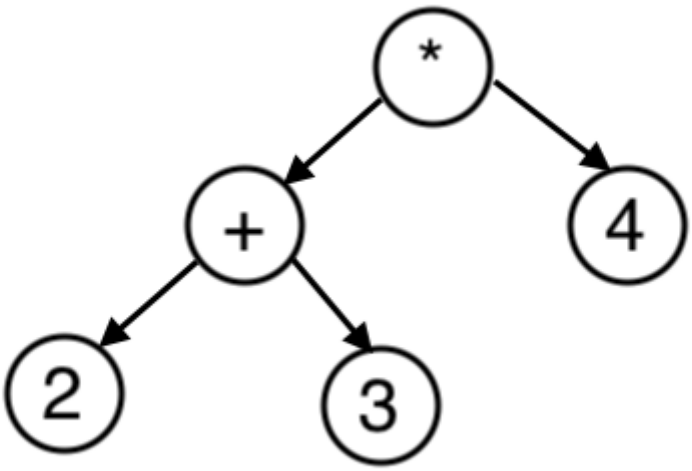
```
e = "2+2"
print("\nВыражение: {}".format(e))
obj = Str2Tree(e)
print("Разложение: {}".format(obj.tree))
r_obj = Tree2Result(obj.tree)
print("Результат вычисления: {}".format(r_obj.result))
print("Результат по eval: {}".format(eval(e)))
```

```
Выражение: 2+2
Разложение:
  +
 / \
2   2

Результат вычисления: 4
Результат по eval: 4
```

Выражение (2+3)*4

Графовое представление



Программная реализация

In [12]:

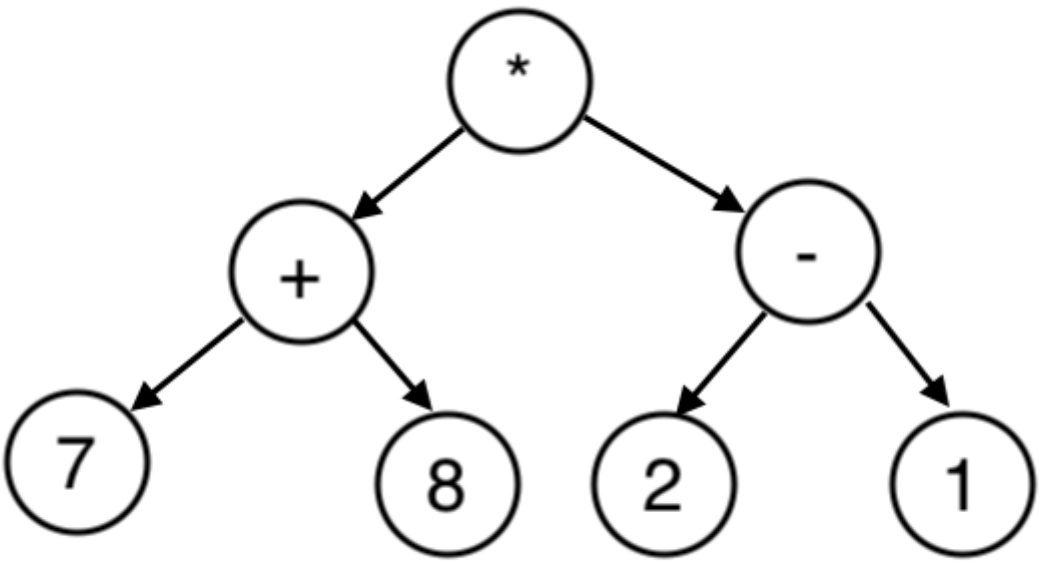
```
e = "(2+3)*4"
print("\nВыражение: {}".format(e))
obj = Str2Tree(e)
print("Разложение: {}".format(obj.tree))
r_obj = Tree2Result(obj.tree)
print("Результат вычисления: {}".format(r_obj.result))
print("Результат по eval: {}".format(eval(e)))
```

```
Выражение: (2+3)*4
Разложение:
      __*
    /    \
  +       4
 / \
2   3

Результат вычисления: 20
Результат по eval: 20
```

Выражение (7+8)*(2-1)

Графовое представление



Программная реализация

In [13]:

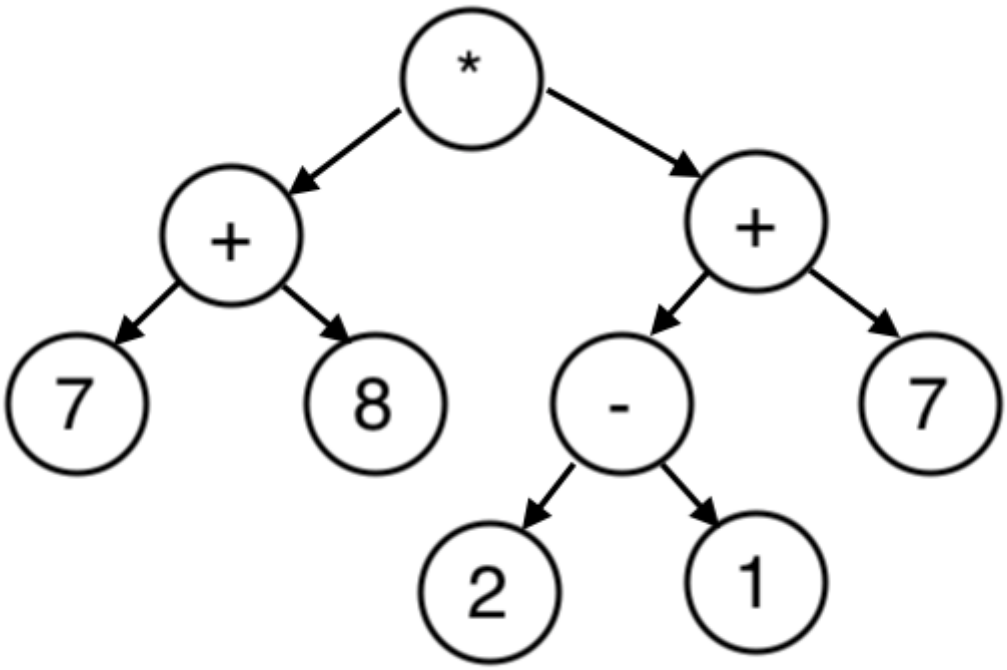
```
e = "(7+8)*(2-1)"
print("\nВыражение: {}".format(e))
obj = Str2Tree(e)
print("Разложение: {}".format(obj.tree))
r_obj = Tree2Result(obj.tree)
print("Результат вычисления: {}".format(r_obj.result))
print("Результат по eval: {}".format(eval(e)))
```

```
Выражение: (7+8)*(2-1)
Разложение:
     __*__
    /     \
   +       -
  / \     / \
 7   8   2   1

Результат вычисления: 15
Результат по eval: 15
```

Выражение (7+8)*(2-1)+7

Графовое представление



Программная реализация

In [14]:

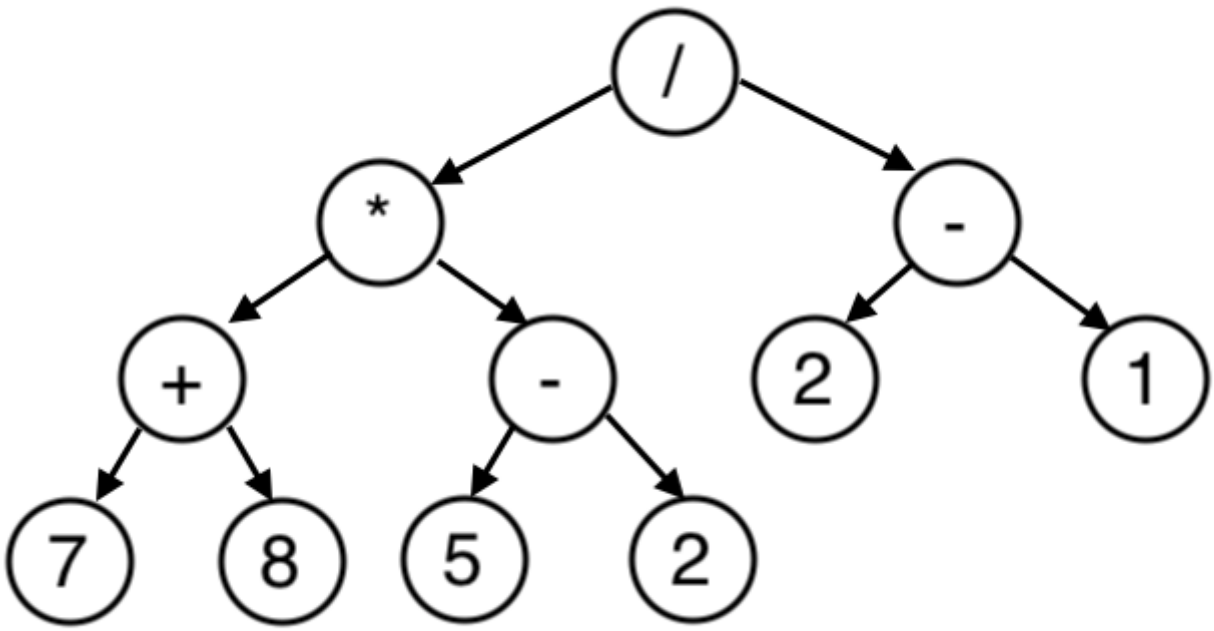
```
e = "(7+8)*(2-1)+7"
print("\nВыражение: {}".format(e))
obj = Str2Tree(e)
print("Разложение: {}".format(obj.tree))
r_obj = Tree2Result(obj.tree)
print("Результат вычисления: {}".format(r_obj.result))
print("Результат по eval: {}".format(eval(e)))
```

```
Выражение: (7+8)*(2-1)+7
Разложение:
      __+
    /    \
   +      7__
  / \        \
 7   8        -
             / \
            2   1

Результат вычисления: 22
Результат по eval: 22
```

Выражение (7+8)*(5-2)/(2-1)

Графовое представление



Программная реализация

In [15]:

```
e = "((7+8)*(5-2))/(2-1)"
print("\nВыражение: {}".format(e))
obj = Str2Tree(e)
print("Разложение: {}".format(obj.tree))
r_obj = Tree2Result(obj.tree)
print("Результат вычисления: {}".format(r_obj.result))
print("Результат по eval: {}".format(eval(e)))
```

```
Выражение: ((7+8)*(5-2))/(2-1)
Разложение:
          _______/__
         /          \
      __*__          -
     /     \        / \
    +       -      2   1
   / \     / \
  7   8   5   2

Результат вычисления: 45.0
Результат по eval: 45.0
```

In [ ]:

```

```

In [ ]:

```

```