Michał Pawlikowski, Informatyka Stosowana P2

Zadanie na 3.0

```
using System;

namespace HelloWorld
{
 abstract class Container
  {
     protected int pointer = -1;
     protected int[] buffer = new int[10];
     public abstract int pop();
     public abstract void push(int value);
     public abstract void show();

  }
  class Kolejka : Container
  {
          public override int pop()
    {
          int tmp = buffer[0];
       for(int i=1; i<=pointer; i++)
       {
          buffer[i-1] = buffer[i];
       }
       pointer--;
       if(pointer<-1)
       {
       pointer=-1;
       }
       return tmp;
    }
    public override void push(int value)
    {
          pointer++;
          if(pointer < 10)
       {
          buffer[pointer] = value;
       }
       else
       {
          pointer--;
       }
    }

    public override void show()
    {
          Console.Write("Kolejka: ");
       for(int i=0; i<=pointer; i++)
       {
          Console.Write(buffer[i]);
         Console.Write("  ");
       }
       Console.WriteLine("  ");
    }
  }
  class Stos : Container
  {
          public override void push(int value)
    {
          pointer++;
```

Run

```
        public override void push(int value)
        {
            pointer++;
            if(pointer < 10)
            {
                buffer[pointer] = value;
            }
            else
            {
                pointer--;
            }
        }

        public override int pop()
        {

            if(pointer < 0)
            {
                pointer=0;
            }
            int tmp1 = buffer[pointer];
            pointer--;
            return tmp1;
        }

        public override void show()
        {
            Console.Write("Stos: ");
            for(int i=0; i<=pointer; i++)
            {
                Console.Write(buffer[i]);
                Console.Write("  ");
            }
            Console.WriteLine("  ");
        }
    }
    class Program
    {
        static void Main(string[] args)
        {
           Console.WriteLine("Michał Pawlikowski");

           Kolejka k1 = new Kolejka();
            k1.push(1);
            k1.show();
            k1.push(2);
            k1.show();
            k1.push(3);
            k1.show();
            k1.pop();
            k1.show();

           Stos s1 = new Stos();
            s1.push(5);
            s1.push(7);
            s1.show();
            s1.push(9);
            s1.show();
            s1.pop();
            s1.show();
        }
    }
}
```

```
Michał Pawlikowski
Kolejka: 1
Kolejka: 1  2
Kolejka: 1  2  3
Kolejka: 2  3
Stos: 5  7
Stos: 5  7  9
Stos: 5  7
```

```
            if(pointer < 10)
        {
            buffer[pointer] = value;
        }
        else
        {
            pointer--;
        }
    }

    public override int pop()
    {

            if(pointer < 0)
        {
            pointer=0;
        }
        int tmp1 = buffer[pointer];
        pointer--;
        return tmp1;
    }

            public override void show()
    {
            Console.Write("Stos: ");
        for(int i=0; i<=pointer; i++)
        {
            Console.Write(buffer[i]);
           Console.Write("  ");
        }
        Console.WriteLine("  ");
    }
  }
  class Program
  {
    static void Main(string[] args)
    {
      Console.WriteLine("Michał Pawlikowski");

      Kolejka k1 = new Kolejka();
                        k1.push(1);
       k1.show();
       k1.push(2);
       k1.show();
       k1.push(3);
       k1.show();
       k1.pop();
       k1.show();

      Stos s1 = new Stos();
       s1.push(5);
       s1.push(7);
       s1.show();
       s1.push(9);
       s1.show();
       s1.pop();
       s1.show();
    }
  }
}
```

Zadanie na 4.0

```
using System;

namespace HelloWorld
{
  abstract class Container
    {
       protected int pointer = -1;
       protected int[] buffer = new int[10];
       public abstract int pop();
       public abstract void push(int value);
       public abstract void show();

       public void clear()
       {
              buffer = new int[10];
          pointer = -1;
       }

       public int getcount()
       {
              return pointer+1;
       }

       public Boolean isempty()
       {
          if (pointer == -1)
             return true;
          else
             return false;
       }

       public Boolean isfull()
       {
          if (pointer+1 == 10)
             return true;
          else
             return false;
       }

    }
    class Kolejka : Container
    {
              public override int pop()
       {
              int tmp = buffer[0];
          for(int i=1; i<=pointer; i++)
          {
              buffer[i-1] = buffer[i];
          }
          pointer--;
          if(pointer<-1)
          {
          pointer=-1;
          }
          return tmp;
       }
       public override void push(int value)
       {
              pointer++;
              if(pointer < 10)
          {
```

```
            buffer[pointer] = value;
        }
        else
        {
            pointer--;
        }
    }

    public override void show()
    {
            Console.Write("Kolejka: ");
        for(int i=0; i<=pointer; i++)
        {
            Console.Write(buffer[i]);
           Console.Write("  ");
        }
        Console.WriteLine("  ");
    }
}
class Stos : Container
{
            public override void push(int value)
    {
            pointer++;
            if(pointer < 10)
        {
            buffer[pointer] = value;
        }
        else
        {
            pointer--;
        }
    }

    public override int pop()
    {

            if(pointer < 0)
        {
            pointer=0;
        }
        int tmp1 = buffer[pointer];
        pointer--;
        return tmp1;
    }

            public override void show()
    {
            Console.Write("Stos: ");
        for(int i=0; i<=pointer; i++)
        {
            Console.Write(buffer[i]);
           Console.Write("  ");
        }
        Console.WriteLine("  ");
    }
}
class Program
{
    static void Main(string[] args)
    {
       Console.WriteLine("Michał Pawlikowski");
```

```
        Kolejka k1 = new Kolejka();
                        k1.push(1);
         k1.show();
         k1.push(2);
         k1.show();
         k1.push(3);
         k1.show();
         k1.pop();
         k1.show();
         Console.WriteLine(k1.getcount());
             k1.push(4);
         k1.push(5);
         k1.show();
         Console.WriteLine(k1.getcount());
             k1.clear();
         k1.show();
         Console.WriteLine(k1.getcount());
         Console.WriteLine(k1.isempty());
         k1.push(5);
         Console.WriteLine(k1.isempty());
             k1.push(5);
         k1.push(5);
         k1.push(5);
         k1.push(5);
         k1.push(5);
         k1.push(5);
         k1.push(5);
         k1.push(5);
         k1.push(5);
         k1.show();
         Console.WriteLine(k1.getcount());
         Console.WriteLine(k1.isfull());

        Stos s1 = new Stos();
         s1.push(5);
         s1.push(7);
         s1.show();
         s1.push(9);
         s1.show();
         s1.pop();
         s1.show();
         Console.WriteLine(s1.getcount());
         Console.WriteLine(s1.isfull());
         s1.pop();
         s1.pop();
         s1.show();
         Console.WriteLine(s1.isempty());
      }
    }
}
```

Testowanie

Run

```
            Console.WriteLine("Michał Pawlikowski");

            Kolejka k1 = new Kolejka();
             k1.push(1);
             k1.show();
             k1.push(2);
             k1.show();
             k1.push(3);
             k1.show();
             k1.pop();
             k1.show();
             Console.WriteLine(k1.getcount());
             k1.push(4);
             k1.push(5);
             k1.show();
             Console.WriteLine(k1.getcount());
             k1.clear();
             k1.show();
             Console.WriteLine(k1.getcount());
             Console.WriteLine(k1.isempty());
             k1.push(5);
             Console.WriteLine(k1.isempty());
             k1.push(5);
             k1.push(5);
             k1.push(5);
             k1.push(5);
             k1.push(5);
             k1.push(5);
             k1.push(5);
             k1.push(5);
             k1.push(5);
             k1.show();
             Console.WriteLine(k1.getcount());
             Console.WriteLine(k1.isfull());

            Stos s1 = new Stos();
             s1.push(5);
             s1.push(7);
             s1.show();
             s1.push(9);
             s1.show();
             s1.pop();
             s1.show();
             Console.WriteLine(s1.getcount());
             Console.WriteLine(s1.isfull());
             s1.pop();
             s1.pop();
             s1.show();
             Console.WriteLine(s1.isempty());
        }
    }
}
```

```
Michał Pawlikowski
Kolejka: 1
Kolejka: 1  2
Kolejka: 1  2  3
Kolejka: 2  3
2
Kolejka: 2  3  4  5
4
Kolejka:
0
True
False
Kolejka: 5  5  5  5  5  5  5  5  5  5
10
True
Stos: 5  7
Stos: 5  7  9
Stos: 5  7
2
False
Stos:
True
```

Zadanie na 5.0

```
using System;

namespace HelloWorld
{
   abstract class Container<T>
   {
      protected int pointer = -1;
      protected T[] buffer = new T[10];
      protected int size = 10;
      public abstract T pop();
      public abstract void push(T value);
      public abstract void show();
      public Container()
      {

      }
      public Container(int size_)
      {
         if (size > 0)
         {
            buffer = new T[size_];
            size = size_;
         }
      }


      public void clear()
      {
         buffer = new T[10];
         pointer = -1;
      }

      public int getcount()
      {
         return pointer + 1;
      }

      public Boolean isempty()
      {
         if (pointer == -1)
            return true;
         else
            return false;
      }

      public Boolean isfull()
      {
         if (pointer + 1 == size)
            return true;
         else
            return false;
      }

   }
   class Kolejka<T> : Container<T>
   {
      public Kolejka(int size): base(size)
      {

      }
      public override T pop()
```

```
    {
        T tmp = buffer[0];
        for (int i = 1; i <= pointer; i++)
        {
            buffer[i - 1] = buffer[i];
        }
        pointer--;
        if (pointer < -1)
        {
            pointer = -1;
        }
        return tmp;
    }
    public override void push(T value)
    {
        pointer++;
        if (pointer < size)
        {
            buffer[pointer] = value;
        }
        else
        {
            pointer = size-1;
        }
    }

    public override void show()
    {
        Console.Write("Kolejka: ");
        for (int i = 0; i <= pointer; i++)
        {
            Console.Write(buffer[i]);
            Console.Write("  ");
        }
        Console.WriteLine("  ");
    }
}
class Stos<T> : Container<T>
{
    public Stos(int size) : base(size)
    {

    }
    public override void push(T value)
    {
        pointer++;
        if (pointer < size)
        {
            buffer[pointer] = value;
        }
        else
        {
            pointer= size-1;
        }
    }

    public override T pop()
    {

        if (pointer < 0)
        {
            pointer = 0;
        }
```

```
        T tmp1 = buffer[pointer];
        pointer--;
        return tmp1;
    }

    public override void show()
    {
        Console.Write("Stos: ");
        for (int i = 0; i <= pointer; i++)
        {
            Console.Write(buffer[i]);
            Console.Write("  ");
        }
        Console.WriteLine("  ");
    }
}
class Program
{
    static void Main(string[] args)
    {
        Console.WriteLine("Michał Pawlikowski");

        Kolejka<int> k1 = new Kolejka<int>(5);
        k1.push(1);
        k1.show();
        k1.push(2);
        k1.show();
        k1.push(3);
        k1.show();
        k1.pop();
        k1.show();
        Console.WriteLine(k1.getcount());
        k1.push(4);
        k1.push(5);
        k1.show();
        Console.WriteLine(k1.getcount());
        k1.clear();
        k1.show();
        Console.WriteLine(k1.getcount());
        Console.WriteLine(k1.isempty());
        k1.push(5);
        Console.WriteLine(k1.isempty());
        k1.push(5);
        k1.push(5);
        k1.push(5);
        k1.push(5);
        k1.push(5);
        k1.push(5);
        k1.push(5);
        k1.push(5);
        k1.push(5);
        k1.show();
        Console.WriteLine(k1.getcount());
        Console.WriteLine(k1.isfull());

        Stos<int> s1 = new Stos<int>(5);
        s1.push(5);
        s1.push(7);
        s1.show();
        s1.push(9);
        s1.show();
        s1.pop();
        s1.show();
```

```
        Console.WriteLine(s1.getcount());
        Console.WriteLine(s1.isfull());
        s1.pop();
        s1.pop();
        s1.show();
        Console.WriteLine(s1.isempty());


        Kolejka<string> k2 = new Kolejka<string>(6);
        k2.push("M");
        k2.push("I");
        k2.push("C");
        k2.push("H");
        k2.push("A");
        k2.push("L");
        k2.show();
        Console.WriteLine(k2.getcount());
        Console.WriteLine(k2.isfull());
        k2.clear();
        k2.show();
        Console.WriteLine(k2.isempty());

      }
    }
}
```

Testowanie

```
        k1.push(3);
        k1.show();
        k1.pop();
        k1.show();
        Console.WriteLine(k1.getcount());
        k1.push(4);
        k1.push(5);
        k1.show();
        Console.WriteLine(k1.getcount());
        k1.clear();
        k1.show();
        Console.WriteLine(k1.getcount());
        Console.WriteLine(k1.isempty());
        k1.push(5);
        Console.WriteLine(k1.isempty());
        k1.push(5);
        k1.push(5);
        k1.push(5);
        k1.push(5);
        k1.push(5);
        k1.push(5);
        k1.push(5);
        k1.push(5);
        k1.push(5);
        k1.show();
        Console.WriteLine(k1.getcount());
        Console.WriteLine(k1.isfull());

        Stos<int> s1 = new Stos<int>(5);
        s1.push(5);
        s1.push(7);
        s1.show();
        s1.push(9);
        s1.show();
        s1.pop();
        s1.show();
        Console.WriteLine(s1.getcount());
        Console.WriteLine(s1.isfull());
        s1.pop();
        s1.pop();
        s1.show();
        Console.WriteLine(s1.isempty());


        Kolejka<string> k2 = new Kolejka<string>(6);
        k2.push("M");
        k2.push("I");
        k2.push("C");
        k2.push("H");
        k2.push("A");
        k2.push("L");
        k2.show();
        Console.WriteLine(k2.getcount());
        Console.WriteLine(k2.isfull());
        k2.clear();
        k2.show();
        Console.WriteLine(k2.isempty());

      }
    }
}
```

```
Michał Pawlikowski
Kolejka: 1
Kolejka: 1  2
Kolejka: 1  2  3
Kolejka: 2  3
2
Kolejka: 2  3  4  5
4
Kolejka:
0
True
False
Kolejka: 5  5  5  5  5
5
True
Stos: 5  7
Stos: 5  7  9
Stos: 5  7
2
False
Stos:
True
Kolejka: M  I  C  H  A  L
6
True
Kolejka:
True
```

Metoda clear() - czyści tablicę(stosu, kolejki)

Metoda getcount() - liczy długość tablicy(stosu, kolejki)

Metoda isempty() - sprawdza czy tablica(stosu, kolejki) jest pusta

Metoda isfull() - sprawdza czy tablica(stosu, kolejki) jest pełna

Parametr <T> umożliwia nam określenie dowolnego typu T do metody w czasie kompilacji, bez określania konkretnego typu w deklaracji metody lub klasy.

W kolejce po użyciu pop(), kasujemy pierwszą dodaną zmienną, natomiast w stosie kasujemy ostatnią dodaną do tablicy.