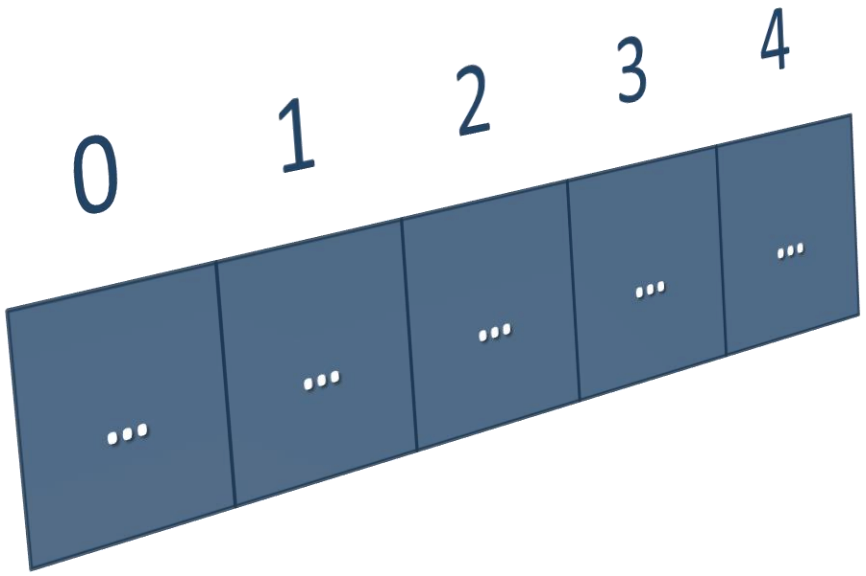
Стек и опашка

Последователност от елементи



SoftUni Foundation

CC BY NC SA

Проект "Отворено учебно съдържание по програмиране и ИТ", СофтУни Фондация
https://github.com/BG-IT-Edu

Курс "Структури от данни и алгоритми"

Софтуерни и хардуерни науки

Съдържание

1. **Структури от данни**
   - Линейни структури от данни
2. **Стек – Stack<T>**
   - `Push()`, `Pop()`, `Peek()`, `ToArray()`, `Contains()` и `Count`
3. **Опашка – Queue<T>**
   - `Enqueue()`, `Dequeue()`, `Peek()`, `ToArray()`, `Contains()` и `Count`

Структури от данни

Същност и видове

Структури от данни

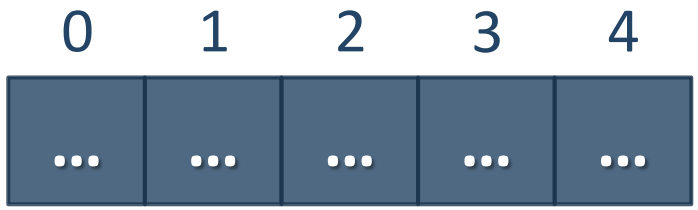
- **Структура от данни** == начин на организация на данните, който позволява **ефективен достъп** и **модификация**
- Примери за структура от данни:
  - Масив от числа – `int[]`
  - Списък от низове – `List<string>`
  - Опашка от хора – `Queue<Person>`

Защо стуктурите от данни са важни?

- **Структурите от данни** и алгоритмите стоят в основата на програмирането
- **Алгоритмичното мислене**, **решаването на задачи** и **стуктурите от данни** са важни за софтуерните инженери
  - C# програмистите трябва да знаят кога да използват `T[]` `List<T>`, `Stack<T>`, `Queue<T>`, `Dictionary<K, T>`, `LinkedList<T>`, `HashSet<T>`, `SortedDictionary<K, T>` и `SortedSet<T>`
- **Програмиране == алгоритми + структури от данни**!

Линейни структури от данни

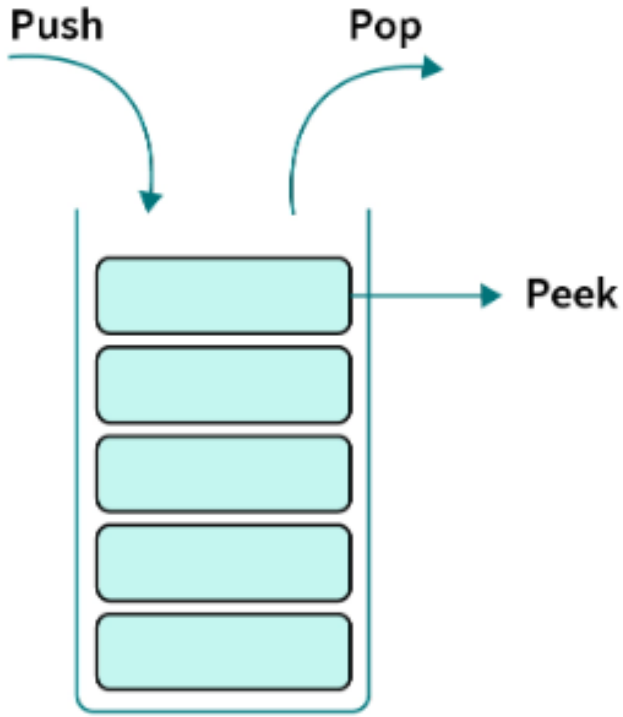
- **Линейни структури от данни**: масив, списък, стек, опашка



Масив/Списък

(индексирана група от елементи)

Опашка

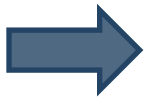


Стек

Списък с числа – Пример

- **Списък с числа**, представляващ последователност от суми на доходите:

```
List<double> incomes =
  new List<double> {
    150, 200, 70.50, 120
  };
```



Елемент	Стойност
incomes[0]	150
incomes[1]	200
incomes[2]	70.50
incomes[3]	120
incomes[4]	300

- Добавяне на **нов доход**:

```
incomes.Add(300);
```

- **Модифициране** на съществуващ доход:

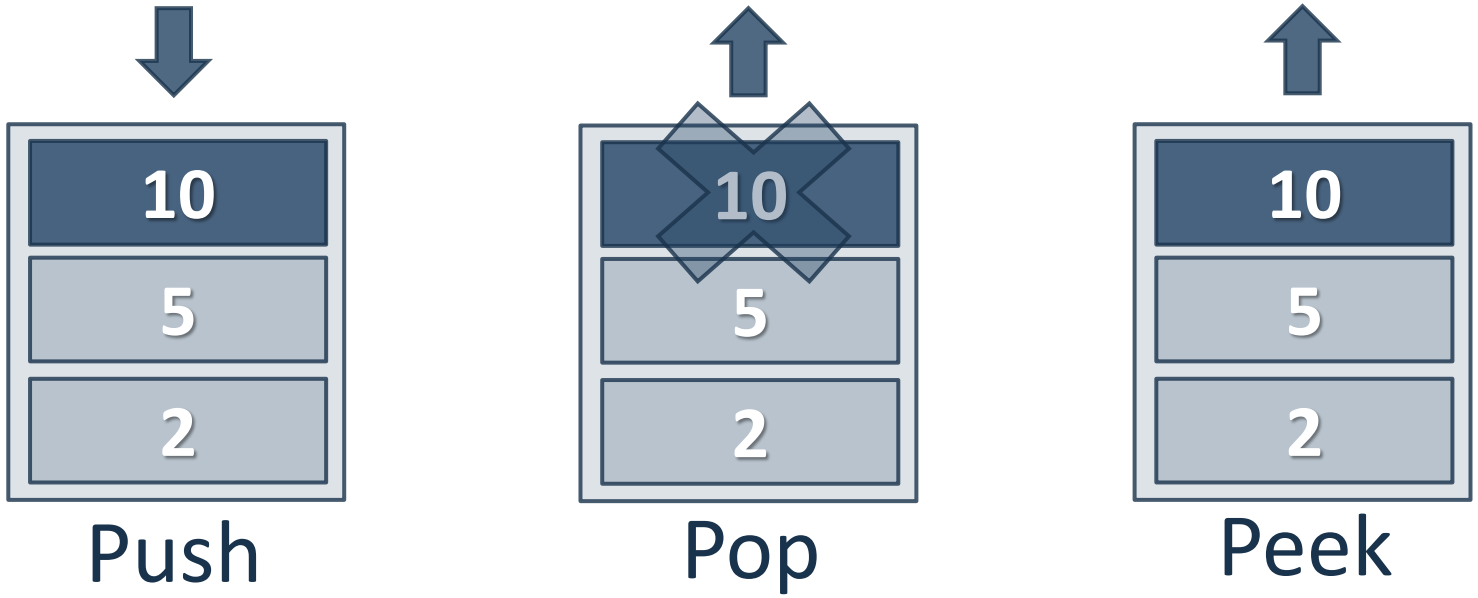
```
incomes[1] = 250;
```

Стек (Stack)

Push(), Peek(), Pop()

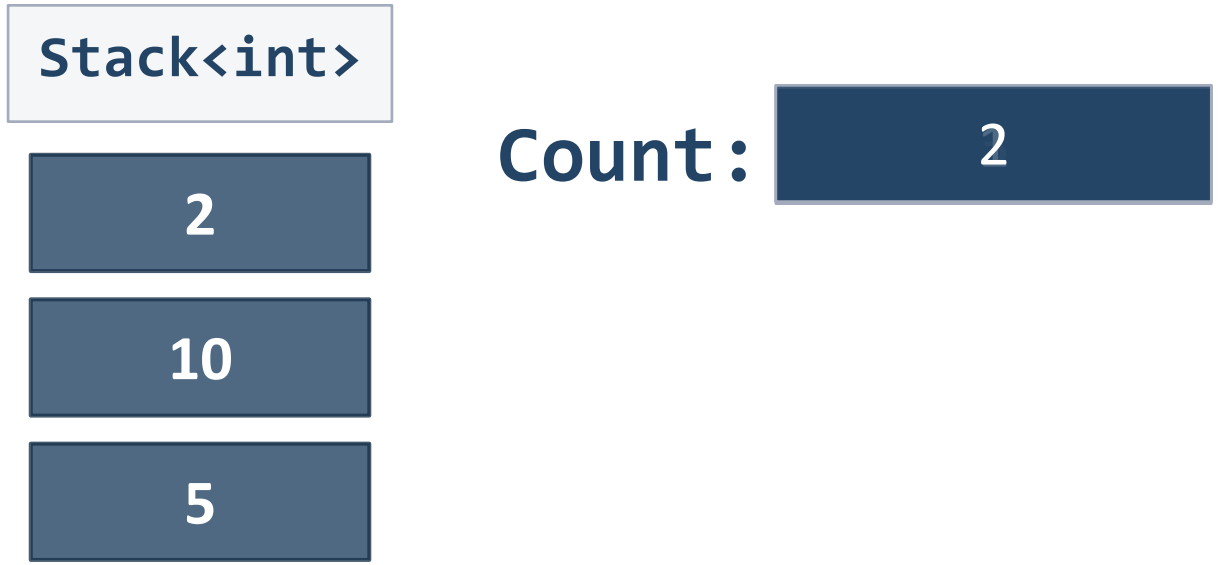
Стек – Абстрактен тип данни

- **Стекът** предоставя следните функции:
  - **Вкарване** на елемент
  - **Премахане** на последния елемент
  - **Връщане** на последния елемент без премахване

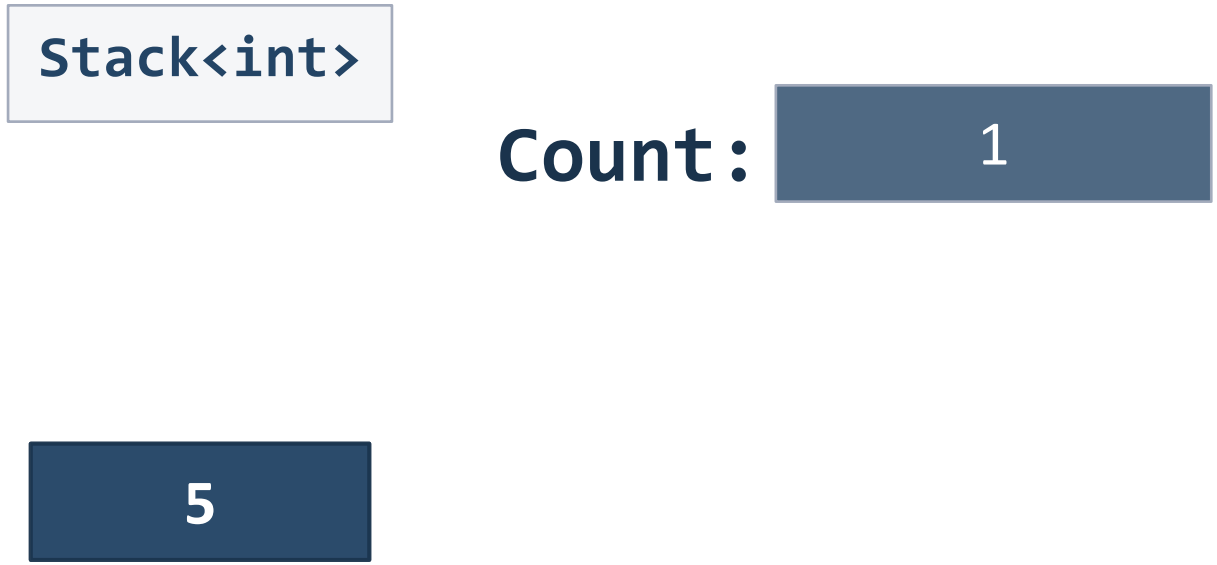


Push() – Вкарване на елемент в края

Pop() – Премахане и връщане на последния елемент



Peek() – Връщане на последния елемент



Стек – Методи

```
Stack<int> stack = new Stack<int>();

int count = stack.Count;
bool exists = stack.Contains(2);
int[] array = stack.ToArray();
stack.Clear();
stack.TrimExcess();
```

Връща броя на елементите

Проверява дали стекът съдържа елемента

Превръща стека в масив

Премахва всички елементи

Преоразмерява вътрешния масив

Задача: Обърнат низ

- Създайте програма, която:
  - Чете **вход от низ**
  - **Обръща** го чрез **стек**

```
I Love C#
```

```
#C evoL I
```

```
Stacks and Queues
```

```
seueuQ dna skcatS
```

Решение: Обърнат низ

```
var input = Console.ReadLine();
var stack = new Stack<char>();
foreach (var ch in input)
{
  stack.Push(ch);
}
while (stack.Count != 0)
{
  Console.Write(stack.Pop());
}
Console.WriteLine();
```

Тествайте решението си в Judge: https://judge.softuni.org/Contests/Practice/Index/4153#0

Задача: Сума на стек

- Пресметнете **сумата на числата от стека**
  - Преди това ще получавате команди
    - **Add:** добавя две числа
    - **Remove**: премахва n на брой числа

```
1 2 3 4
adD 5 6
REmove 3
eNd
```

```
Sum: 6
```

```
3 5 8 4 1 9
add 19 32
remove 10
add 89 22
end
```

```
Sum: 192
```

Решение: Сума на стек (1)

SoftUni Foundation

```
string[] input =
     Console.ReadLine().Split().Select(int.Parse).ToArray();
Stack<int> stack = new Stack<int>(input);
string command = Console.ReadLine().ToLower();

while (command != "end")
{
  string[] tokens = commandInfo.Split().ToArray();
  string action = tokens[0].ToLower();
  if (action == "add")
    // TODO: Добавете числата
  else if(…)
```

Решение: Сума на стек (2)

```
    else if(action == "remove") {
        var countOfRemovedNums = int.Parse(tokens[1]);
        if (stack.Count < countOfRemovedNums)
            continue;
        for (int i = 0; i < countOfRemovedNums; i++)
            stack.Pop();
    }
    command = Console.ReadLine().ToLower();
}

var sum = stack.Sum();
Console.WriteLine($"Sum: {sum}");
```

Тествайте решението си в Judge: https://judge.softuni.org/Contests/Practice/Index/4153#1

Задача: Прост калкулатор

- Създайте **прост калкулатор**, който може да пресмята прости изрази (само събиране и изваждане)

`2 + 5 + 10 - 2 - 1`	⇨	`14`
`2 - 2 + 5`	⇨	`5`
`2 - 1 + 5`	⇨	`6`
`2 - 0 + 5`	⇨	`7`

Решение: Прост калкулатор (1)

SoftUni Foundation

```
var input = Console.ReadLine();
var values = input.Split(' ');
var stack = new Stack<string>(values.Reverse());
while (stack.Count > 1)
{
  int first = int.Parse(stack.Pop());
  string operator = stack.Pop();
  int second = int.Parse(stack.Pop());
  // TODO: Добавете switch за операциите
}
Console.WriteLine(stack.Pop());
```

Решение: Прост калкулатор (2)

```
switch (operator)
{
   case "+":
      stack.Push((first + second).ToString());
      break;
   case "-":
      stack.Push((first - second).ToString());
      break;
}
```

Тествайте решението си в Judge: https://judge.softuni.org/Contests/Practice/Index/4153#2

Задача: Математически скоби

- Даден е **аритметичен израз** със скоби (с **влагане**)
- **Извлечете всички подизрази** в скоби

```
1 + (2 - (2 + 3) * 4 / (3 + 1)) * 5
```

```
(2 + 3)
(3 + 1)
(2 - (2 + 3) * 4 / (3 + 1))
```

Решение: Математически скоби

```
var input = Console.ReadLine();
var stack = new Stack<int>();
for (int i = 0; i < input.Length; i++) {
  char ch = input[i];
  if (ch == '(') {
    stack.Push(i);
  } else if (ch == ')') {
    int startIndex = stack.Pop();
    string contents = input.Substring(
                      startIndex, i - startIndex + 1);
    Console.WriteLine(contents);
}
```

Тествайте решението си в Judge: https://judge.softuni.org/Contests/Practice/Index/4153#3

Опашка (Queue)

Enqueue(), Dequeue(), Peek()

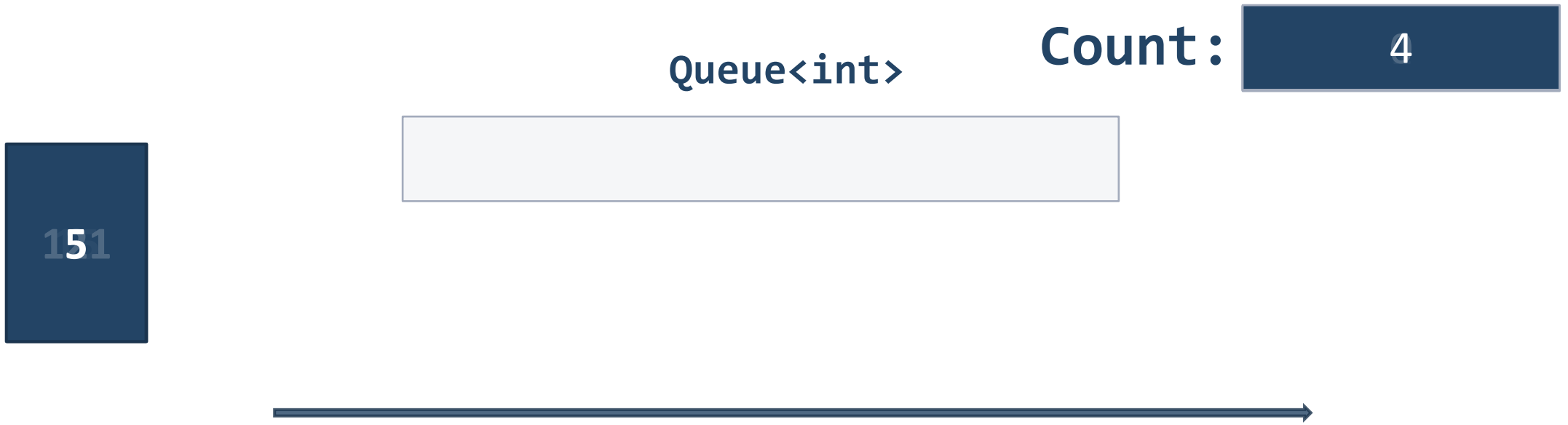
Опашка – Абстрактен тип данни

- **Опашката** осигурява следните функции:
  - **Добавяне** на елемент в края на опашката

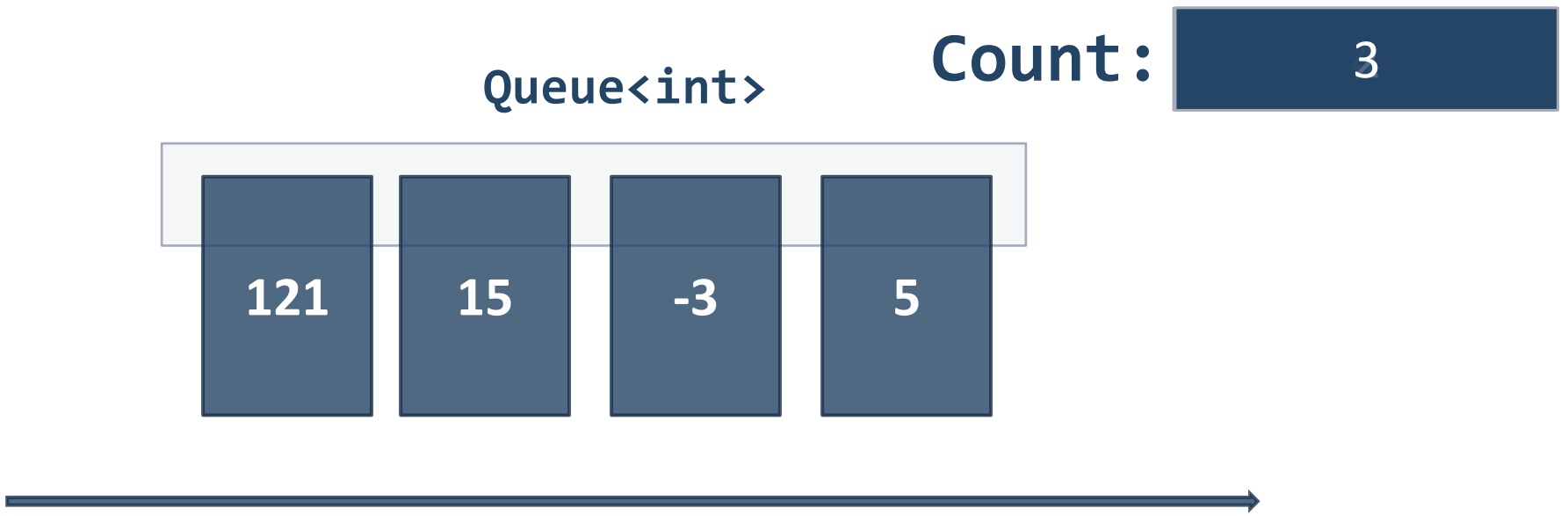
  - **Премахване** на първия елемент

  - **Връщане** на първия елемент без да го премахва

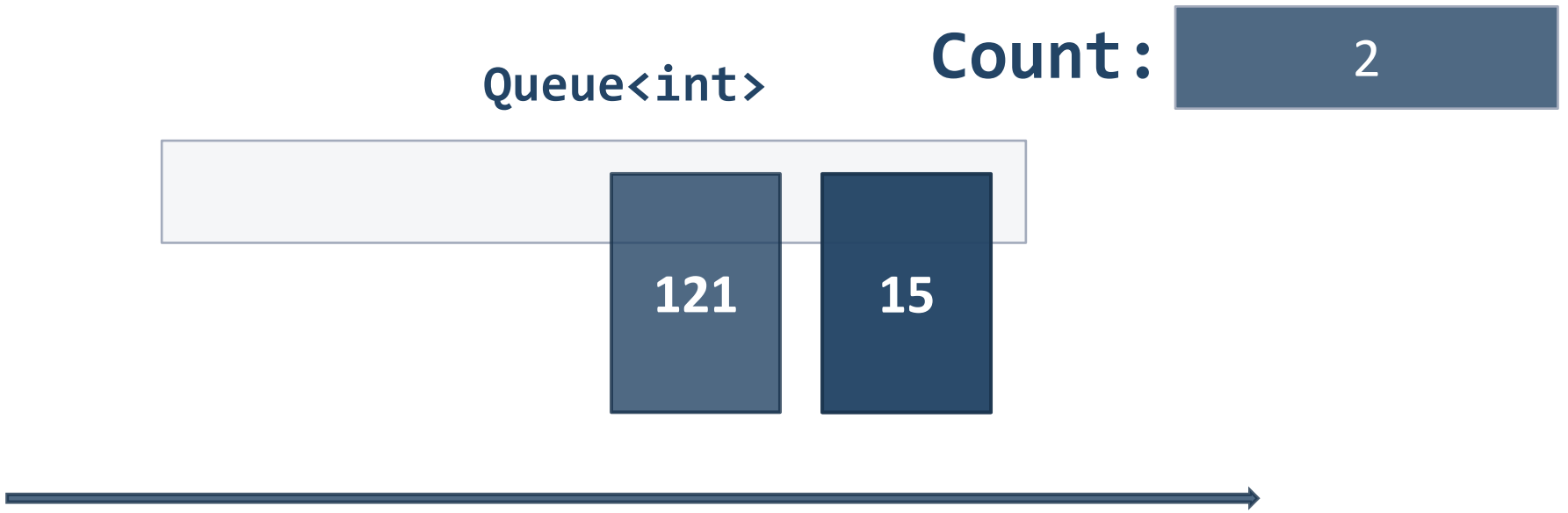
Enqueue() – Вкарване на елемент в края



Dequeue() – Премахане и връщане на първия елемент



Peek() – Връщане на първия елемент без премахване



Опашка – Методи

```
Queue<int> queue = new
Queue<int>();
int count = queue.Count;
bool exists = queue.Contains(2);
int[] array = queue.ToArray();
queue.Clear();
queue.TrimExcess();
```

Връща броя на елементите

Проверява дали опашката съдържа елемента

Превръща опашката в масив

Преоразмерява вътрешния масив

Премахва всички елементи

Задача: Задръстване

- Колите чакат на **опашка** пред **светофар**
- На всяка **зелена светлина** n коли **минават** през кръстовището
- След **командата "end"** принтирайте **колко коли** са **минали**

```
3
Enzo's car
Jade's car
Mercedes CLS
Audi
green
BMW X5
green
end
```

```
Enzo's car passed!
Jade's car passed!
Mercedes CLS passed!
Audi passed!
BMW X5 passed!
5 cars passed the crossroads.
```

Решение: Задръстване

```
int n = int.Parse(Console.ReadLine());
var queue = new Queue<string>();
int count = 0;
string command;
while ((command = Console.ReadLine()) != "end")
{
  if (command == "green")
    // TODO: Добавете логика за зелена светлина
  else
    queue.Enqueue(command);
}
Console.WriteLine($"{count} cars passed the crossroads.");
```

Тествайте решението си в Judge: https://judge.softuni.org/Contests/Practice/Index/4153#5

Задача: Горещ картоф

- Деца са се **наредили в кръг** и си подават горещ картоф по **часовниковата стрелка**.
- При всяко n-то хвърляне **дете се отстранява**, докато **остане само едно**
- **След отстраняване на дете** картофът се **предава**
- Принтирайте последното дете:

```
Alva James William
2
```

```
Removed James
Removed Alva
Last is William
```

Решение: Горещ картоф

```
var children = Console.ReadLine().Split(' ');
var number = int.Parse(Console.ReadLine());
Queue<string> queue = new Queue<string>(children);
while (queue.Count != 1)
{
  for (int i = 1; i < number; i++)
  {
    queue.Enqueue(queue.Dequeue());
  }
  Console.WriteLine($"Removed {queue.Dequeue()}");
}
Console.WriteLine($"Last in {queue.Dequeue()}");
```

Копираме елементи от колекцията и запазваме реда им

Тествайте решението си в Judge : https://judge.softuni.org/Contests/Practice/Index/4153#4

Какво научихме днес?

SoftUni Foundation

- Линейната структура от данни == поредица от елементи
  - **Stack<T>**
  - **Queue<T>**
- **Работа с вградени методи**

Въпроси?

Проект "Отворено учебно съдържание по програмиране и ИТ", СофтУни Фондация (лиценз CC-BY-NC-SA)