

```

1  package main
2
3  import      "fmt"
4
5  // Изменяем только данные слайса-параметра
6  func f1(c []int) {
7      for i, _ := range c {
8          c[i] +=100
9      }
10     fmt.Println(c, len(c), cap(c))
11 }
12
13 func f2(pc *[]int) {
14     for i, _ := range *pc {
15         (*pc)[i] +=100
16     }
17     fmt.Println(*pc, len(*pc), cap(*pc))
18 }
19
20 func f3(c []int) []int {
21     for i, _ := range c {
22         c[i] +=100
23     }
24     fmt.Println(c, len(c), cap(c))
25     return c
26 }
27
28 func main() {
29     a:= [...]int {0,1,2,3,4,5,6,7,8,9}
30     fmt.Println(a)                // [0 1 2 3 4 5 6 7 8 9]
31
32     arr:= a
33     b:= arr[2:7]
34     fmt.Println(b, len(b), cap(b)) // [2 3 4 5 6] 5 8
35     f1(b)                          // [102 103 104 105 106] 5 8
36     fmt.Println(b, len(b), cap(b)) // [102 103 104 105 106] 5 8
37     fmt.Println(arr)               // [0 1 102 103 104 105 106 7 8 9]
38
39     arr = a
40     b = arr[2:7]
41     fmt.Println(b, len(b), cap(b)) // [2 3 4 5 6] 5 8
42     f2(&b)                          // [102 103 104 105 106] 5 8
43     fmt.Println(b, len(b), cap(b)) // [102 103 104 105 106] 5 8
44     fmt.Println(arr)               // [0 1 102 103 104 105 106 7 8 9]
45
46     arr = a
47     b = arr[2:7]
48     fmt.Println(b, len(b), cap(b)) // [2 3 4 5 6] 5 8
49     b = f3(b)                      // [102 103 104 105 106] 5 8
50     fmt.Println(b, len(b), cap(b)) // [102 103 104 105 106] 5 8
51     fmt.Println(arr)               // [0 1 102 103 104 105 106 7 8 9]
52 }

```

```

1  package main
2
3  import      "fmt"
4
5  // Изменяем только длину слайса-параметра
6  func f1(c []int) {
7      c = c[:cap(c)]
8      fmt.Println(c, len(c), cap(c))
9  }
10
11 func f2(pc *[]int) {
12     *pc = (*pc)[:cap(*pc)]
13     fmt.Println(*pc, len(*pc), cap(*pc))
14 }
15
16 func f3(c []int) []int {
17     c = c[:cap(c)]
18     fmt.Println(c, len(c), cap(c))
19     return c
20 }
21
22 func main() {
23     a:= [...]int {0,1,2,3,4,5,6,7,8,9}
24     fmt.Println(a)                                // [0 1 2 3 4 5 6 7 8 9]
25
26     arr:= a
27     b:= arr[2:7]
28     fmt.Println(b, len(b), cap(b))                // [2 3 4 5 6] 5 8
29     f1(b)                                          // [2 3 4 5 6 7 8 9] 8 8
30     fmt.Println(b, len(b), cap(b))                // [2 3 4 5 6] 5 8
31     fmt.Println(arr)                              // [0 1 2 3 4 5 6 7 8 9]
32
33     arr = a
34     b = arr[2:7]
35     fmt.Println(b, len(b), cap(b))                // [2 3 4 5 6] 5 8
36     f2(&b)                                         // [2 3 4 5 6 7 8 9] 8 8
37     fmt.Println(b, len(b), cap(b))                // [2 3 4 5 6 7 8 9] 8 8
38     fmt.Println(arr)                              // [0 1 2 3 4 5 6 7 8 9]
39
40     arr = a
41     b = arr[2:7]
42     fmt.Println(b, len(b), cap(b))                // [2 3 4 5 6] 5 8
43     b = f3(b)                                     // [2 3 4 5 6 7 8 9] 8 8
44     fmt.Println(b, len(b), cap(b))                // [2 3 4 5 6 7 8 9] 8 8
45     fmt.Println(arr)                              // [0 1 2 3 4 5 6 7 8 9]
46 }

```

01b.go

```

1  package main
2
3  import      "fmt"
4
5  // Изменяем длину слайса-параметра, а затем данные
6  func f1(c []int) {
7      c = c[:cap(c)]
8      for i, _ := range c {
9          c[i] +=100
10     }
11     fmt.Println(c, len(c), cap(c))
12 }
13
14 func f2(pc *[]int) {
15     *pc = (*pc)[:cap(*pc)]
16     for i, _ := range *pc {
17         (*pc)[i] +=100
18     }
19     fmt.Println(*pc, len(*pc), cap(*pc))
20 }
21
22 func f3(c []int) []int {
23     c = c[:cap(c)]
24     for i, _ := range c {
25         c[i] +=100
26     }
27     fmt.Println(c, len(c), cap(c))
28     return c
29 }
30
31 func main() {
32     a:= [...]int {0,1,2,3,4,5,6,7,8,9}
33     fmt.Println(a)                // [0 1 2 3 4 5 6 7 8 9]
34
35     arr:= a
36     b:= arr[2:7]
37     fmt.Println(b, len(b), cap(b)) // [2 3 4 5 6] 5 8
38     f1(b)                          // [102 103 104 105 106 107 108 109] 8 8
39     fmt.Println(b, len(b), cap(b)) // [102 103 104 105 106] 5 8
40     fmt.Println(arr)               // [0 1 102 103 104 105 106 107 108 109]
41
42     arr = a
43     b = arr[2:7]
44     fmt.Println(b, len(b), cap(b)) // [2 3 4 5 6] 5 8
45     f2(&b)                         // [102 103 104 105 106 107 108 109] 8 8
46     fmt.Println(b, len(b), cap(b)) // [102 103 104 105 106 107 108 109] 8 8
47     fmt.Println(arr)               // [0 1 102 103 104 105 106 107 108 109]
48
49     arr = a
50     b = arr[2:7]
51     fmt.Println(b, len(b), cap(b)) // [2 3 4 5 6] 5 8
52     b = f3(b)                      // [102 103 104 105 106 107 108 109] 8 8
53     fmt.Println(b, len(b), cap(b)) // [102 103 104 105 106 107 108 109] 8 8
54     fmt.Println(arr)               // [0 1 102 103 104 105 106 107 108 109]
55 }

```

01c.go

```

1 package main
2
3 import     "fmt"
4
5 // Добавляем в слайс-параметр данные по одному,
6 // пока не произойдёт перераспределения памяти
7 func f1(c []int) {
8     ccap:= cap(c)
9     for cap(c) == ccap {
10         c = append(c, len(c)*10)
11     }
12     fmt.Println(c, len(c), cap(c))
13 }
14
15 func f2(pc *[]int) {
16     ccap:= cap(*pc)
17     for cap(*pc) == ccap {
18         *pc = append(*pc, len(*pc)*10)
19     }
20     fmt.Println(*pc, len(*pc), cap(*pc))
21 }
22
23 func f3(c []int) []int {
24     ccap:= cap(c)
25     for cap(c) == ccap {
26         c = append(c, len(c)*10)
27     }
28     fmt.Println(c, len(c), cap(c))
29     return c
30 }
31
32 func main() {
33     a:= [...]int {0,1,2,3,4,5,6,7,8,9}
34     fmt.Println(a)                                // [0 1 2 3 4 5 6 7 8 9]
35
36     arr:= a
37     b:= arr[2:7]
38     fmt.Println(b, len(b), cap(b))                // [2 3 4 5 6] 5 8
39     f1(b)                                           // [2 3 4 5 6 50 60 70 80] 9 16
40     fmt.Println(b, len(b), cap(b))                // [2 3 4 5 6] 5 8
41     fmt.Println(arr)                               // [0 1 2 3 4 5 6 50 60 70]
42
43     arr = a
44     b = arr[2:7]
45     fmt.Println(b, len(b), cap(b))                // [2 3 4 5 6] 5 8
46     f2(&b)                                           // [2 3 4 5 6 50 60 70 80] 9 16
47     fmt.Println(b, len(b), cap(b))                // [2 3 4 5 6 50 60 70 80] 9 16
48     fmt.Println(arr)                               // [0 1 2 3 4 5 6 50 60 70]
49
50     arr = a
51     b = arr[2:7]
52     fmt.Println(b, len(b), cap(b))                // [2 3 4 5 6] 5 8
53     b = f3(b)                                       // [2 3 4 5 6 50 60 70 80] 9 16
54     fmt.Println(b, len(b), cap(b))                // [2 3 4 5 6 50 60 70 80] 9 16
55     fmt.Println(arr)                               // [0 1 2 3 4 5 6 50 60 70]
56 }

```

```

1 package main
2
3 import "fmt"
4
5 // Добавляем в слайс-параметр данные группой, размер
6 // которой требует перераспределения памяти
7 func f1(c []int) {
8     cc:= make([]int, cap(c) - len(c) + 1)
9     c = append(c, cc...)
10    fmt.Println(c, len(c), cap(c))
11 }
12
13 func f2(pc *[]int) {
14     cc:= make([]int, cap(*pc) - len(*pc) + 1)
15     *pc = append(*pc, cc...)
16     fmt.Println(*pc, len(*pc), cap(*pc))
17 }
18
19 func f3(c []int) []int {
20     cc:= make([]int, cap(c) - len(c) + 1)
21     c = append(c, cc...)
22     fmt.Println(c, len(c), cap(c))
23     return c
24 }
25
26 func main() {
27     a:= [...]int {0,1,2,3,4,5,6,7,8,9}
28     fmt.Println(a) // [0 1 2 3 4 5 6 7 8 9]
29
30     arr:= a
31     b:= arr[2:7]
32     fmt.Println(b, len(b), cap(b)) // [2 3 4 5 6] 5 8
33     f1(b) // [2 3 4 5 6 0 0 0 0] 9 16
34     fmt.Println(b, len(b), cap(b)) // [2 3 4 5 6] 5 8
35     fmt.Println(arr) // [0 1 2 3 4 5 6 7 8 9]
36
37     arr = a
38     b = arr[2:7]
39     fmt.Println(b, len(b), cap(b)) // [2 3 4 5 6] 5 8
40     f2(&b) // [2 3 4 5 6 0 0 0 0] 9 16
41     fmt.Println(b, len(b), cap(b)) // [2 3 4 5 6 0 0 0 0] 9 16
42     fmt.Println(arr) // [0 1 2 3 4 5 6 7 8 9]
43
44     arr = a
45     b = arr[2:7]
46     fmt.Println(b, len(b), cap(b)) // [2 3 4 5 6] 5 8
47     b = f3(b) // [2 3 4 5 6 0 0 0 0] 9 16
48     fmt.Println(b, len(b), cap(b)) // [2 3 4 5 6 0 0 0 0] 9 16
49     fmt.Println(arr) // [0 1 2 3 4 5 6 7 8 9]
50 }

```

01e.go

```

1  package main
2
3  // Запуск из командной строки:
4  //      go test -bench . benchmark_sample_test.go
5  // В данном случае benchmark_sample_test.go - это имя файла,
6  // в котором находится данная программа. Имя тестируемого
7  // файла обязательно должно заканчиваться на _test
8
9  import "testing"
10
11 func lala(s []int) {
12     for i, x := range(s) {
13         s[i] = (x-1)*x*(x+1)
14     }
15 }
16 // Названия тестируемых функций должны начинаться на Benchmark,
17 // за которым идёт название, начинающееся с большой буквы
18
19 // Оценивается функция BenchmarkLala.
20 func BenchmarkLala(b *testing.B) {
21     var s [100000]int
22     for i:= 0; i < b.N; i++ {
23         lala(s[:])
24     }
25 }
26
27 // Оценивается функция BenchmarkLalala.
28 func BenchmarkLalala(b *testing.B) {
29     var s [100000]int
30     for k:= 0; k < b.N; k++ {
31         for i, x := range s {
32             s[i] = (x-1)*x*(x+1)
33         }
34     }
35 }
36
37 // Оценивается функция BenchmarkLalala2.
38 func BenchmarkLalala2(b *testing.B) {
39     s:= make([]int, 100000)
40     for k:= 0; k < b.N; k++ {
41         for i, x := range s {
42             s[i] = (x-1)*x*(x+1)
43         }
44     }
45 }
46
47 // OUTPUT
48 // goos: windows
49 // goarch: amd64
50 // BenchmarkLala-4          12103          99203 ns/op
51 // BenchmarkLalala-4        7063          161336 ns/op
52 // BenchmarkLalala2-4       12094          98955 ns/op
53 // PASS
54 // ok      command-line-arguments  5.765s

```

benchmark_sample_test.go

```

1  package main
2
3  import (
4      "testing"
5      "fmt"
6  )
7
8  func lala(s []int) {
9      for i, x := range(s) {
10         s[i] = (x-1)*x*(x+1)
11     }
12 }
13
14 func main() {
15     var s [100000]int
16     res:= testing.Benchmark(
17         func(b *testing.B) {
18             for i:= 0; i < b.N; i++ {
19                 lala(s[:])
20             }
21         })
22     fmt.Println(res)
23 }

```

benchmark_sample_02.go

```

1  // Запуск:
2  //      go test -bench . insert_test.go
3
4  // файл должен иметь суффикс _test
5  // оцениваемые функции должны начинаться с Benchmark,
6  // за которым идёт название, начинающееся с большой буквы
7
8  package insert
9
10 import (
11     "container/list"
12     "testing"
13 )
14
15 var (
16     slice10    = slice(10)
17     slice100   = slice(100)
18     slice1k    = slice(1000)
19     slice10k   = slice(10000)
20     slice100k  = slice(100000)
21     slice1m    = slice(1000000)
22 )
23
24 func slice(size int) []byte { return make([]byte, size) }
25
26 func BenchmarkInsertHeadSlice10(b *testing.B)      {
27     benchmarkInsertSlice(b, slice10) }
28 func BenchmarkInsertHeadSlice100(b *testing.B)     {
29     benchmarkInsertSlice(b, slice100) }
30 func BenchmarkInsertHeadSlice1000(b *testing.B)    {
31     benchmarkInsertSlice(b, slice1k) }
32 func BenchmarkInsertHeadSlice10000(b *testing.B)   {
33     benchmarkInsertSlice(b, slice10k) }

```

```

34 func BenchmarkInsertHeadSlice100000(b *testing.B) {
35     benchmarkInsertSlice(b, slice100k) }
36 func BenchmarkInsertHeadSlice1000000(b *testing.B) {
37     benchmarkInsertSlice(b, slice1m) }
38
39 func benchmarkInsertSlice(b *testing.B, a []byte) {
40     c := cap(a)
41     for n := 0; n < b.N; n++ {
42         // insert at i trick:
43         // a = append(a[:i], append([]T{x}, a[i:]...)...)
44         // could be simplified as we insert to head
45         a = append([]byte{0}, a...)
46         // reset a
47         a = a[:c:c]
48     }
49 }
50
51 var (
52     linkedList10    = linkedList(10)
53     linkedList100   = linkedList(100)
54     linkedList1k    = linkedList(1000)
55     linkedList10k   = linkedList(10000)
56     linkedList100k  = linkedList(100000)
57     linkedList1m    = linkedList(1000000)
58 )
59
60 func linkedList(size int) *list.List {
61     l := list.New()
62     for i := 0; i < size; i++ {
63         l.PushBack(0)
64     }
65     return l
66 }
67
68 func BenchmarkInsertHeadList10(b *testing.B)      {
69     benchmarkInsertList(b, linkedList10) }
70 func BenchmarkInsertHeadList100(b *testing.B)     {
71     benchmarkInsertList(b, linkedList100) }
72 func BenchmarkInsertHeadList1000(b *testing.B)    {
73     benchmarkInsertList(b, linkedList1k) }
74 func BenchmarkInsertHeadList10000(b *testing.B)   {
75     benchmarkInsertList(b, linkedList10k) }
76 func BenchmarkInsertHeadList100000(b *testing.B) {
77     benchmarkInsertList(b, linkedList100k) }
78 func BenchmarkInsertHeadList1000000(b *testing.B) {
79     benchmarkInsertList(b, linkedList1m) }
80
81 func benchmarkInsertList(b *testing.B, l *list.List) {
82     for n := 0; n < b.N; n++ {
83         l.PushFront(0)
84         // reset l
85         l.Remove(l.Front())
86     }
87 }

```

insert_test.go