

# Функции

доц. д-р Нора Ангелова

---

# Функции

Дефиниция на функция

- Заглавие

<тип\_на\_функция> <име\_на\_функция> (<формални\_параметри>)

<тип\_на\_функция> ::= име на тип.

Типът на резултата от изпълнението на функцията.

Ако функцията не връща резултат – тип void.

<име\_на\_функция> ::= идентификатор.

<формални\_параметри> ::= множество от параметри. Те изпълняват ролята на входните данни на функцията.

Функцията може да няма формални параметри.

<параметър> ::= <име\_на\_тип> <име\_на\_параметър>.

- Тяло ::= редица от оператори и дефиниции оградени в { }.

{ <тяло> }

# Функции

Декларация на функция

<тип\_на\_функция> <име\_на\_функция>(<формални\_параметри>);

- Декларацията на функцията се нарича също **прототип**.
- Имената на формалните параметри в декларацията могат да се пропускат.

Пример:

```
void testFunction(int);
```

```
int testFunction2(double);
```

# Функции

Извикване на функция

```
<име_на_функция>( <фактически_параметри> );
```

Пример:

```
testFunction(5);
```

# Функции

- Преди да се извика една функция, тя трябва да е декларирана/дефинирана.

Ако дадена функция `example` извиква функцията `testFunction`, функцията `testFunction` трябва да бъде декларирана/дефинирана преди `example`.

# ФУНКЦИИ

Пример:

```
void printParam(int param) {  
    std::cout << param << std::endl;  
}
```

```
int main() {  
    printParam(5);  
  
    return 0;  
}
```

# ФУНКЦИИ

Пример:

```
void printParam(int param);
```

```
int main() {  
    printParam(5);
```

```
    return 0;  
}
```

```
void printParam(int param) {  
    std::cout << param << std::endl;  
}
```

# ФУНКЦИИ

Пример:

```
void printParam(int);
```

```
int main() {  
    printParam(5);
```

```
    return 0;  
}
```

```
void printParam(int param) {  
    std::cout << param << std::endl;  
}
```



# Функции

Свързване на формални с фактически параметри:

- Формален параметър – стойност.
- Формален параметър – указател.

# ФУНКЦИИ

```
void testFunction(int param) {  
    ...  
}  
    ...
```

**param**



```
int value = 5;  
testFunction(value);
```

**value**

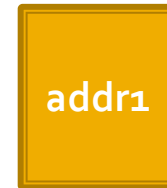


**addr1**

# ФУНКЦИИ

```
void testFunction(int * param) {  
    ...  
}  
    ...
```

**param**



```
int value = 5;  
testFunction(&value);  
->
```

**value**



**addr1**

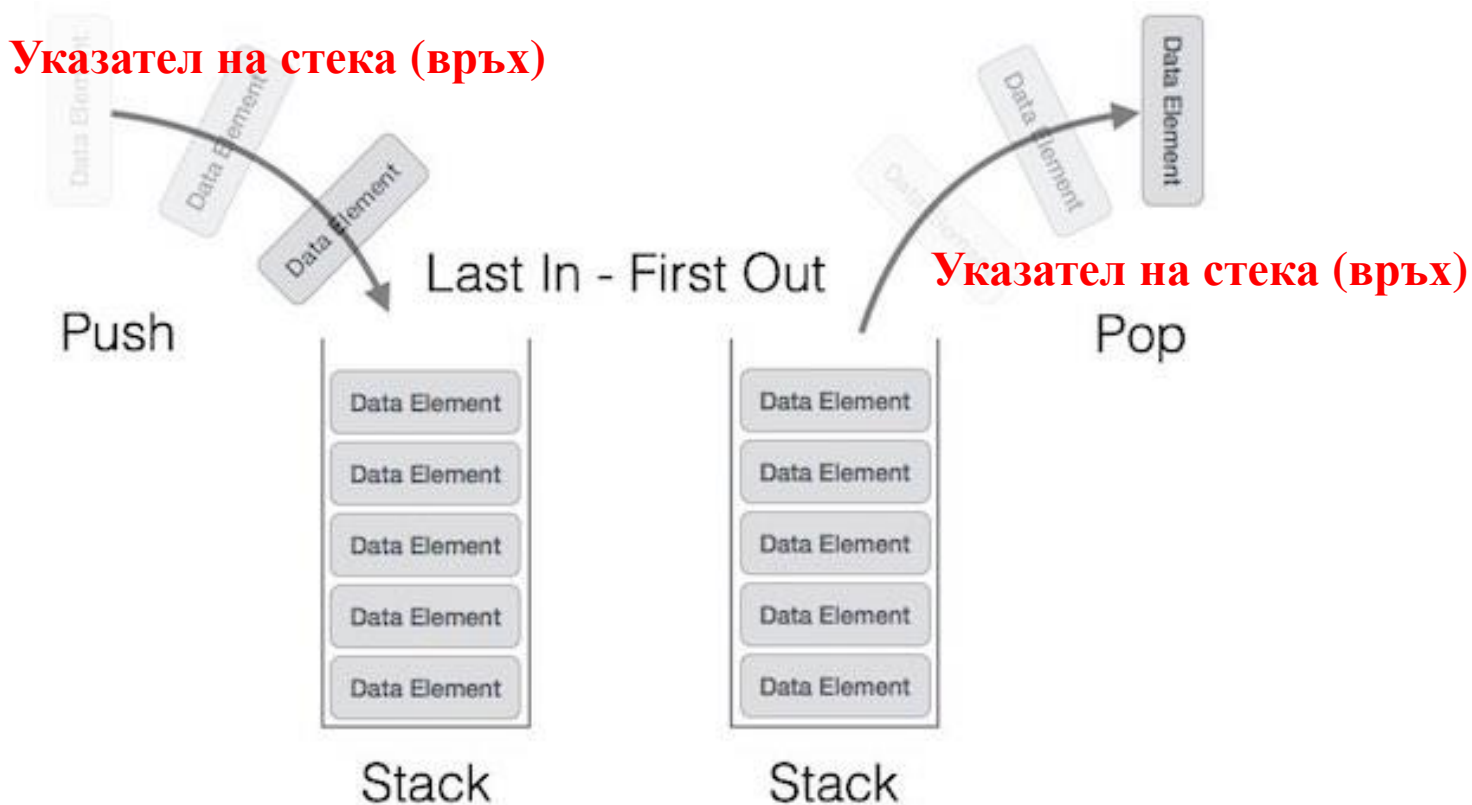
# Функции

Разпределение на оперативната памет за програма:

- Програмен код – записан е изпълнимият код на всички функции, изграждащи програмата.
- Статични данни – записани са глобални и статични обекти.
- Динамични данни – чрез средства за динамично разпределение на паметта се заделя и освобождава памет в процеса на изпълнение на програмата (не преди това). Тази памет е от областта на динамичните данни.
- Програмен стек – записани са данните за функциите.

# Стек

- Хомогенна линейна структура от елементи.
- „последен влязъл - пръв излязъл“ (LIFO).
- Пряк достъп, включване, изключване на елемент.



# Функции

## Програмен стек

- Елементите на стека са „блокове“ от памет.
- Записани са данните за функциите.
- Наричат се още **стекови рамки**.

# Функции

## Програмен стек

- В дъното на стека е стековата рамка на `main`.
- На върха на стека е стековата рамка на функцията, която се обработва в момента.
- Под нея е стековата рамка на функцията, извикала функцията, обработваща се в момента.

# Функции

Свързване на формални с фактически параметри:

- Формален параметър – стойност

*В стековата рамка на функцията:*

*За формалния параметър се отделя толкова памет, колкото типът му изисква и в нея се копира стойността на фактическия параметър.*

- Формален параметър – указател

*В стековата рамка на функцията:*

*За формалния параметър се отделя памет за един указател. В нея се записва стойността на фактическия параметър, който трябва да бъде адрес на променлива.*



# Функции

```
int gcd(int x, int y) {
    while(x != y) {
        if (x > y) {
            x = x - y;
        } else {
            y = y - x;
        }
    }

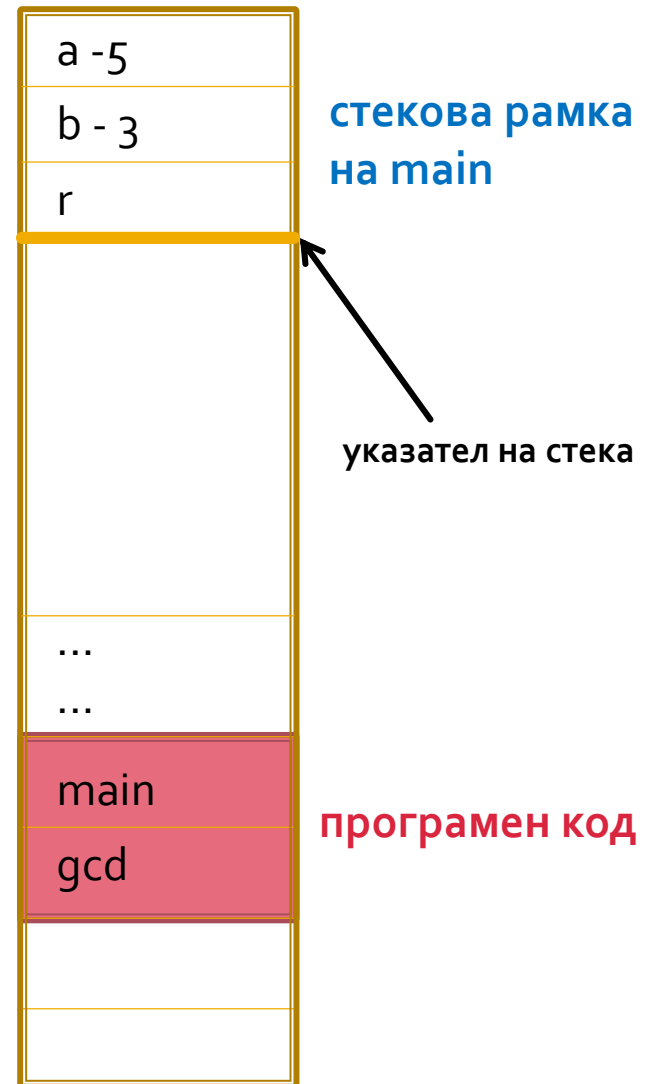
    return x;
}

int main() {
    int a = 5, b = 3;
    int r = gcd(a, b);

    std::cout << r << std::endl;
    return 0;
}
```

# Функции

```
int gcd(int x, int y) {  
    while(x != y) {  
        if (x > y) {  
            x = x - y;  
        } else {  
            y = y - x;  
        }  
    }  
  
    return x;  
}  
  
int main() {  
    int a = 5, b = 3;  
    int r = gcd(a, b);  
  
    std::cout << r << std::endl;  
    return 0;  
}
```



# Функции

```
int gcd(int x, int y) {  
    while(x != y) {  
        if (x > y) {  
            x = x - y;  
        } else {  
            y = y - x;  
        }  
    }  
  
    return x;  
}  
  
int main() {  
    int a = 5, b = 3;  
    int r = gcd(a, b);  
  
    cout << r << endl;  
    return 0;  
}
```



# Функции

```
int gcd(int x, int y) {
    while(x != y) {
        if (x > y) {
            x = x - y;
        } else {
            y = y - x;
        }
    }

    return x;
}

int main() {
    int a = 5, b = 3;
    int r = gcd(a, b);

    cout << r << endl;
    return 0;
}
```



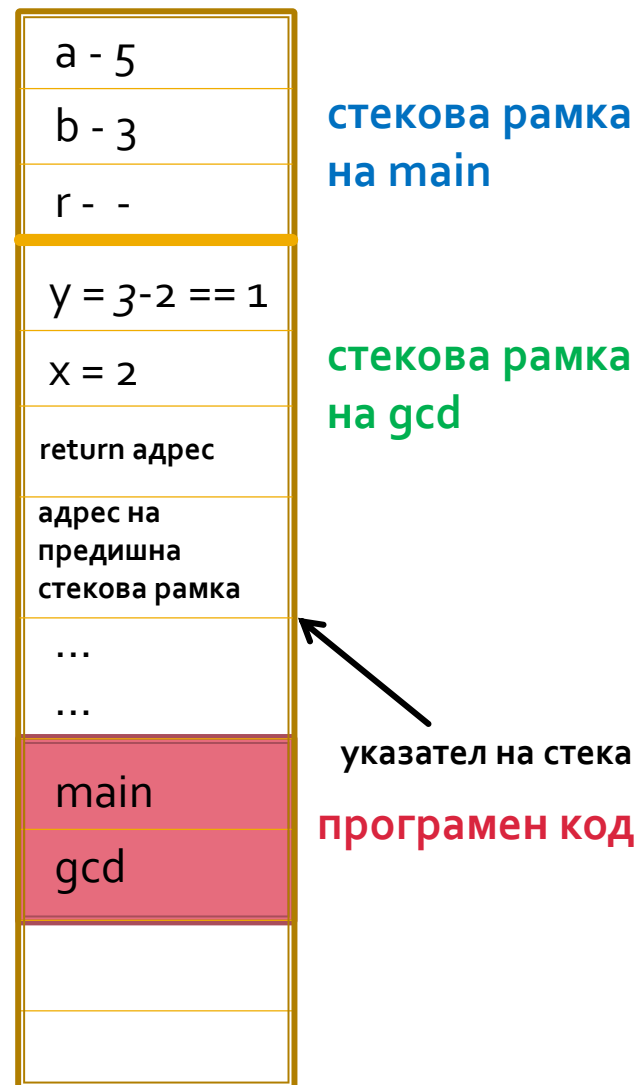
# Функции

```
int gcd(int x, int y) {
    while(x != y) {
        if (x > y) {
            x = x - y;
        } else {
            y = y - x;
        }
    }

    return x;
}

int main() {
    int a = 5, b = 3;
    int r = gcd(a, b);

    cout << r << endl;
    return 0;
}
```



# Функции

```
int gcd(int x, int y) {  
    while(x != y) {  
        if (x > y) {  
            x = x - y;  
        } else {  
            y = y - x;  
        }  
    }  
  
    return x;  
}  
  
int main() {  
    int a = 5, b = 3;  
    int r = gcd(a, b);  
  
    cout << r << endl;  
    return 0;  
}
```



# Функции

```
int gcd(int x, int y) {  
    while(x != y) {  
        if (x > y) {  
            x = x - y;  
        } else {  
            y = y - x;  
        }  
    }  
  
    return x;  
}  
  
int main() {  
    int a = 5, b = 3;  
    int r = gcd(a, b);  
  
    cout << r << endl;  
    return 0;  
}
```



# Функции

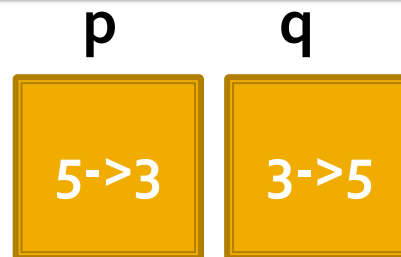
```
int gcd(int x, int y) {  
    while(x != y) {  
        if (x > y) {  
            x = x - y;  
        } else {  
            y = y - x;  
        }  
    }  
  
    return x;  
}  
  
int main() {  
    int a = 5, b = 3;  
    int r = gcd(a, b);  
  
    cout << r << endl;  
    return 0;  
}
```



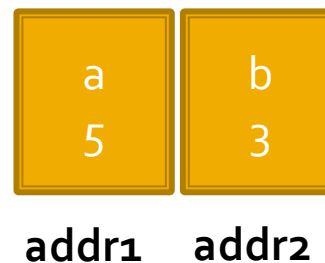


# Функции

```
void swap(int p, int q) {  
    int temp = p;  
    p = q;  
    q = temp;  
}
```



```
int main() {  
    int a = 5, b = 3;  
    swap(a, b); // 5 3  
  
    return 0;  
}
```



# Функции

```
void swap(int p, int q) {  
    int temp = p;  
    p = q;  
    q = temp;  
}
```

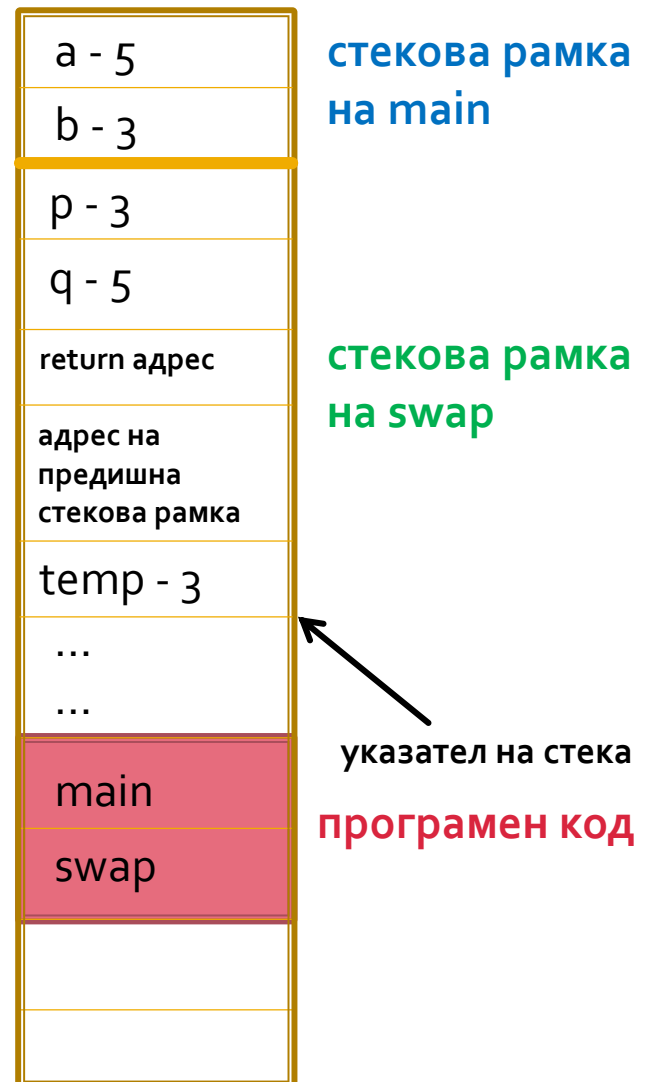
```
int main() {  
    int a = 5, b = 3;  
    swap(a, b); // 5 3  
    return 0;  
}
```



# Функции

```
void swap(int p, int q) {  
    int temp = p;  
    p = q;  
    q = temp;  
}
```

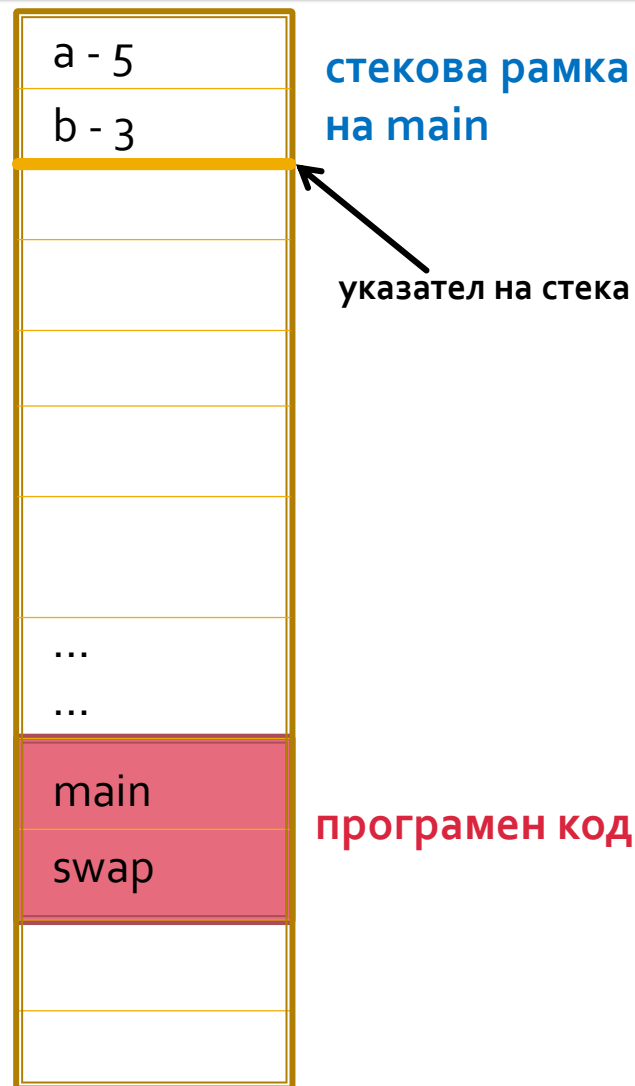
```
int main() {  
    int a = 5, b = 3;  
    swap(a, b); // 5 3  
    return 0;  
}
```



# Функции

```
void swap(int p, int q) {  
    int temp = p;  
    p = q;  
    q = temp;  
}
```

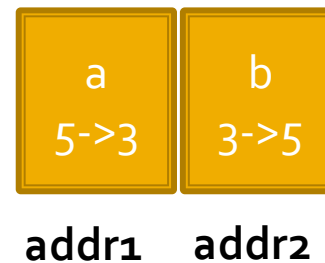
```
int main() {  
    int a = 5, b = 3;  
    swap(a, b); // 5 3  
    return 0;  
}
```



# Функции

```
void swap(int* p, int* q) {  
    int temp = *p;  
    *p = *q;  
    *q = temp;  
}
```

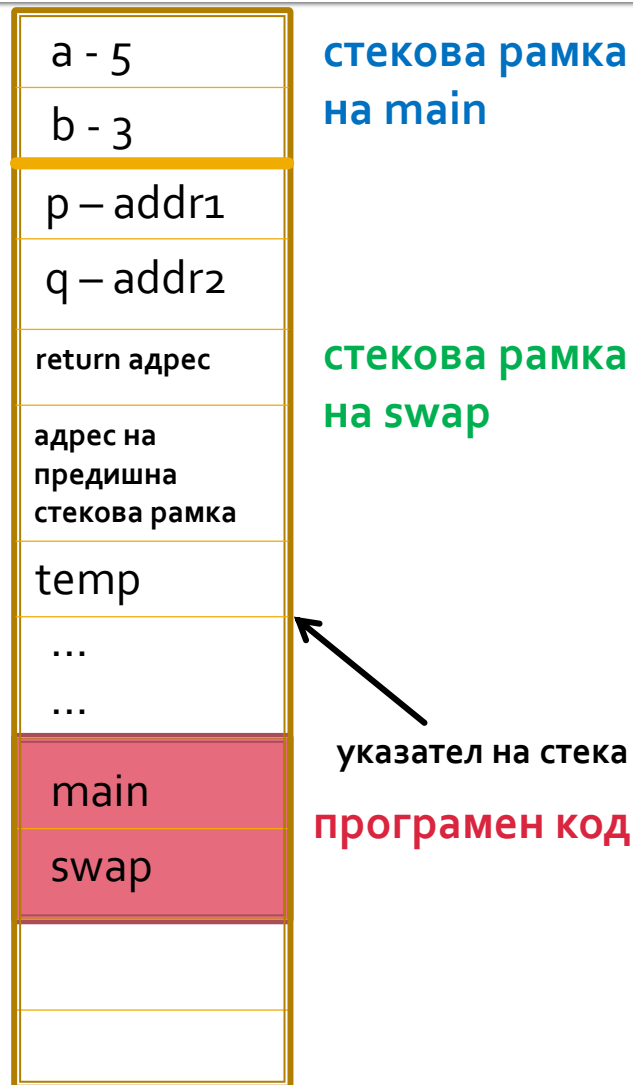
```
int main() {  
    int a = 5, b = 3;  
    swap(&a, &b); // 3 5  
    return 0;  
}
```



# Функции

```
void swap(int* p, int* q) {  
    int temp = *p;  
    *p = *q;  
    *q = temp;  
}
```

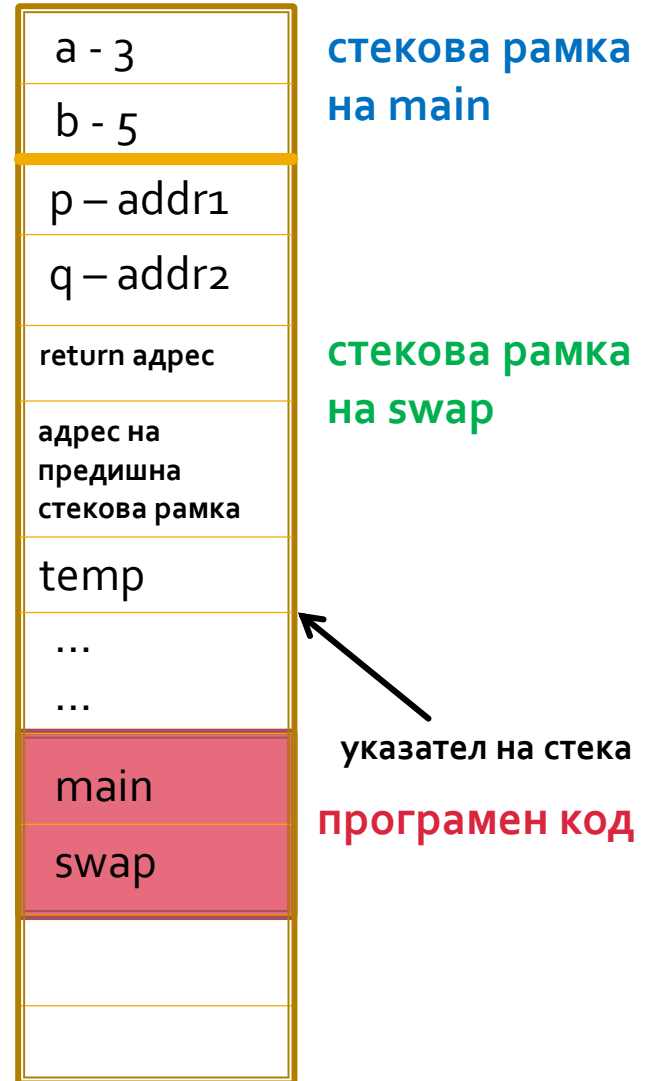
```
int main() {  
    int a = 5, b = 3;  
    swap(&a, &b); // 3 5  
    return 0;  
}
```



# Функции

```
void swap(int* p, int* q) {  
    int temp = *p;  
    *p = *q;  
    *q = temp;  
}
```

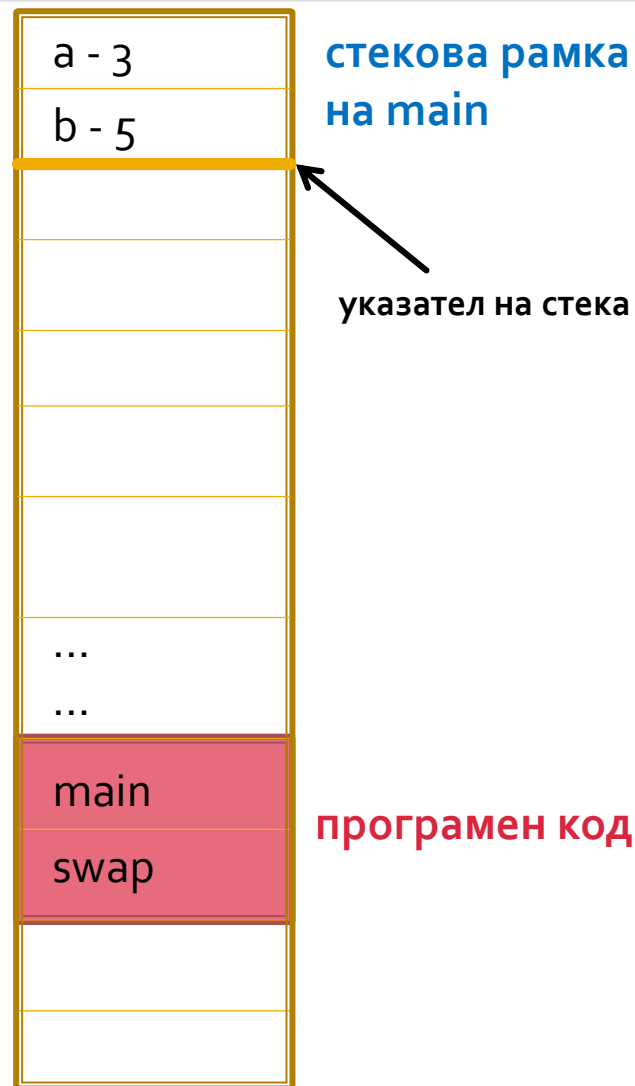
```
int main() {  
    int a = 5, b = 3;  
    swap(&a, &b); // 3 5  
    return 0;  
}
```



# Функции

```
void swap(int* p, int* q) {  
    int temp = *p;  
    *p = *q;  
    *q = temp;  
}
```

```
int main() {  
    int a = 5, b = 3;  
    swap(&a, &b); // 3 5  
    return 0;  
}
```





# Функции

Област на идентификатори:

- **Глобална** – дефинирани са пред всички функции, константи и променливи. Могат да се използват във всички функции, освен ако не е дефиниран локален идентификатор със същото име в някоя от функциите.
- **Локална** – дефинирани са във функция и не могат да се използват в други функции. Областта им започва от дефиницията и завършва с края на блока, в който идентификаторът е дефиниран.  
**Локалният идентификатор „скрива“ нелокалния в областта си.**  
Областта на формалните параметри е локална и е тялото на функцията.
- **Област на клас**

# ФУНКЦИИ

```
int sum = 0;
void testFunction() {
    sum += 1;
}

void testFunction2() {
    sum += 2;
}

int main() {
    testFunction();
    testFunction2();
    std::cout << sum << std::endl;

    return 0;
}
```

# ФУНКЦИИ

```
int sum = 0;
void testFunction() {
    sum += 1;
}

void testFunction2() {
    sum += 2;
}

int main() {
    testFunction();
    testFunction2();
    std::cout << sum << std::endl; // 3

    return 0;
}
```

# ФУНКЦИИ

Пример:

```
int sum = 0;
void testFunction() {
    int sum = 10;
    sum += 1;
}

void testFunction2() {
    sum += 2;
}

int main() {
    testFunction();
    testFunction2();
    std::cout << sum << std::endl;

    return 0;
}
```

# ФУНКЦИИ

Пример:

```
int sum = 0;
void testFunction() {
    int sum = 10;
    sum += 1;
}

void testFunction2() {
    sum += 2;
}

int main() {
    testFunction();
    testFunction2();
    std::cout << sum << std::endl;    // 2

    return 0;
}
```

# Функции

Едномерни масиви като формални параметри:

- $T \text{ arr} [ ]$  – формален параметър  $\text{arr}$  от тип едномерен масив от тип  $T$ .
- $T * \text{arr}$  – формален параметър  $\text{arr}$  от тип указател към тип  $T$ .

\* Трябва да се предаде и размерността на масива

# Функции

Да се напише функция, която въвежда елементите на масив.

```
void initArrayElements(int a[], int length) {
    for(int i = 0; i < length; i++) {
        std::cout << "a[" << i << "]=";
        std::cin >> a[i];
    }
}

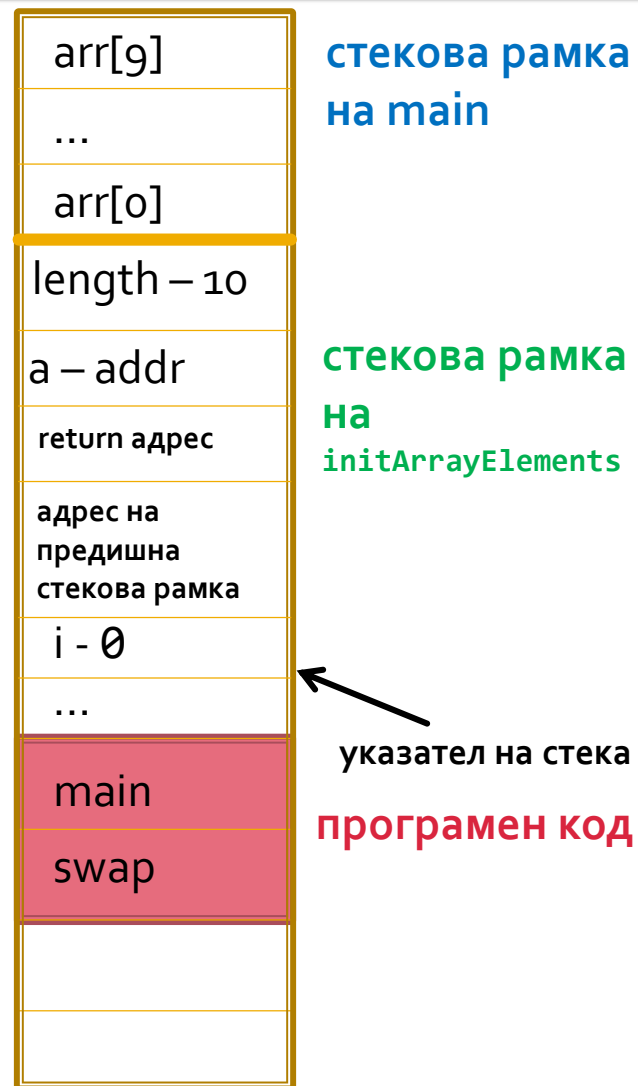
int main() {
    int arr[10];
    initArrayElements(arr, 10);

    return 0;
}
```

# Функции

```
void initArrayElements(int a[], int length) {  
    for(int i = 0; i < length; i++) {  
        std::cout << "a[" << i << "]=";  
        std::cin >> a[i];  
    }  
}
```

```
int main() {  
    int arr[10];  
    initArrayElements(arr, 10);  
  
    return 0;  
}
```

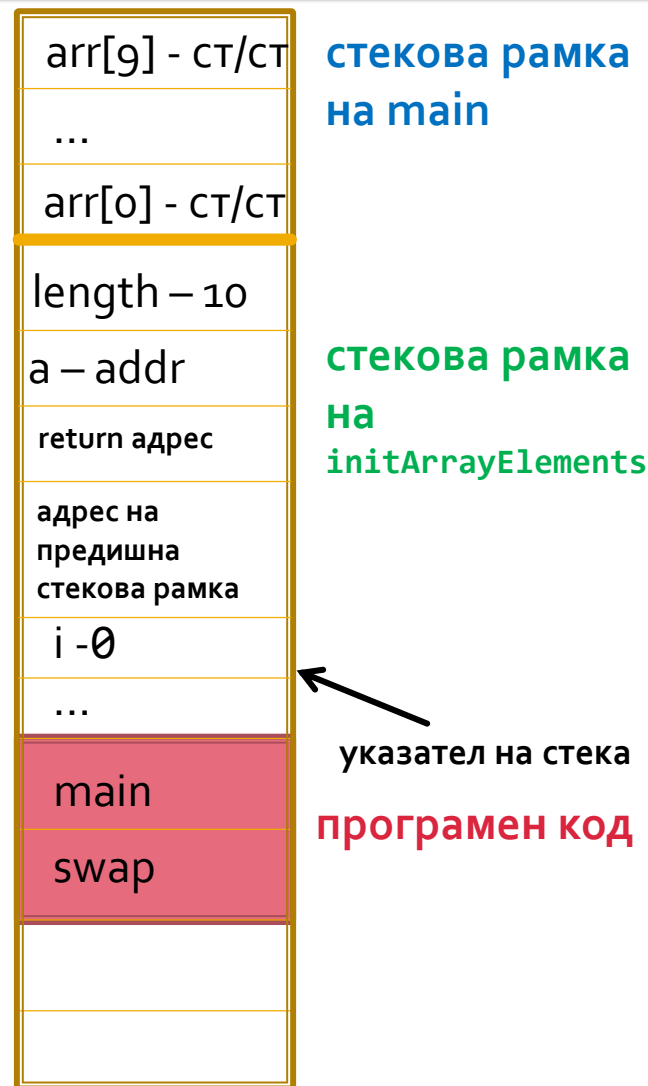




# Функции

```
void initArrayElements(int a[], int length) {  
    for(int i = 0; i < length; i++) {  
        std::cout << "a[" << i << "]=";  
        std::cin >> a[i];*(a+i)  
    }  
}
```

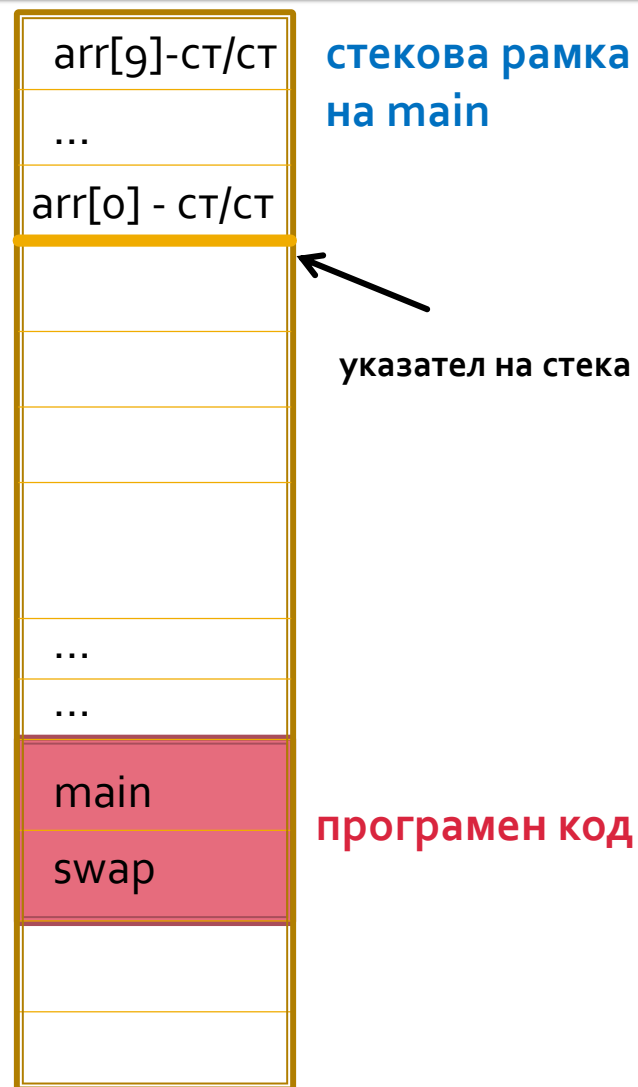
```
int main() {  
    int arr[10];  
    initArrayElements(arr, 10);  
  
    return 0;  
}
```



# Функции

```
void initArrayElements(int a[], int length) {  
    for(int i = 0; i < length; i++) {  
        std::cout << "a[" << i << "]=";  
        std::cin >> a[i/];*(a+i)  
    }  
}
```

```
int main() {  
    int arr[10];  
    initArrayElements(arr, 10);  
  
    return 0;  
}
```



Следва продължение...