Функции
(част 3)

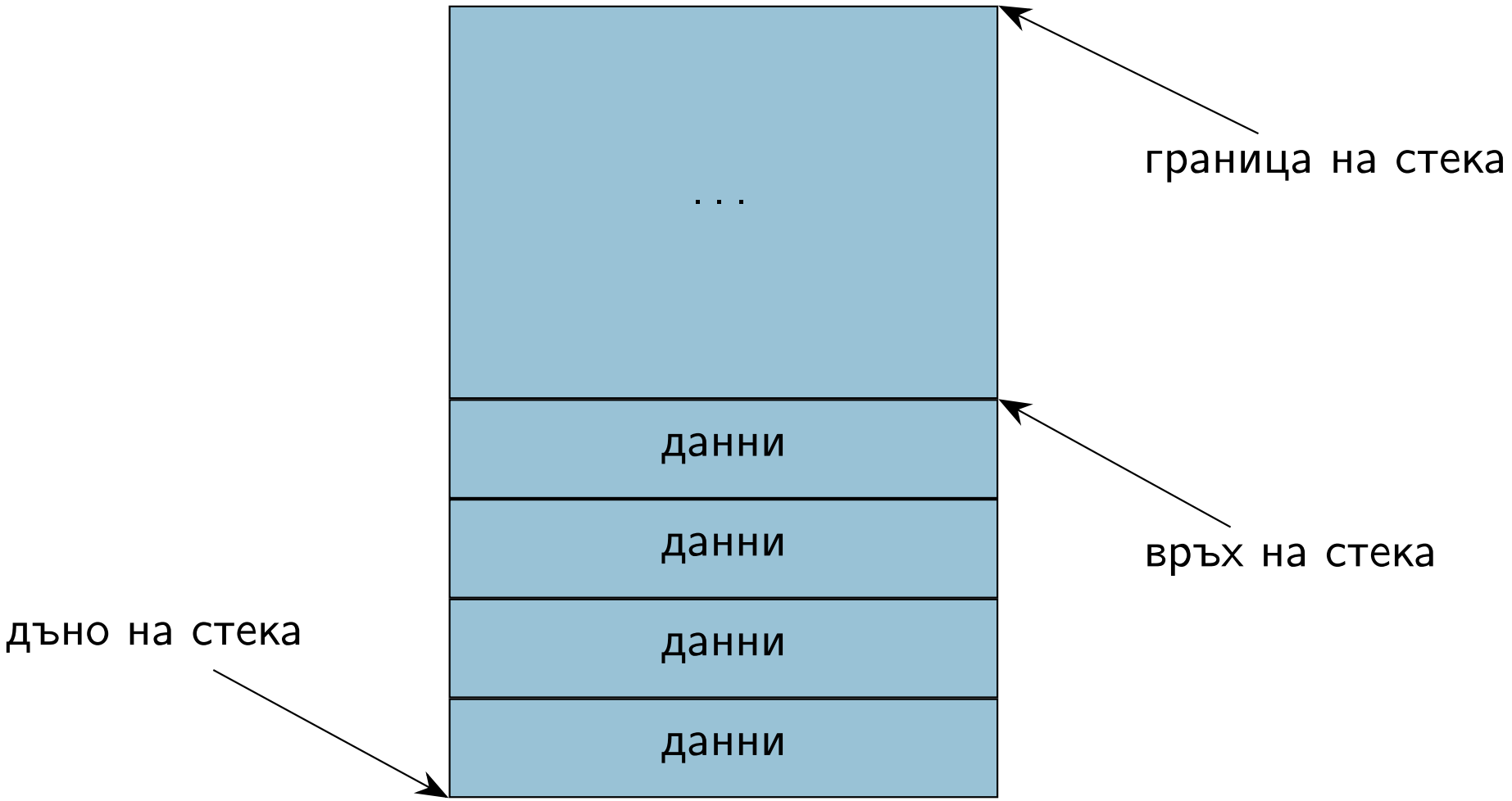
Трифон Трифонов

Увод в програмирането,
спец. Компютърни науки, 1 поток, 2018/19 г.

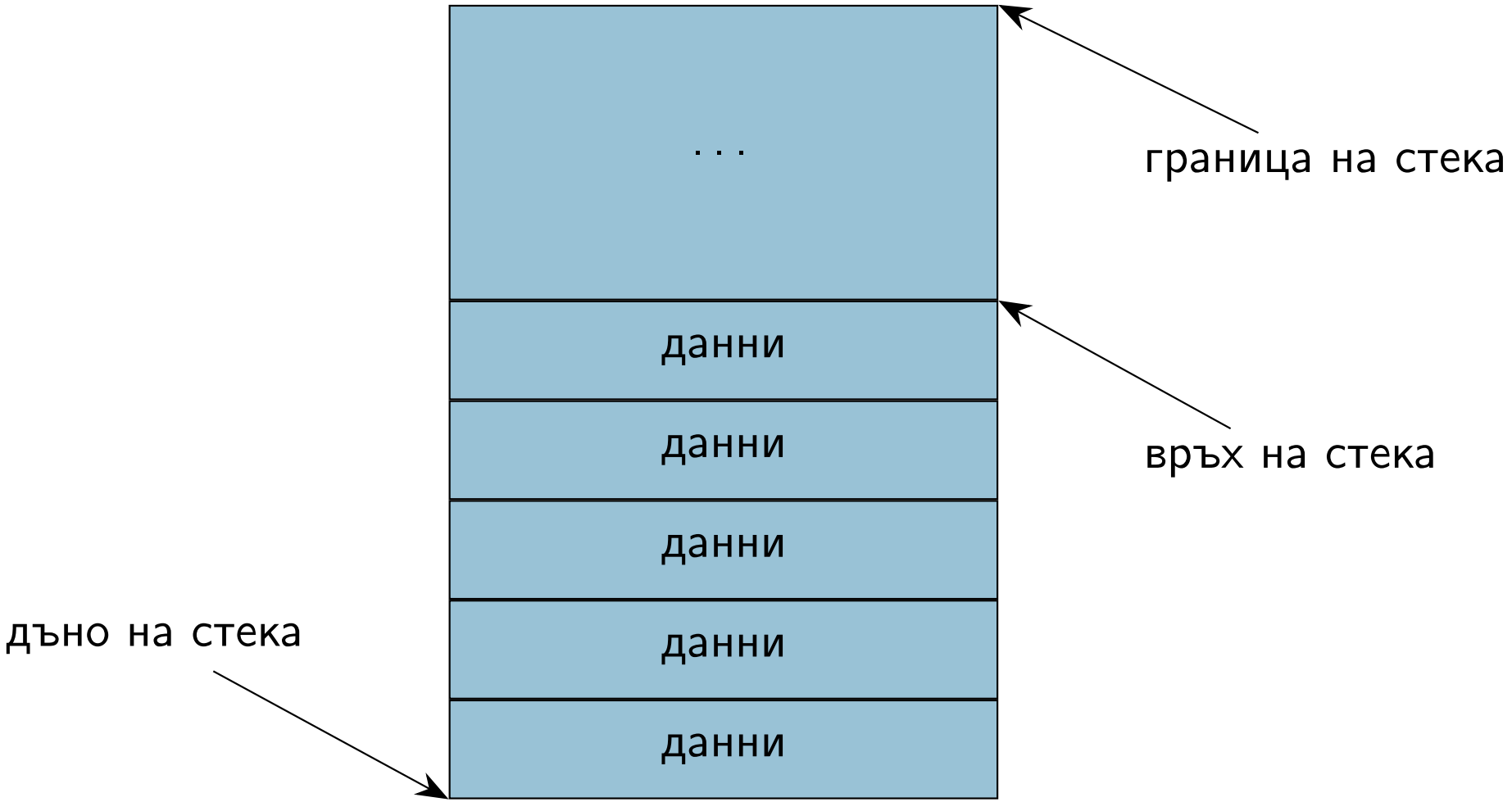
20 декември 2018 г.

Схема на програмната памет

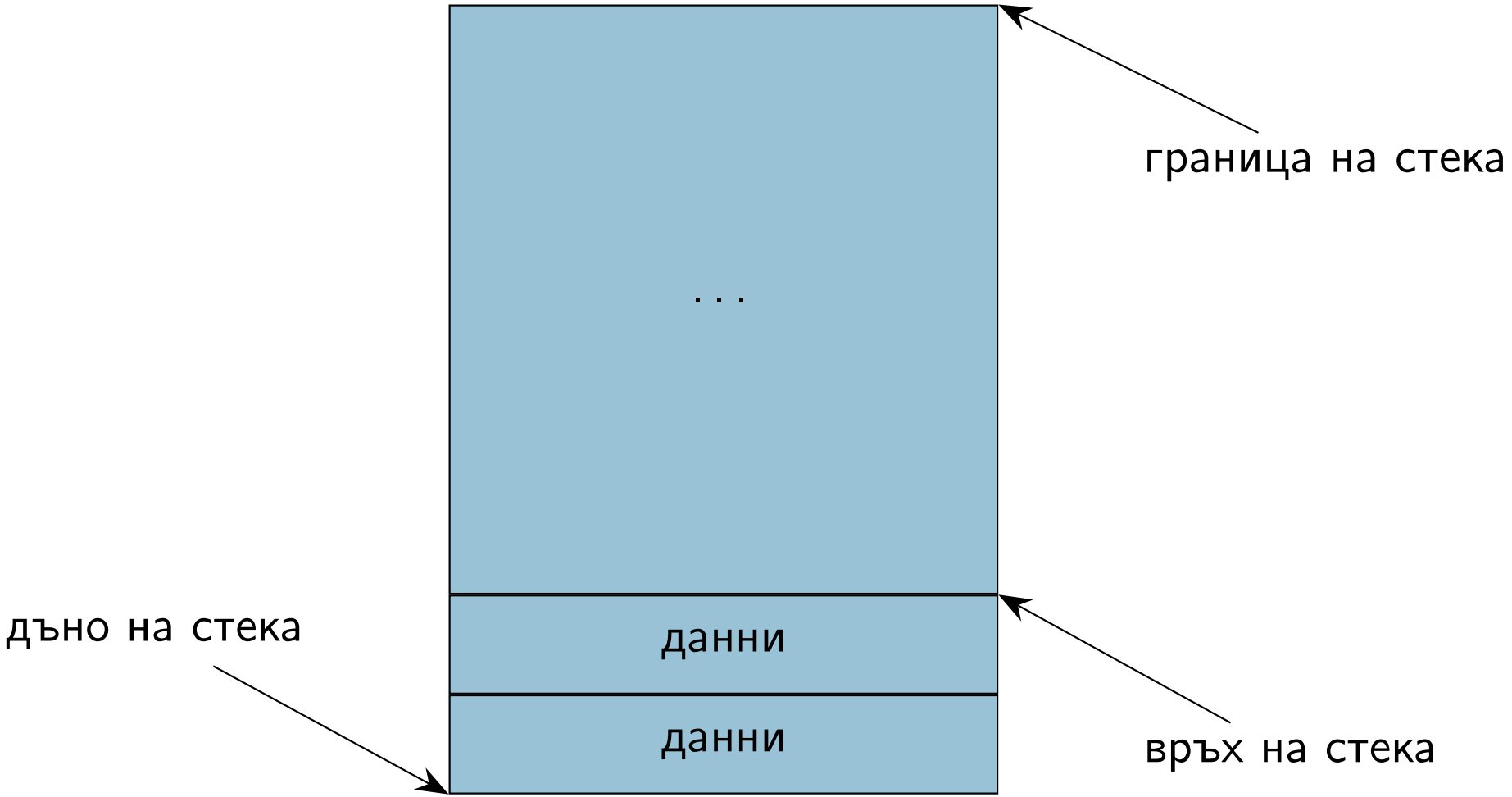
Програмен стек



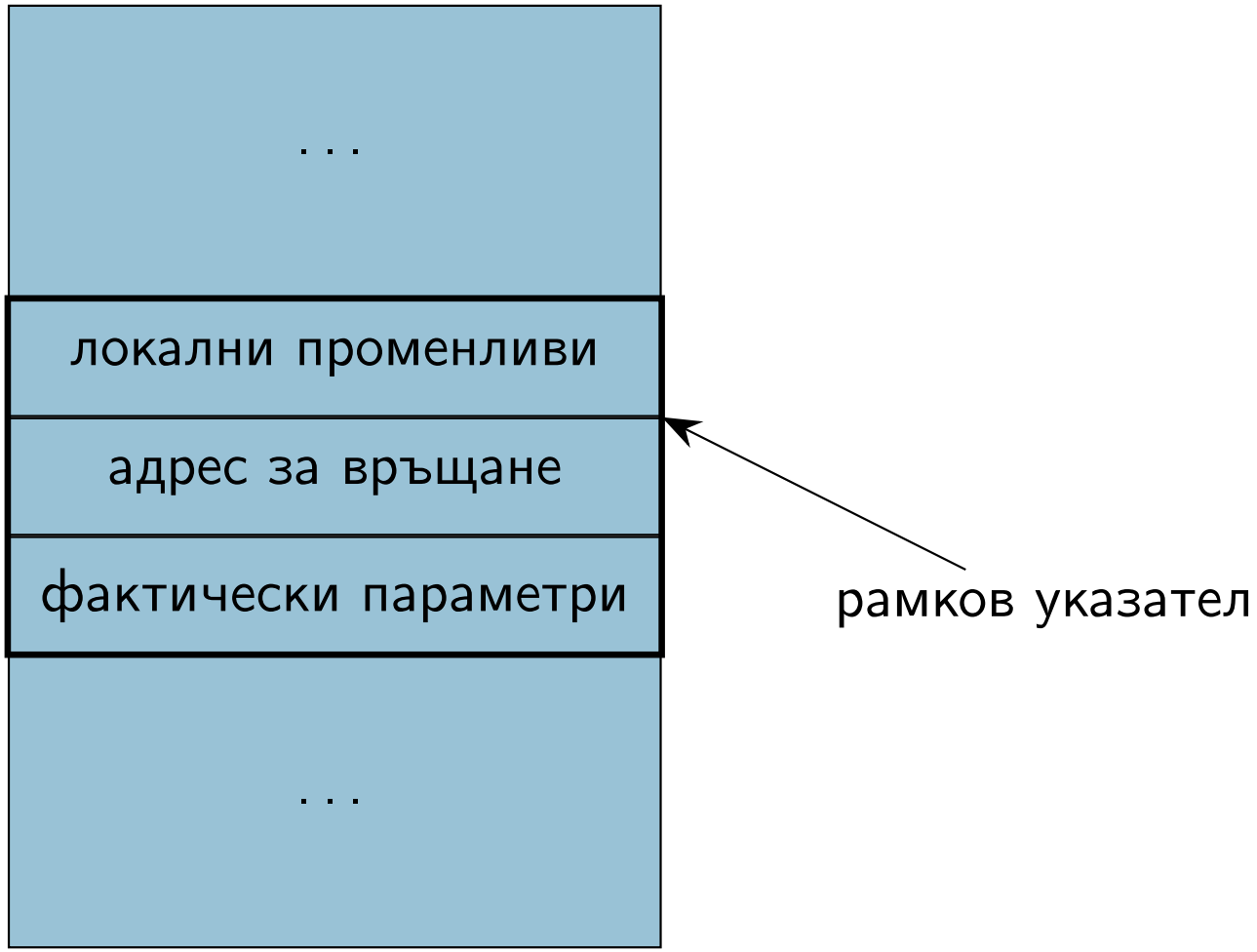
Програмен стек



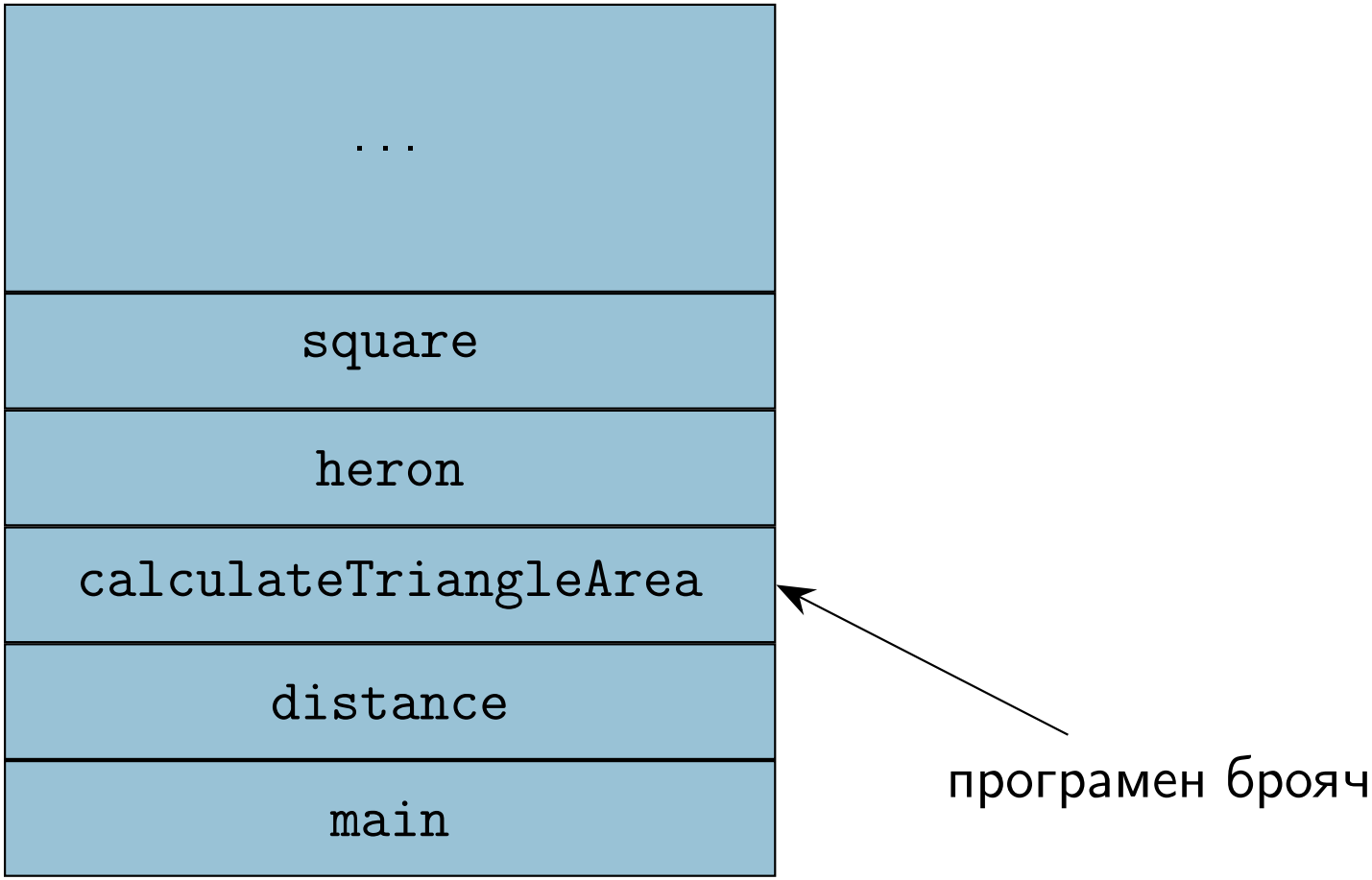
Програмен стек



Стекова рамка на функция



Област за програмен код



Предаване по стойност (call by value)

- пресмята се стойността на фактическия параметър
- в стековата рамка на функцията се създава копие на стойността
- всяка промяна на стойността остава локална за функцията
- при завършване на функцията, предадената стойност и всички промени над нея изчезват

Предаване с препратка (call by reference)

- Понякога искаме промените във **формалните** параметри да се отразят във **фактическите** параметри
- Тогава трябва да обявим, че искаме фактическите параметри да могат да бъдат променяни
- <параметър> ::= <тип>& <идентификатор>
- **Примери:**
  - `int add5(int& x) { x += 5; return x; }`
  - фактическият параметър трябва да е lvalue!
  - ~~`add5(3);`~~
  - `int a = 3; cout << add5(a) << ' ' << a;`

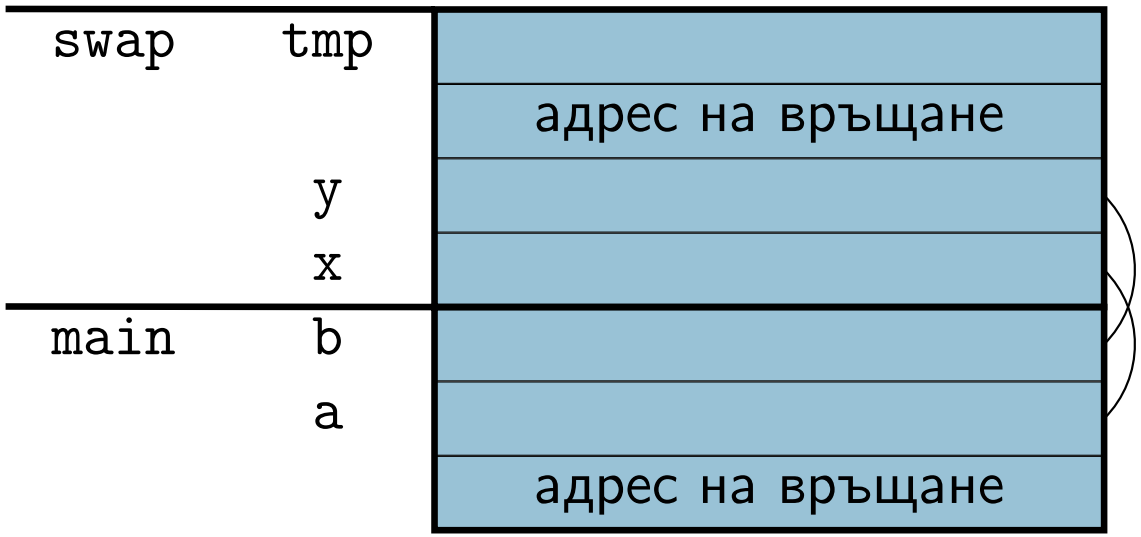
Пример за предаване с препратка

Размяна на две променливи

```
void swap(int& x, int& y) {
  int tmp = x;
  x = y;
  y = tmp;
}

int main() {
  int a = 5, b = 8;
  swap(a, b);
  cout << a << ' ' << b << endl;
}
```

Стекова рамка при предаване с препратки



Предаване по указател/адрес (call by pointer)

- Предава се адрес вместо стойност
- Фактическите параметри трябва да са от тип "указател към нещо"
- Функцията може да променя стойности на външни за функцията променливи през подадените ѝ указатели
- **Примери:**
  - `int add5(int* px) { *px += 5; return *px; }`
  - ~~`add5(3);`~~ ~~`add5(&3);`~~
  - `int a = 3; cout << add5(&a) << ' ' << a;`

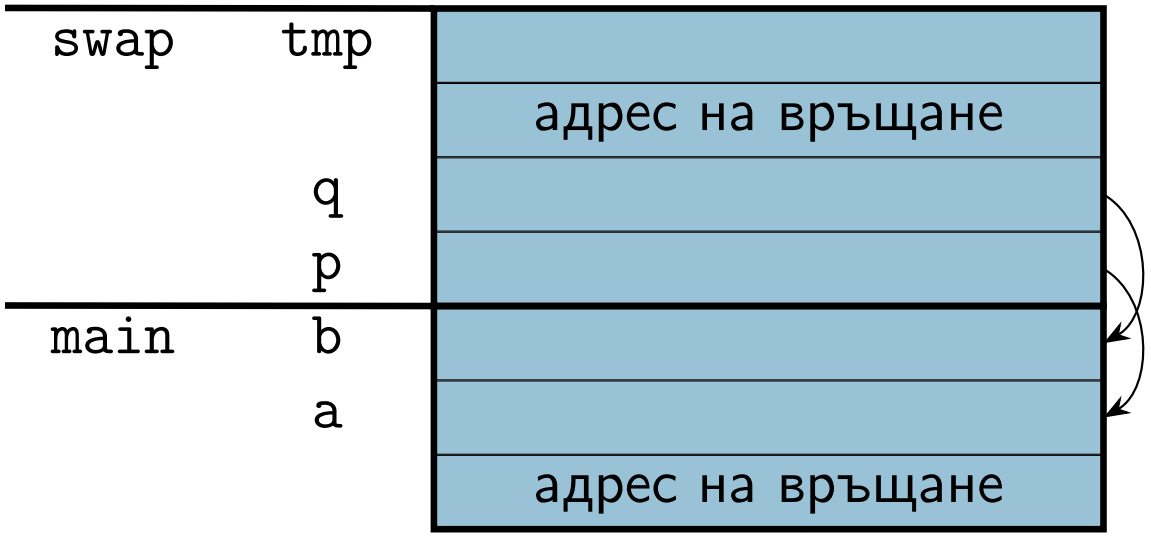
Пример за предаване по указател

Размяна на две променливи

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

int main() {
  int a = 5, b = 8;
  swap(&a, &b);
  cout << a << ' ' << b << endl;
}
```

Стекова рамка при предаване по указател



Предаване на масиви като параметри

- <параметър_масив> ::= <тип> <име> [[<константен_израз>]] |
  <тип>* <име>
- всъщност...
- ...масивите се предават по указател!
- ...затова размерът на масива в скобите се игнорира!
- ...затова промените в винаги се отразяват в оригинала!

Предаване на многомерни масиви като параметри

- <параметър_многомерен_масив> ::=
  <тип> <име> [[<константа>]]{[<константа>]} |
  <тип> (*<име>){[<константа]}
- многомерните масиви също се предават по указател
- първата размерност се игнорира
  - останалите трябва да се укажат, за да работи правилно указателната аритметика
- (поне) първата размерност трябва да се подава като параметър
- Внимание: `int* a[10]` е различно от `int (*a)[10]`!
  - `int* a[10]` $\Longleftrightarrow$ масив от 10 указателя към цели числа
  - `int (*a)[10]` $\Longleftrightarrow$ указател към масив от десет цели числа
  - ...но понеже масивите от тип T могат да се разглеждат като указатели към тип T...
  - `int (*a)[10]` $\Longleftrightarrow$ масив от масив от десет цели числа
  - `int (*a)[10]` $\Longleftrightarrow$ двумерен масив от цели числа с 10 колони

Примерни функции

1. Да се напише функция, която извежда матрица от числа
2. Да се напише функция, която въвежда масив от низове
3. Да се напише функция, която проверява дали дадена дума се съдържа в масив от низове
4. Да се напише функция, която умножава две правоъгълни матрици