Функции
(част 2)

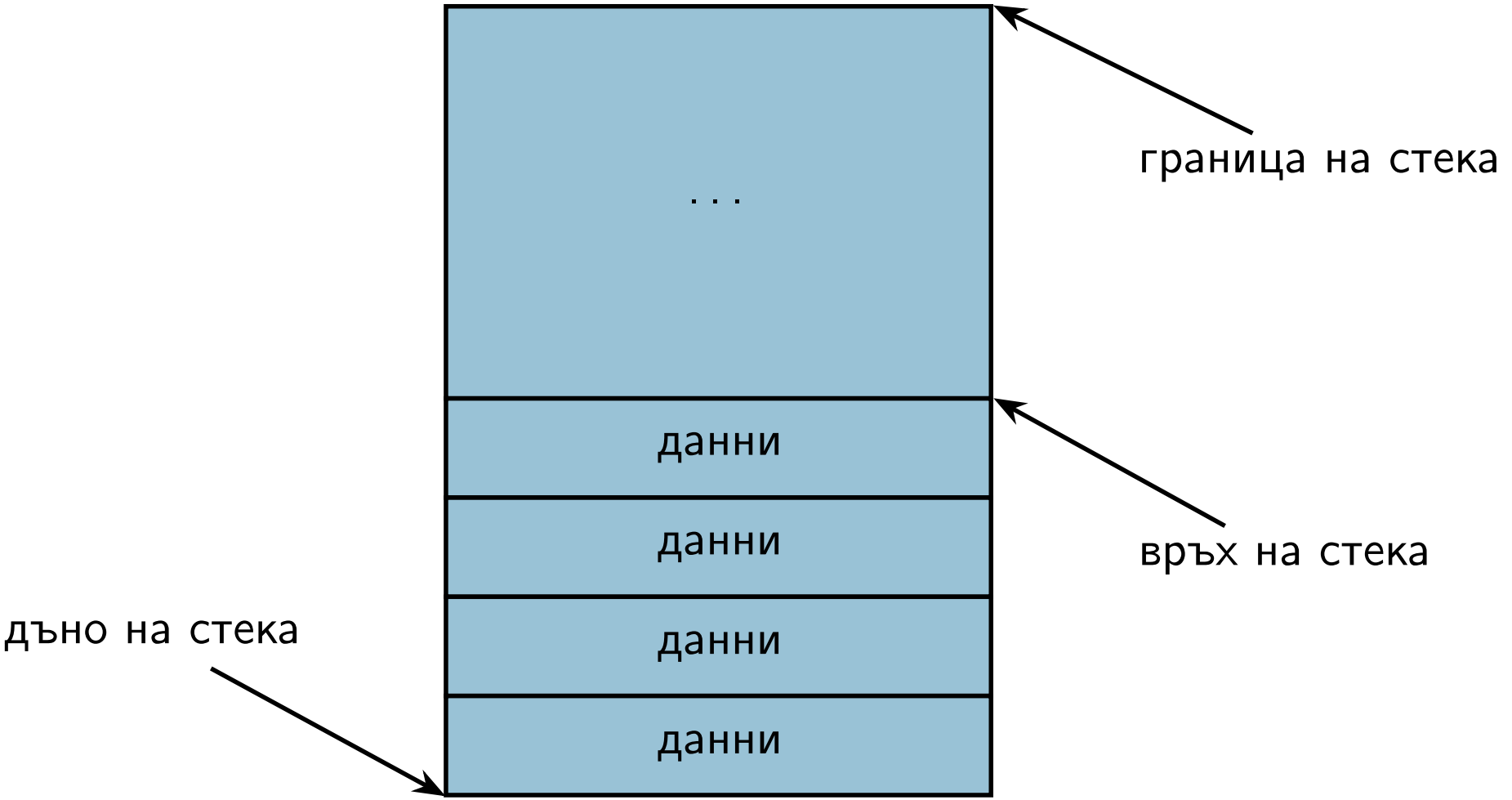
Трифон Трифонов

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

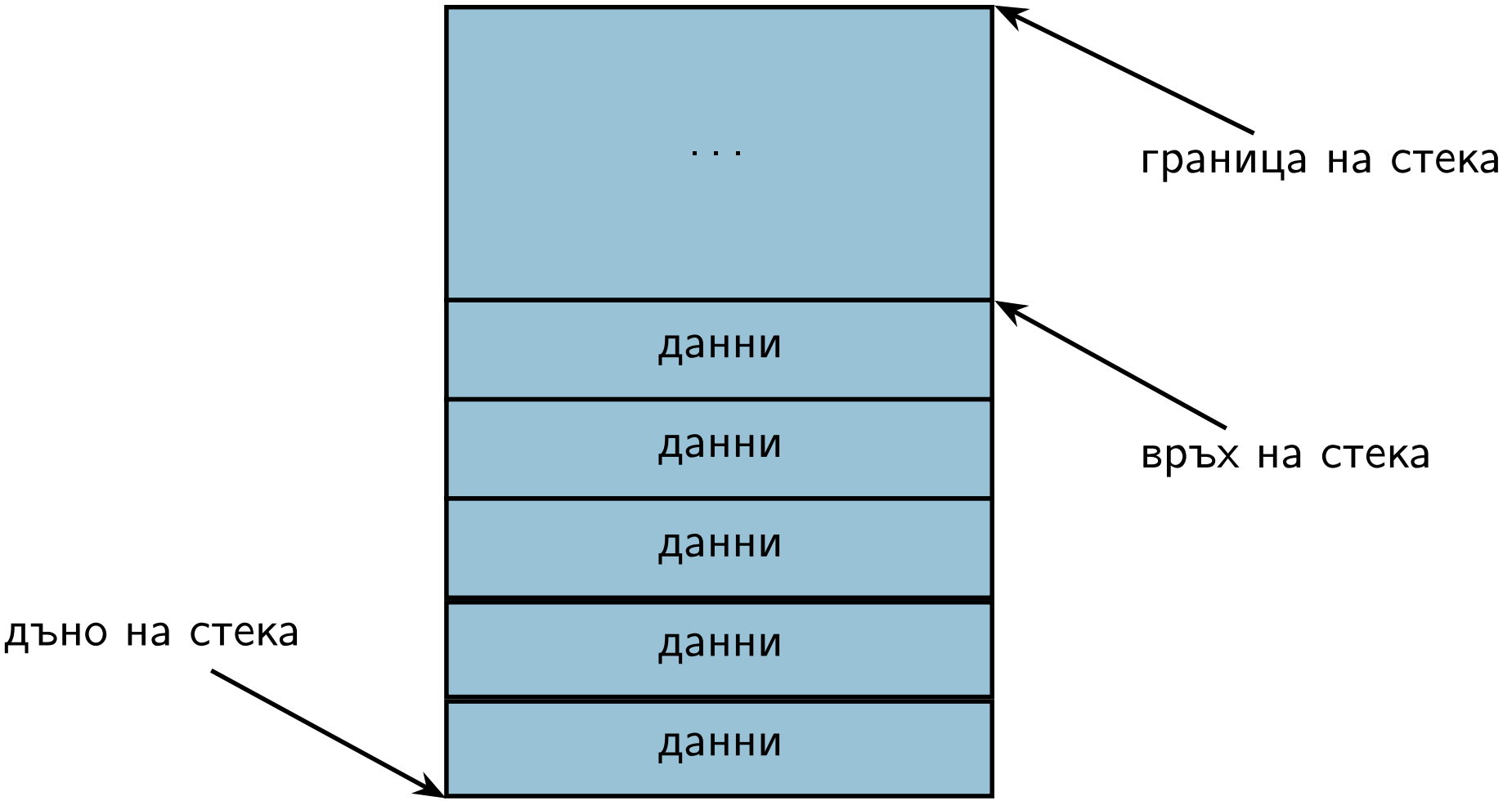
13 декември 2018 г.

Схема на програмната памет

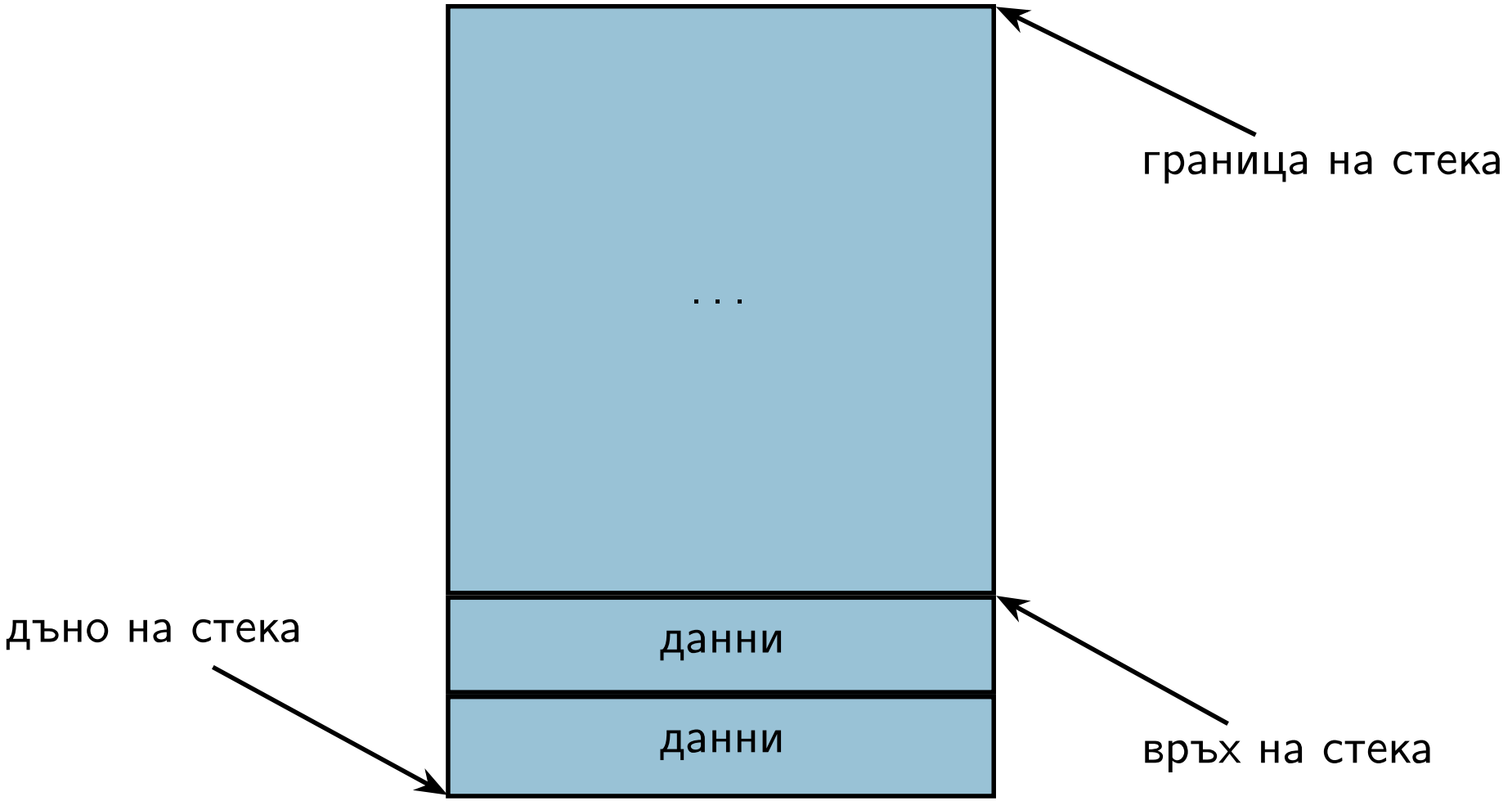
Програмен стек



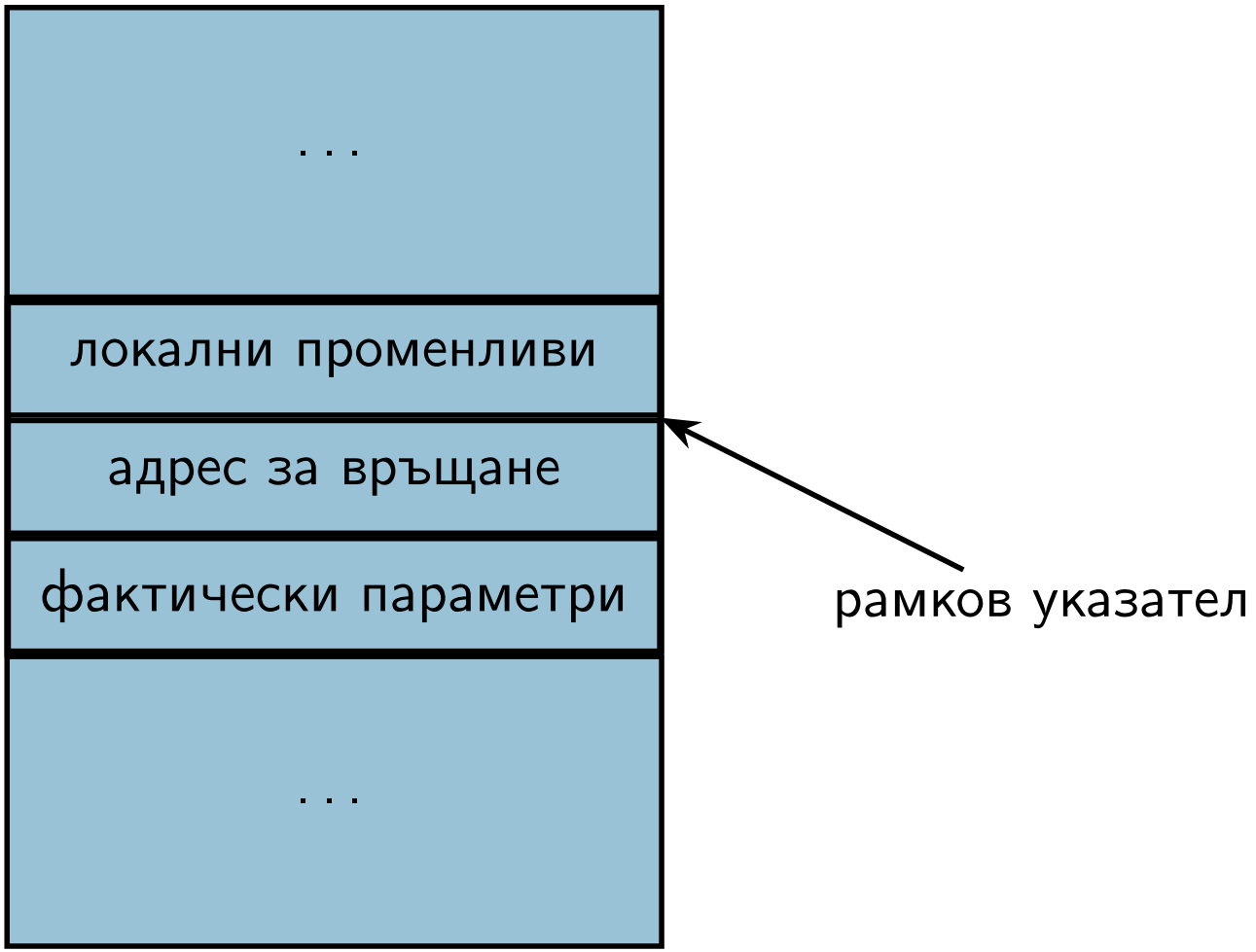
Програмен стек



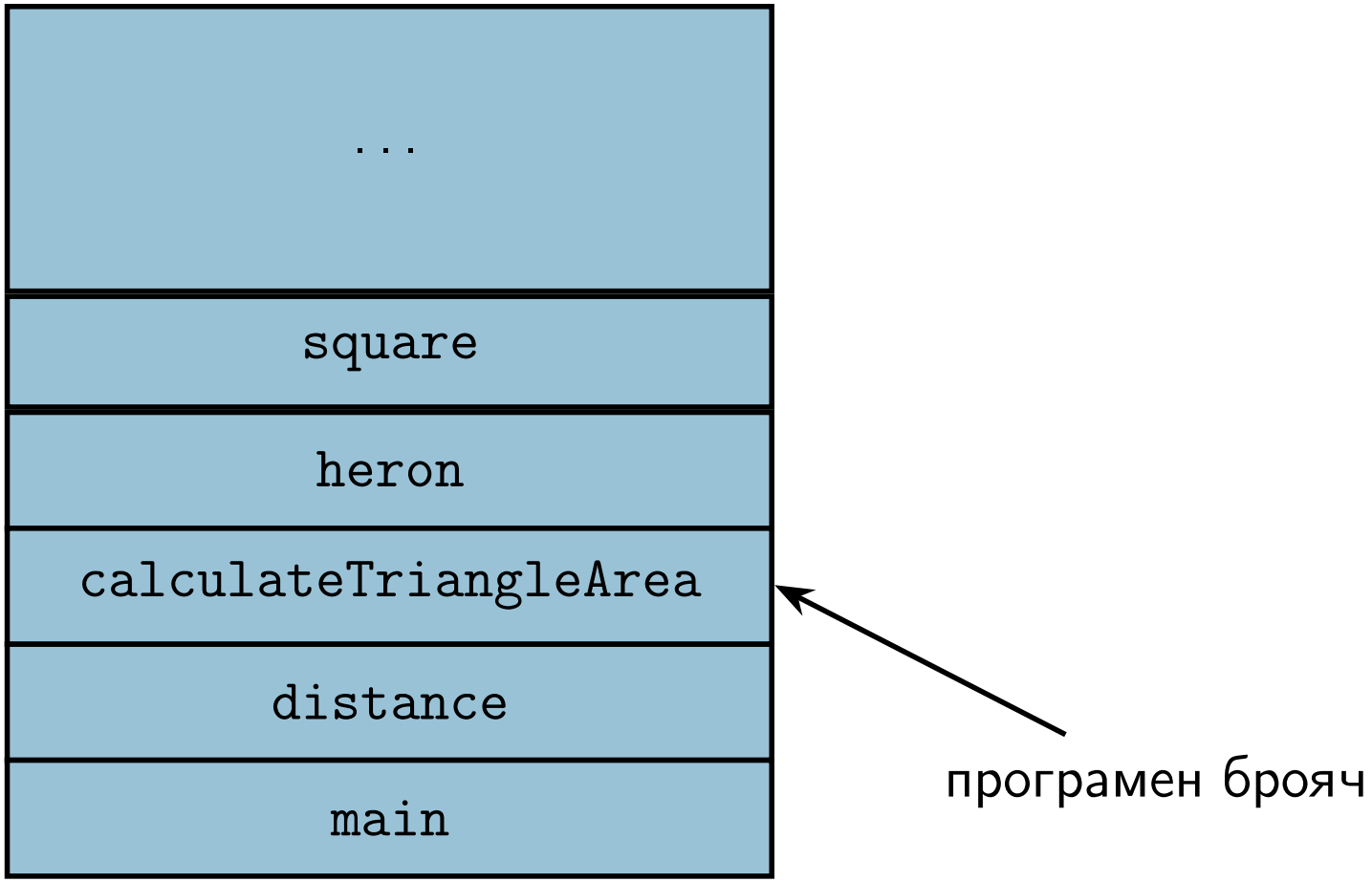
Програмен стек



Стекова рамка на функция



Област за програмен код

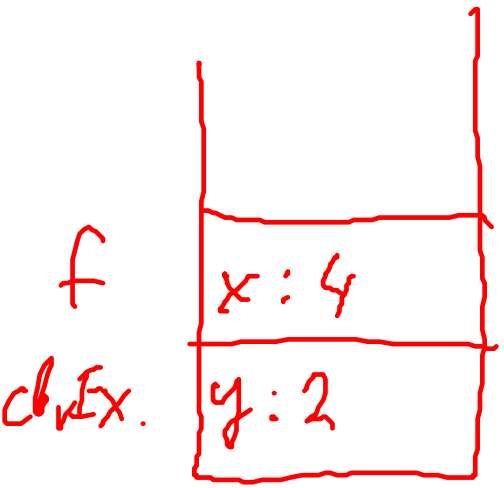


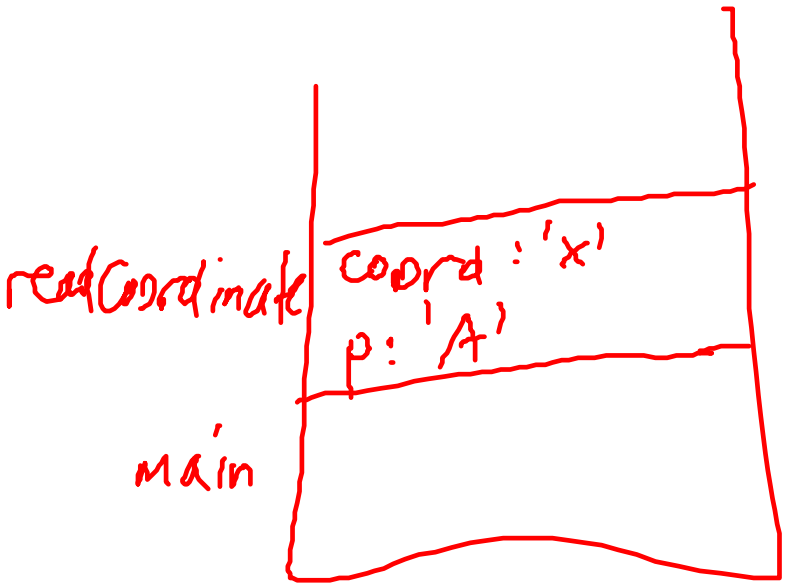
Предаване по стойност (call by value)

- пресмята се стойността на фактическия параметър

Предаване по стойност (call by value)

- пресмята се стойността на фактическия параметър
- в стековата рамка на функцията се създава копие на стойността





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

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

Странични ефекти

Какви странични ефекти може да има функция на C++?

Странични ефекти

Какви странични ефекти може да има функция на C++?

- Използване на глобални променливи

Странични ефекти

Какви странични ефекти може да има функция на C++?

- Използване на глобални променливи
- Използване на статични променливи
  `static` <дефиниция_на_променлива>

Странични ефекти

Какви странични ефекти може да има функция на C++?

- Използване на глобални променливи
- Използване на статични променливи
  `static` <дефиниция_на_променлива>
- Работа с вход или изход

Предаване с препратка (call by reference)

- Понякога искаме промените във **формалните** параметри да се отразят във **фактическите** параметри

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

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>

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:**

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; }`

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!

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);`~~

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

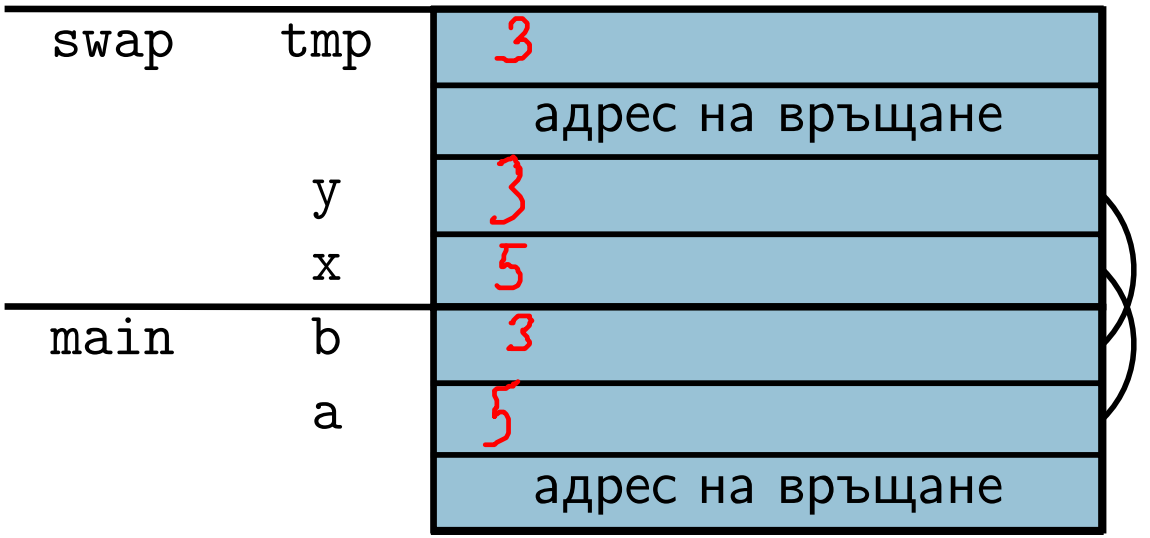
```
void swap(int& x, int& y) {
  int tmp = x;
  x = y;
  y = tmp;
}
```

Пример за предаване с препратка

Размяна на две променливи

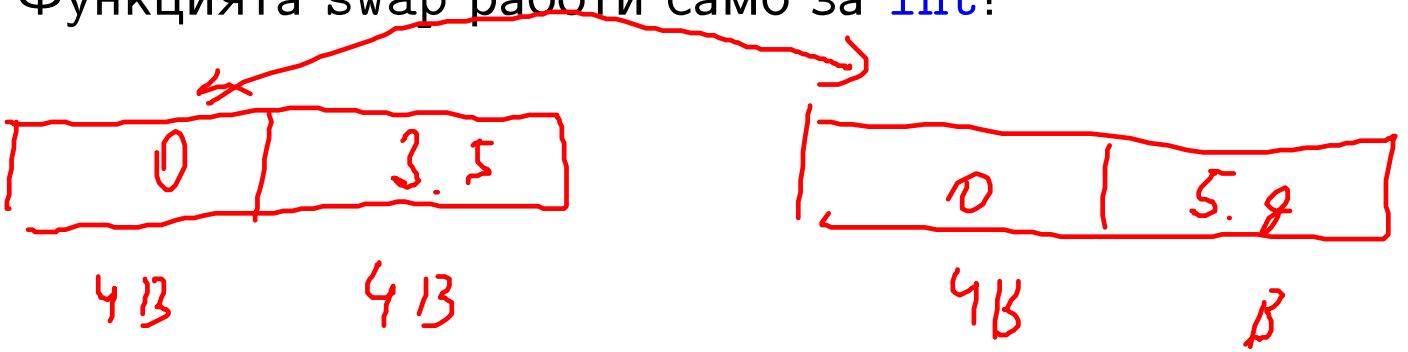
```
void swap(int& x, int& y) {
  int tmp = x;
  x = y;
  y = tmp;
}

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

Стекова рамка при предаване с препратка



Претоварване на функции (overloading)

- **Проблем:** Функцията `swap` работи само за `int`!



Претоварване на функции (overloading)

- **Проблем:** Функцията `swap` работи само за `int`!
- Ако искаме функция за размяна на `double` променливи, трябва да направим нова функция `swap_double(double&, double&);`

Претоварване на функции (overloading)

- **Проблем:** Функцията `swap` работи само за `int`!
- Ако искаме функция за размяна на `double` променливи, трябва да направим нова функция `swap_double(double&, double&);`
- Аналогично за `char`, `long`, ...

Претоварване на функции (overloading)

- **Проблем:** Функцията `swap` работи само за `int`!
- Ако искаме функция за размяна на `double` променливи, трябва да направим нова функция `swap_double(double&, double&);`
- Аналогично за `char`, `long`, ...
- Трябва ли да си измисляме нови имена за всяка от тези функции, които всъщност прави едно и също?

Претоварване на функции (overloading)

- **Проблем:** Функцията `swap` работи само за `int`!
- Ако искаме функция за размяна на `double` променливи, трябва да направим нова функция `swap_double(double&, double&);`
- Аналогично за `char`, `long`, ...
- Трябва ли да си измисляме нови имена за всяка от тези функции, които всъщност прави едно и също?
- Не! Можем да използваме едно и също име за няколко функции!

Претоварване на функции (overloading)

- **Проблем:** Функцията `swap` работи само за `int`!
- Ако искаме функция за размяна на `double` променливи, трябва да направим нова функция `swap_double(double&, double&);`
- Аналогично за `char`, `long`, ...
- Трябва ли да си измисляме нови имена за всяка от тези функции, които всъщност прави едно и също?
- Не! Можем да използваме едно и също име за няколко функции!
- Сигнатурата на функцията зависи от:
  - типа на връщане
  - типа и реда на параметрите

Претоварване на функции (overloading)

- **Проблем:** Функцията `swap` работи само за `int`!
- Ако искаме функция за размяна на `double` променливи, трябва да направим нова функция `swap_double(double&, double&);`
- Аналогично за `char`, `long`, ...
- Трябва ли да си измисляме нови имена за всяка от тези функции, които всъщност прави едно и също?
- Не! Можем да използваме едно и също име за няколко функции!
- Сигнатурата на функцията зависи от:
  - типа на връщане
  - типа и реда на параметрите
- Функции с еднакво име и различна сигнатура са различни

Претоварване на функции (overloading)

- **Проблем:** Функцията `swap` работи само за `int`!
- Ако искаме функция за размяна на `double` променливи, трябва да направим нова функция `swap_double(double&, double&);`
- Аналогично за `char`, `long`, ...
- Трябва ли да си измисляме нови имена за всяка от тези функции, които всъщност прави едно и също?
- Не! Можем да използваме едно и също име за няколко функции!
- Сигнатурата на функцията зависи от:
  - типа на връщане
  - типа и реда на параметрите
- Функции с еднакво име и различна сигнатура са различни
- Казваме, че името е претоварено (overloaded)

Претоварване на функции (overloading)

- **Проблем:** Функцията `swap` работи само за `int`!
- Ако искаме функция за размяна на `double` променливи, трябва да направим нова функция `swap_double(double&, double&);`
- Аналогично за `char`, `long`, ...
- Трябва ли да си измисляме нови имена за всяка от тези функции, които всъщност прави едно и също?
- Не! Можем да използваме едно и също име за няколко функции!
- Сигнатурата на функцията зависи от:
  - типа на връщане
  - типа и реда на параметрите
- Функции с еднакво име и различна сигнатура са различни
- Казваме, че името е претоварено (overloaded)
- Проблем: може да възникне неоднозначност при извикването!

Предаване на масиви като параметри

- <параметър_масив> ::= <тип> <име> [[<константен_израз>]] |
  <тип>* <име>

Предаване на масиви като параметри

- <параметър_масив> ::= <тип> <име> [[<константен_израз>]] |
  <тип>* <име>
- размерът на масива се игнорира!

Предаване на масиви като параметри

- <параметър_масив> ::= <тип> <име> [[<константен_израз>]] |
  <тип>* <име>
- размерът на масива се игнорира!
  - затова обикновено се подава като допълнителен параметър

Предаване на масиви като параметри

- <параметър_масив> ::= <тип> <име> [[<константен_израз>]] |
  <тип>* <име>
- размерът на масива се игнорира!
  - затова обикновено се подава като допълнителен параметър
- промените в масива винаги се отразяват в оригинала

Предаване на масиви като параметри

- <параметър_масив> ::= <тип> <име> [[<константен_израз>]] |
  <тип>* <име>
- размерът на масива се игнорира!
  - затова обикновено се подава като допълнителен параметър
- промените в масива винаги се отразяват в оригинала
- **Примери:**

Предаване на масиви като параметри

- <параметър_масив> ::= <тип> <име> [[<константен_израз>]] |
  <тип>* <име>
- размерът на масива се игнорира!
  - затова обикновено се подава като допълнителен параметър
- промените в масива винаги се отразяват в оригинала
- **Примери:**
  - `int readArray(int a[]);`

Предаване на масиви като параметри

- <параметър_масив> ::= <тип> <име> [[<константен_израз>]] |
  <тип>* <име>
- размерът на масива се игнорира!
  - затова обикновено се подава като допълнителен параметър
- промените в масива винаги се отразяват в оригинала
- **Примери:**
  - `int readArray(int a[]);`
  - `void printArray(int* a, int n);`

Примерни функции

1. Да се напише функция, която въвежда масив

Примерни функции

1. Да се напише функция, която въвежда масив
2. Да се напише функция, която извежда масив

Примерни функции

1. Да се напише функция, която въвежда масив
2. Да се напише функция, която извежда масив
3. Да се напише функция, която търси елемент в масив

Примерни функции

1. Да се напише функция, която въвежда масив
2. Да се напише функция, която извежда масив
3. Да се напише функция, която търси елемент в масив
4. Да се напише функция, която проверява дали два низа са равни

Примерни функции

1. Да се напише функция, която въвежда масив
2. Да се напише функция, която извежда масив
3. Да се напише функция, която търси елемент в масив
4. Да се напише функция, която проверява дали два низа са равни
5. Да се напише функция, която намира най-малкия и най-големия елемент на масив

Връщане на няколко резултата

- Как да върнем едновременно минимума и максимума?

Връщане на няколко резултата

- Как да върнем едновременно минимума и максимума?
- **Идея:** да използваме параметрите за резултат!

Връщане на няколко резултата

- Как да върнем едновременно минимума и максимума?
- **Идея:** да използваме параметрите за резултат!
- Можем да запишем резултат в параметър, предаден с препратка

Връщане на няколко резултата

- Как да върнем едновременно минимума и максимума?
- **Идея:** да използваме параметрите за резултат!
- Можем да запишем резултат в параметър, предаден с препратка
- `void findMinMax(int a[], int n, int& min, int& max);`