

# Функции

(часть 1)

# Функциите в математиката

- Какво е функция в математиката?
- $f : \text{Dom} \rightarrow \text{Ran}$
- Недефинираност на функция
- $\text{Dom}$  – дефиниционна област
  - $\mathbb{N}$ ,  $\{ 1, 2, 3, 4 \}$ ,  $[1;100]$
- $\text{Ran}$  – област от стойности
  - $\mathbb{N}$ ,  $\{\text{false}, \text{true}\}$ ,  $\{ 5, 10, 15, \dots \}$

# Какво е програмна функция?

- Относително независима част от програмата, извършваща дадено пресмятане
- Може да бъде използвана многократно
- Пример 1: `double square(double);`
- Пример 2: `double distance(double, double, double, double);`
- Не съвсем същото като в математиката!

# Какво е подпрограма?

- Относително независима част от програмата, извършваща определена последователност от действия
- Може да бъде изпълнявана многократно
- Още: процедура, метод
- Пример 1: `void printHello();`
- Пример 2: `void printArray(int a[], int n);`

# Процедури и функции

- Функцията извършва пресмятане и връща резултат
- Процедурата изпълнява поредица от оператори
- Понякога двете понятия се смесват...
- Пример: `int readArray(int a[]);`
- В C++ се наричат просто “функции”



# Предимства на функциите

- Разбиване на една сложна задача на няколко по-лесни
- Избягване на повторение на код
- Възможност за преизползване на код
- Разпределение на работата
- Абстрахиране от реализацията
- Улеснява се поддръжката и тестването

# Дефиниране на функция

- $\langle \text{сигнатура} \rangle \{ \langle \text{тяло} \rangle \}$
- $\langle \text{сигнатура} \rangle ::= [ \langle \text{тип} \rangle \mid \text{void} ]$   
 $\langle \text{идентификатор} \rangle ( \langle \text{формални\_параметри} \rangle )$
- $\langle \text{параметри} \rangle ::= \langle \text{празно} \rangle \mid \text{void} \mid$   
 $\langle \text{параметър} \rangle \{ , \langle \text{параметър} \rangle \}$
- $\langle \text{параметър} \rangle ::= \langle \text{тип} \rangle [ \langle \text{идентификатор} \rangle ]$
- $\langle \text{тяло} \rangle ::= \{ \langle \text{оператор} \rangle \}$

# Извикване на функция

- $\langle \text{име} \rangle (\langle \text{фактически\_параметри} \rangle)$
- $\langle \text{фактически\_параметри} \rangle ::= \langle \text{празно} \rangle \mid \text{void} \mid \langle \text{израз} \rangle \{, \langle \text{израз} \rangle \}$
- извикването на функция е операция с много висок приоритет
- типът на израза съвпада с типа на функцията



# Връщане на резултат

- **return** [<израз>;
- <израз> е от типа на функцията
- работата на функцията се прекратява незабавно (като цикъл при break)
- стойността на <израз> се замества на мястото на извикване на функцията

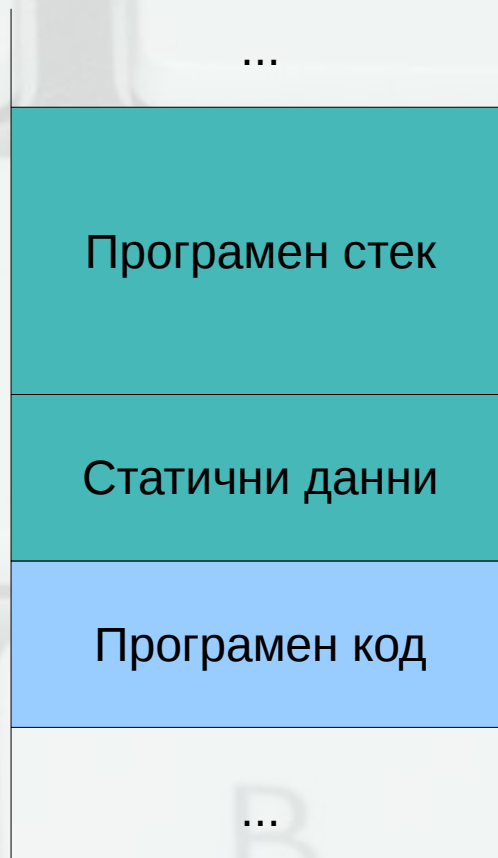
# Деклариране на функция

- <сигнатура>;
- Обещание за дефиниция на функция
- Декларацията не е задължителна
- Може да има много декларации на една и съща функция
- Но може да има само една дефиниция
- Неизпълнените обещания водят до грешка
  - освен когато не се разчита на тях

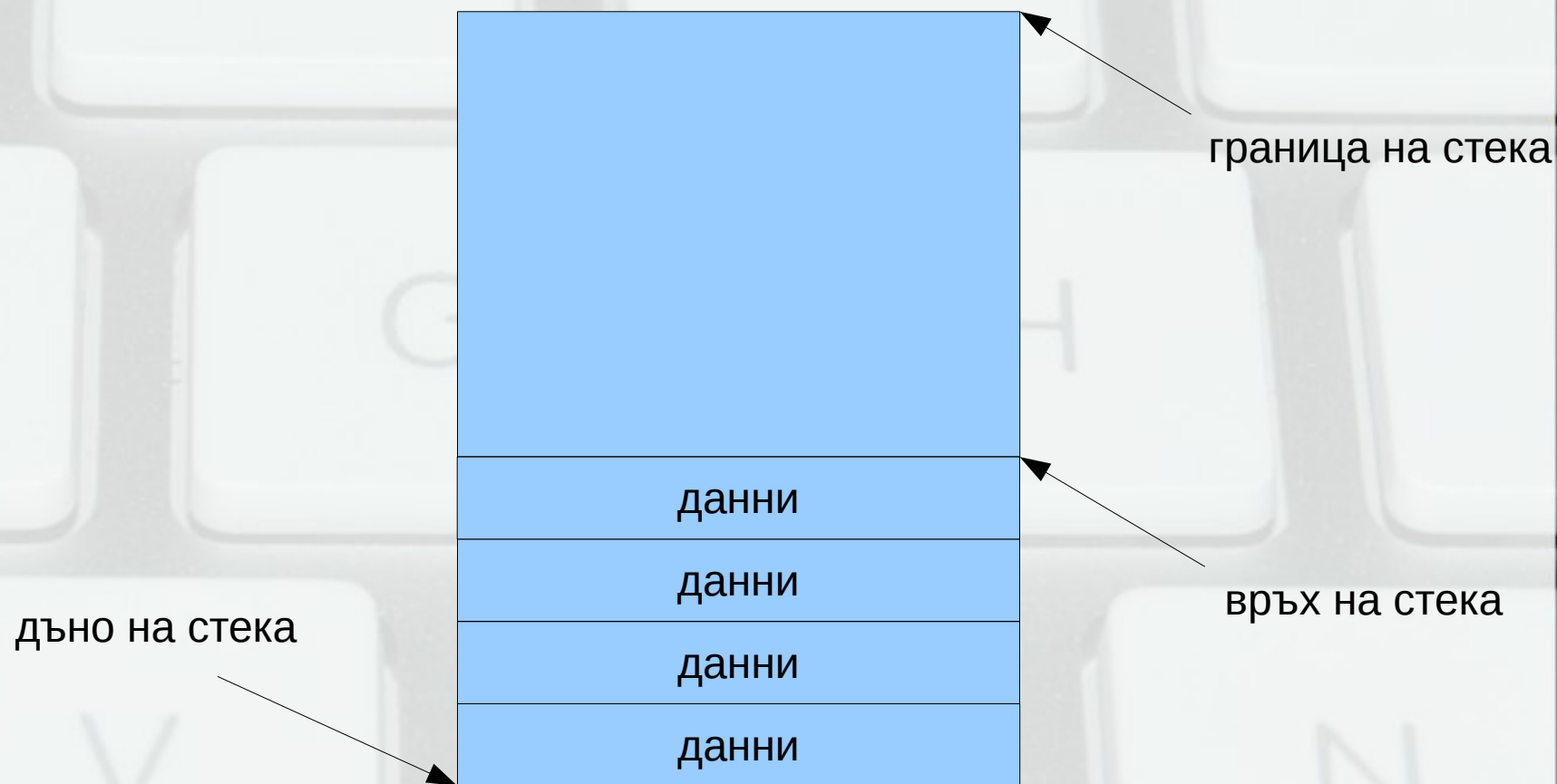
# Пример

```
double triarea(double x1, double y1, double x2, double y2,  
              double x3, double y3) {  
    double a = distance(x1, y1, x2, y2);  
    double b = distance(x2, y2, x3, y3);  
    double c = distance(x1, y1, x3, y3);  
    return heron(a, b, c);  
}  
double distance(double x1, double y1, double x2, double y2) {  
    return sqrt(sqr(x2-x1)+sqr(y2-y1));  
}  
double heron(double a, double b, double c) {  
    double p = (a+b+c)/2;  
    return sqrt(p*(p-a)*(p-b)*(p-c));  
}  
double sqr(double x) { return x*x; }
```

# Схема на програмната памет

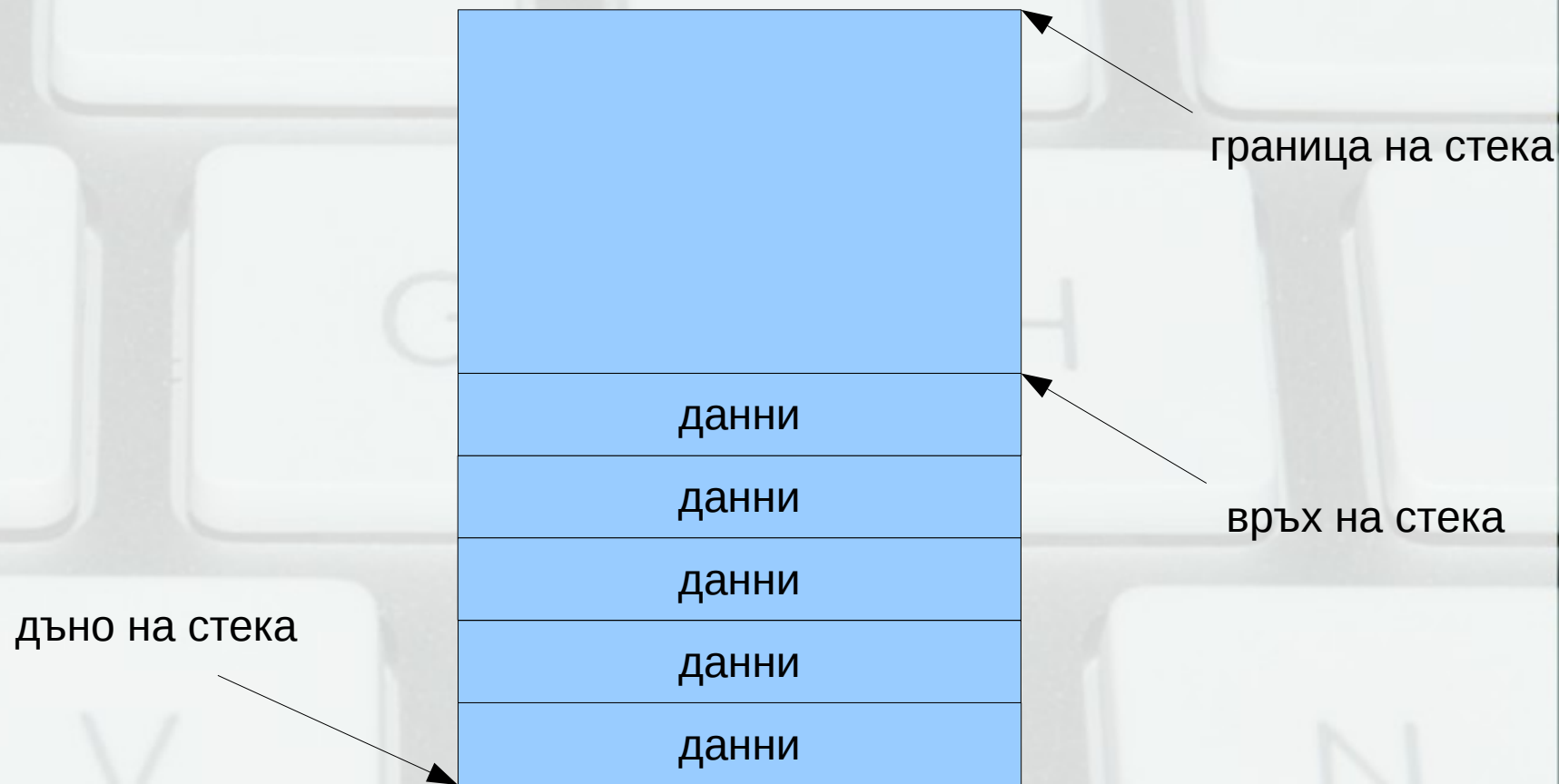


# Програмен стек

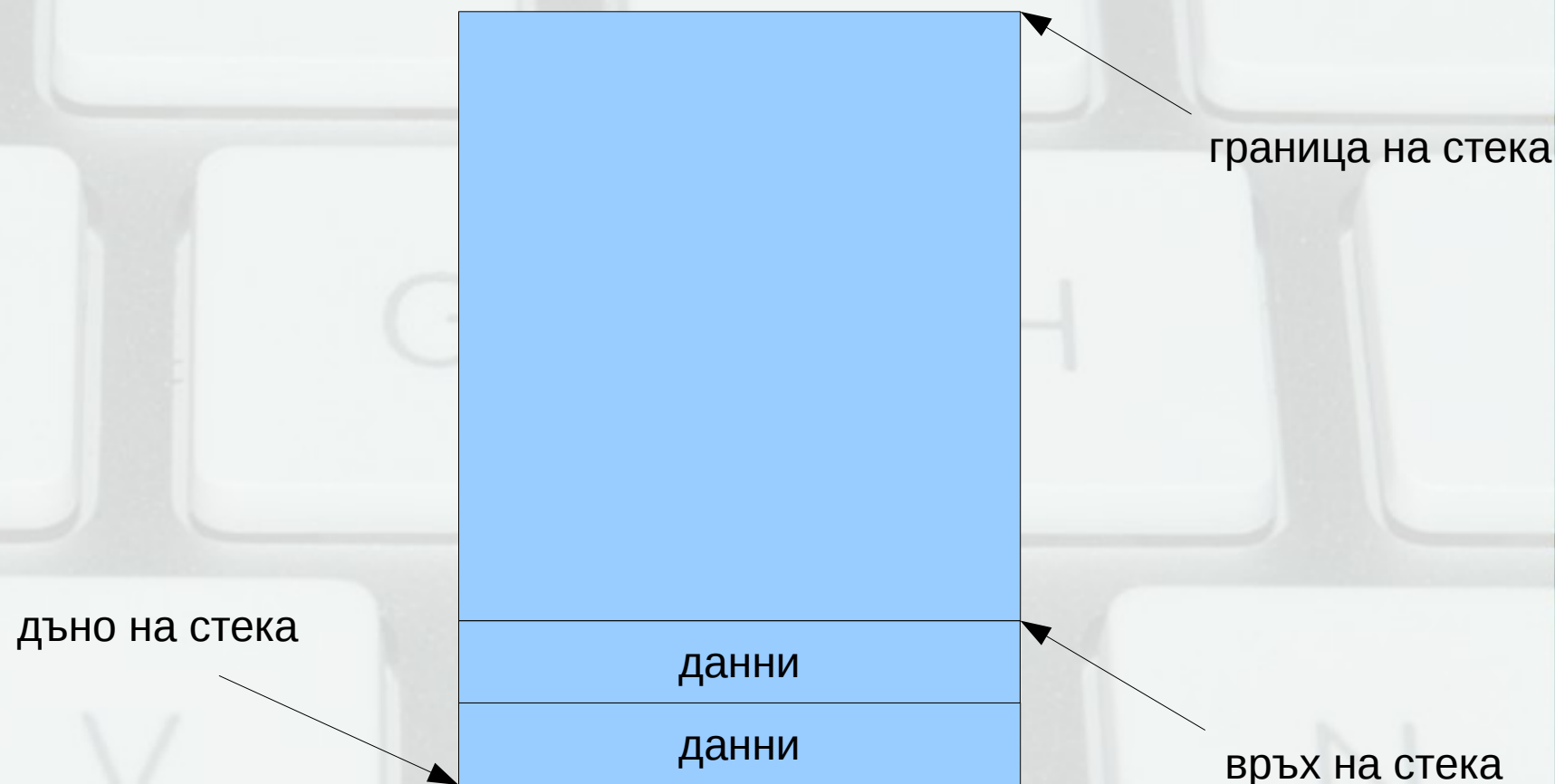




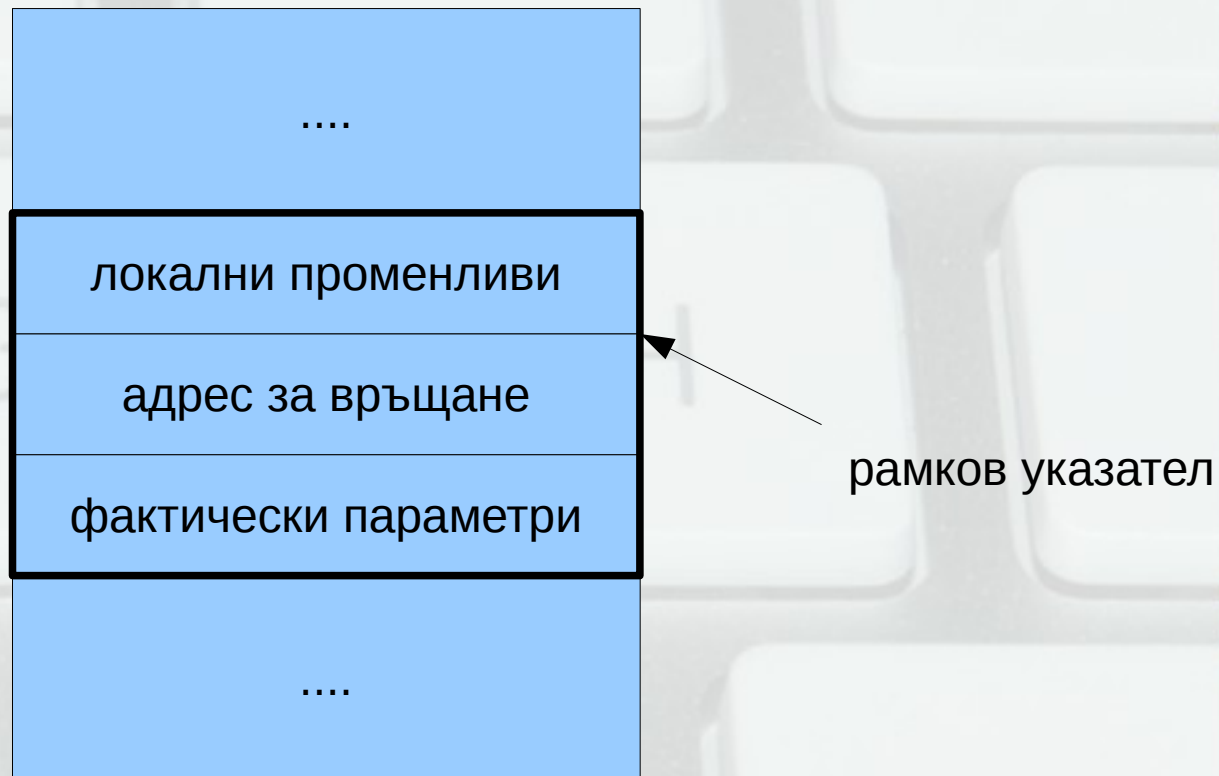
# Програмен стек



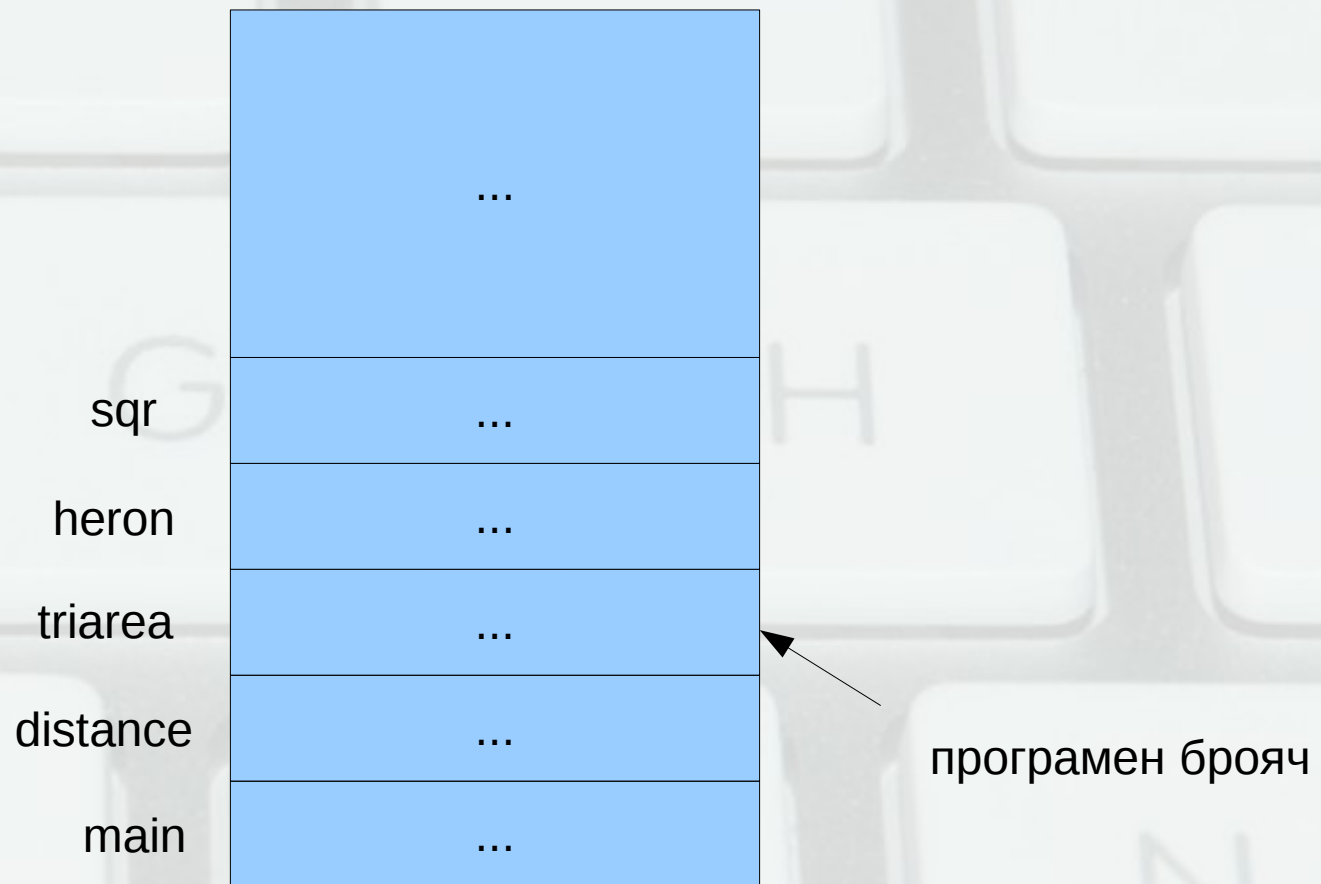
# Програмен стек



# 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 нейни промени изчезват



# Странични ефекти

- Използване на глобални променливи
- Използване на статични променливи **static** <дефиниция\_на\_променлива>
- Работа с вход/изход

## Предаване по псевдоним (референция) (call by reference)

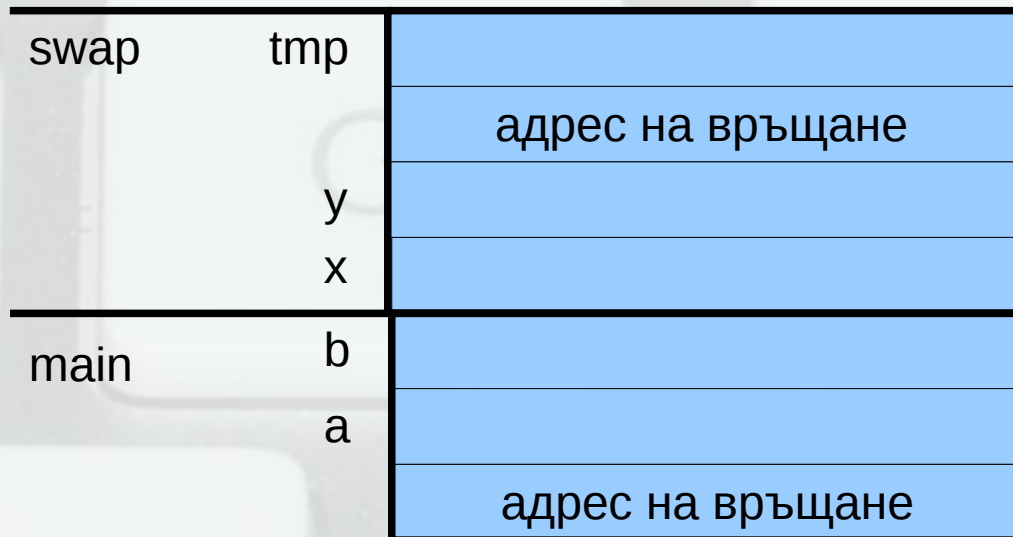
- Понякога искаме промените във **формалните** параметри да се отразят във **фактическите** параметри
- Тогава трябва да обявим, че фактическите параметри могат да бъдат променяни:
- `<параметър> ::= <тип>& <идентификатор>`
- Пример: `int add5(int& x) { x += 5; return x; }`
- `add5(3); int a = 3; cout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

## Предаване по псевдоним (референция) (call by reference)

- Пример: 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 по псевдоним



# Претоварване на функции (overloading)

- Сигнатурата на функцията зависи от:
  - типа на връщане
  - типа и реда на параметрите
- Функции с едно и също име и различна сигнатура се третират като различни
- Предимство: еднакво име независимо от типа
- Проблем: може да възникне нееднозначност при извикването



# Предаване на масиви като параметри

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

# Примерни функции

- Въвеждане на масив
- Извеждане на масив
- Търсене на елемент в масив
- Проверка за равенство на два низа

# Връщане на повече от един резултат

- Пример: намиране на най-малкия и най-големия елемент на масив
- ```
void findMinMax(int a[], int n,  
                int& min, int& max);
```