Функции

Какво е функция?

- Относително независима част от програмата, изпълняваща специфична задача
- Може да бъде изпълнявана многократно в различни моменти от живота на програмата
- Не същото като в математиката!
- Още: процедура, подпрограма, метод

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

- <сигнатура> { <тяло> }
- <сигнатура> ::= [inline] [<тип> | void]
  <идентификатор> (<формални_параметри>)
- <параметри> ::= <празно> | void |
  <параметър> {, <параметър> }
- <параметър> ::= <тип> [<идентификатор>]
- <тяло> ::= { <оператор> }

Извикване на функция

- <име>(<фактически_параметри>)
- <фактически_параметри> ::= <празно> | void | <израз> {, <израз> }
- 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

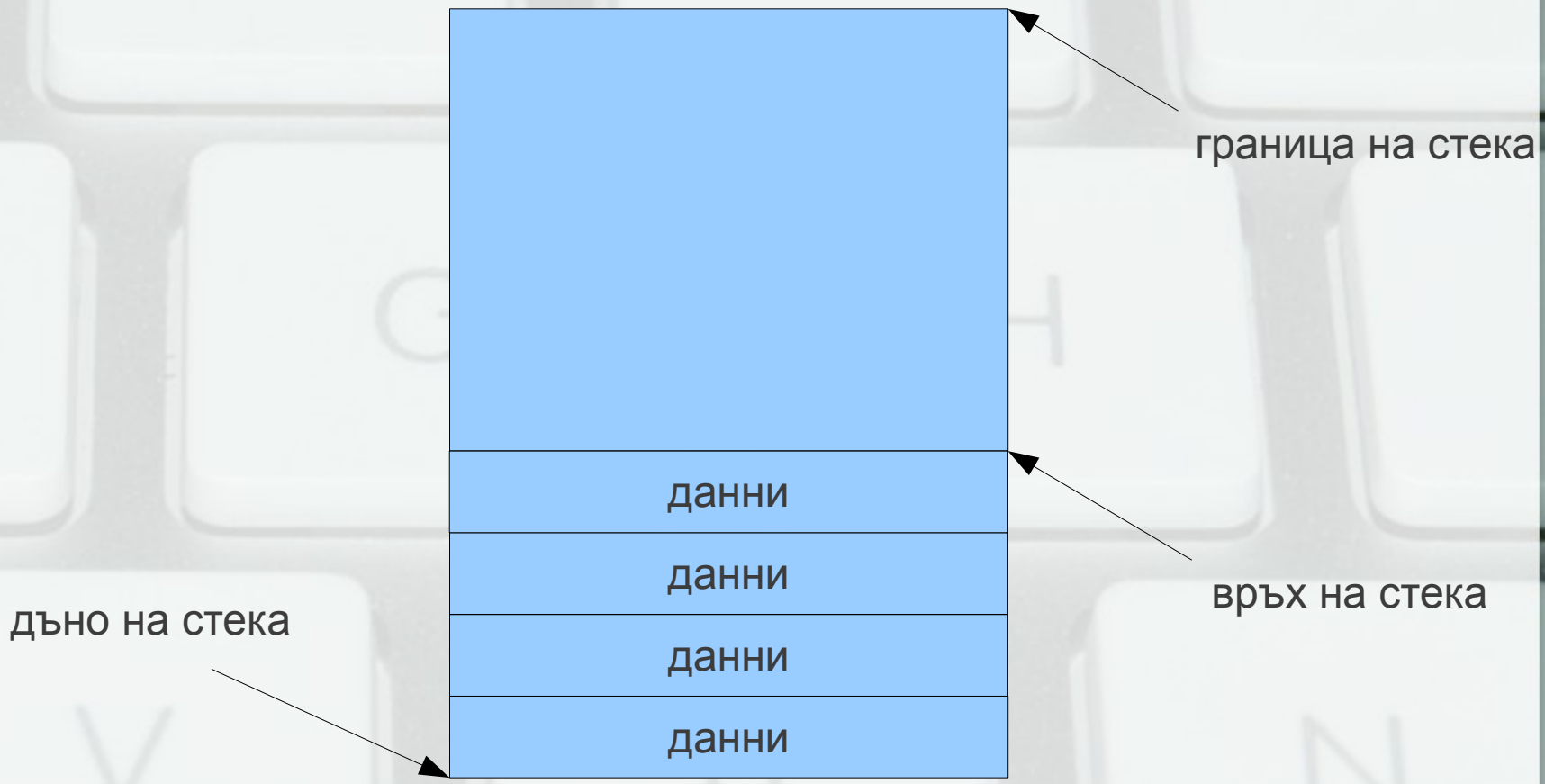
- <сигнатура>;
- Обещание за дефиниция на функция
- Декларацията не е задължителна
- Може да има много декларации на една и съща функция
- Но може да има само една дефиниция
- Неизпълнените обещания водят до грешка
  - освен когато не се разчита на тях

Пример

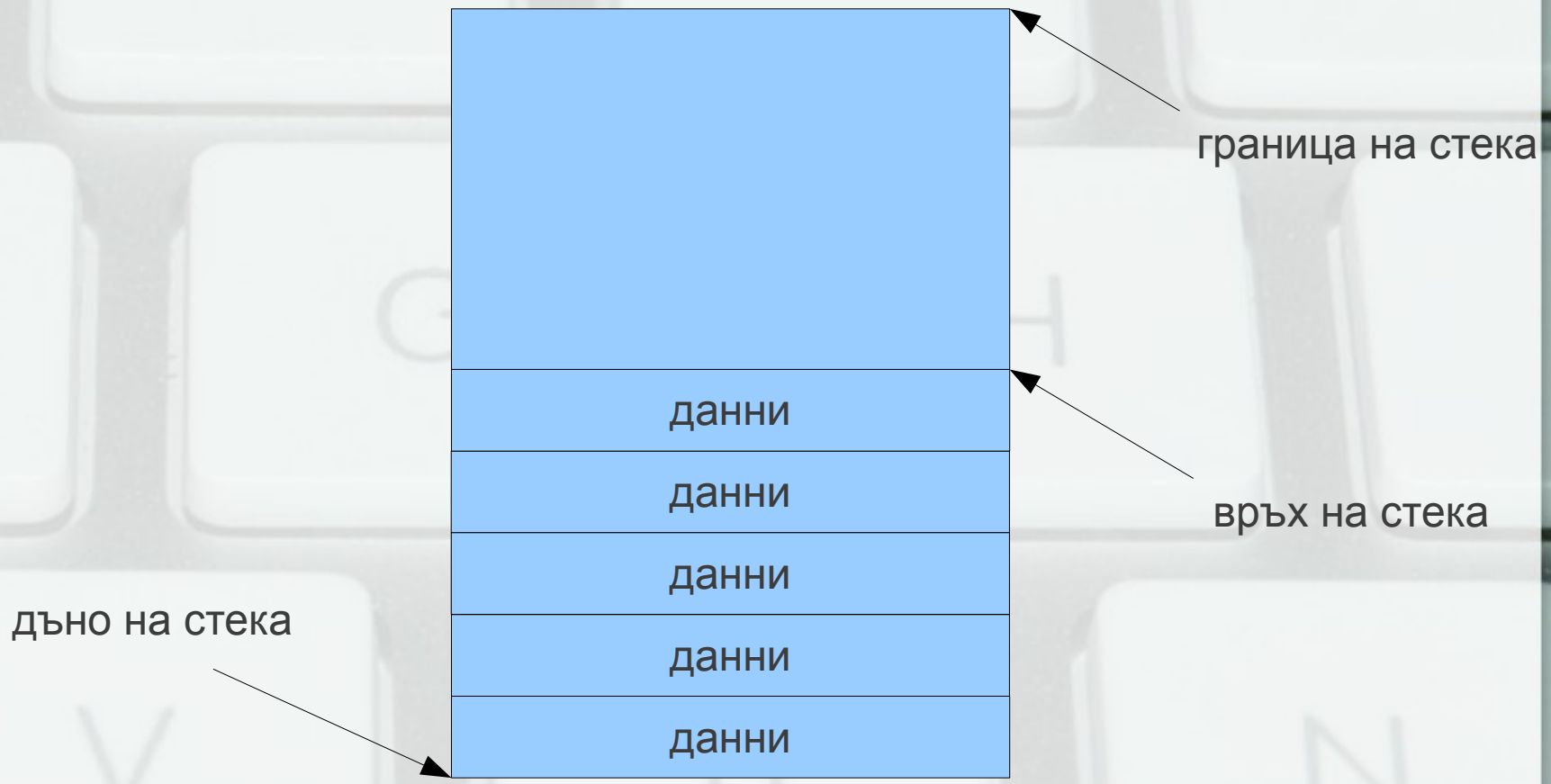
```
double triarea(double x1, double y1, double x2, double y2,
                    double x3, double y3) {
    double a = distance(x1, y1, x2, y2);
    double b = distance(x2, y2, x3, y3);
    double c = distance(x1, y1, x3, y3);
    return heron(a, b, c);
}
double distance(double x1, double y1, double 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

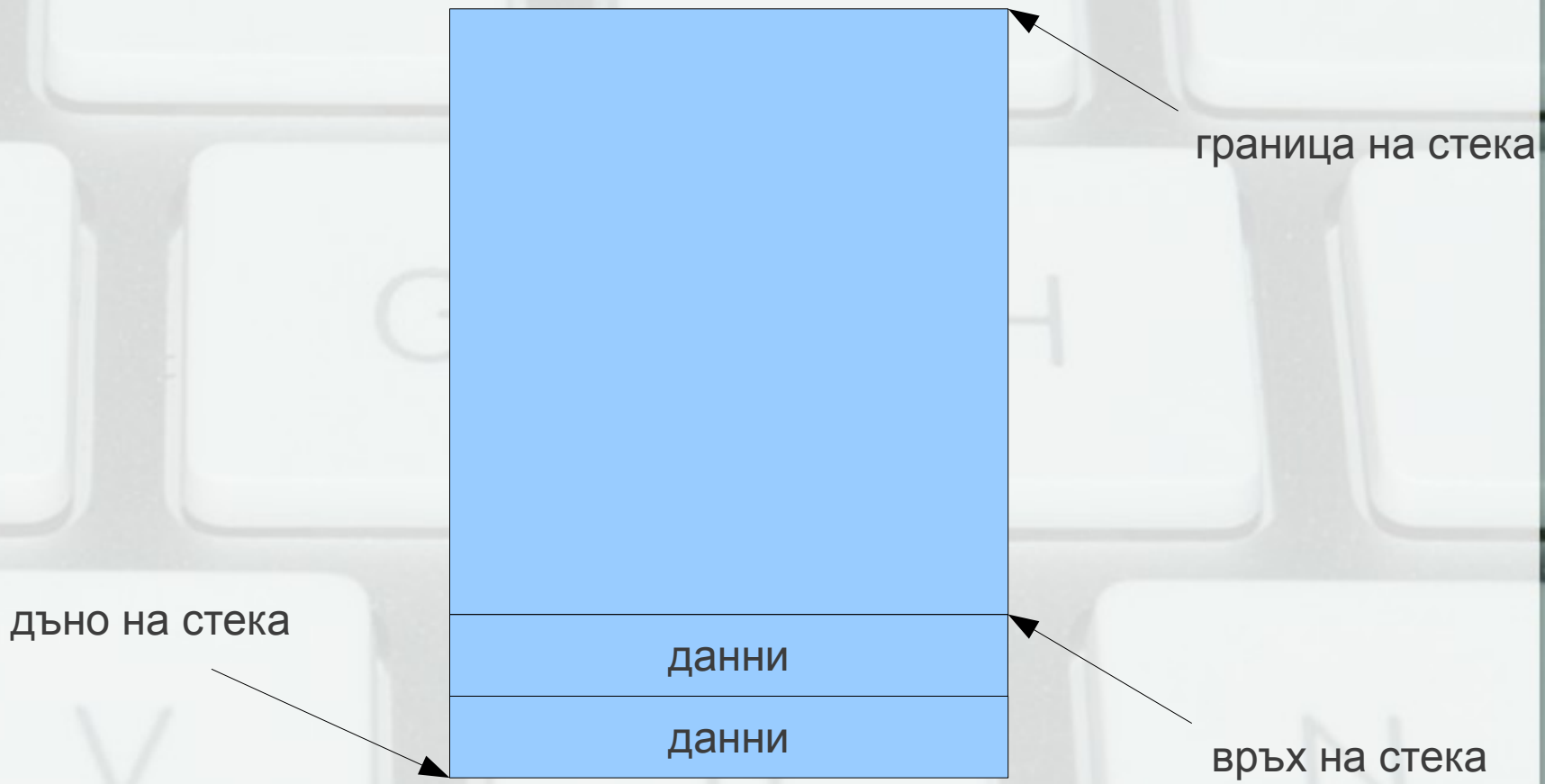
Програмен стек



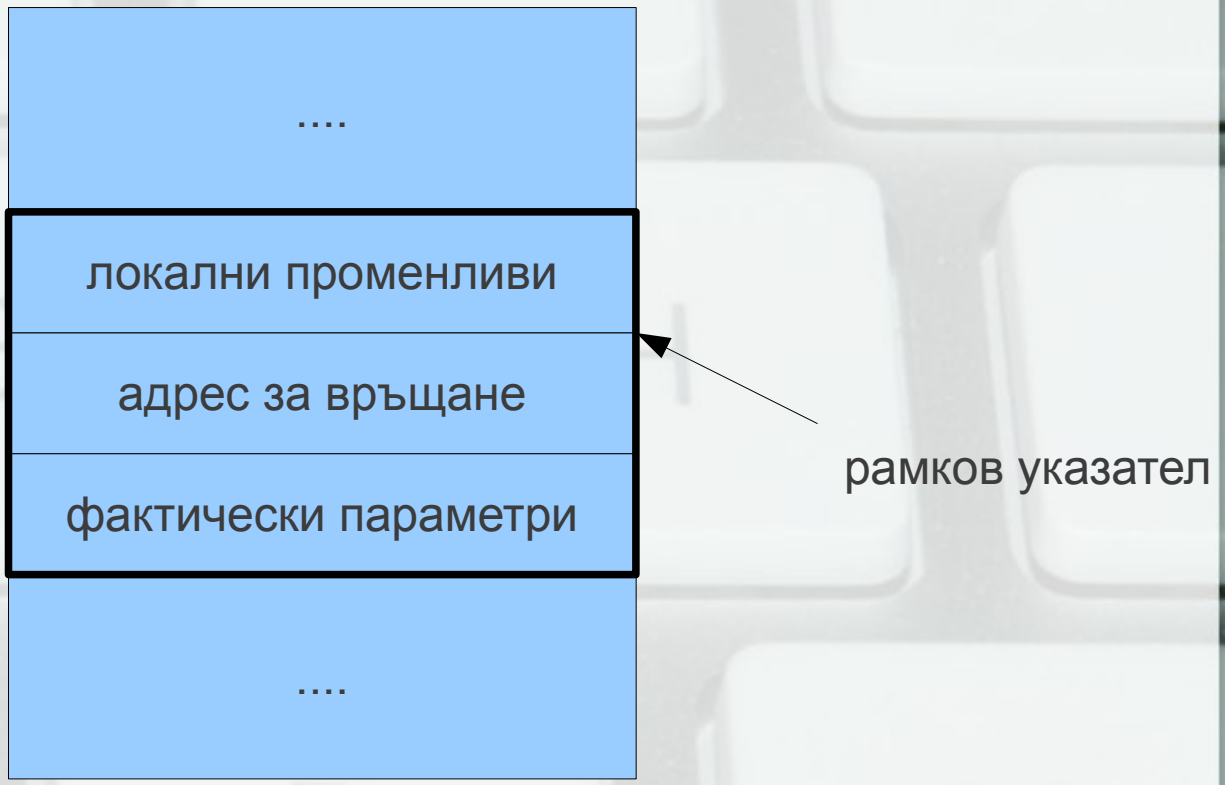
Програмен стек



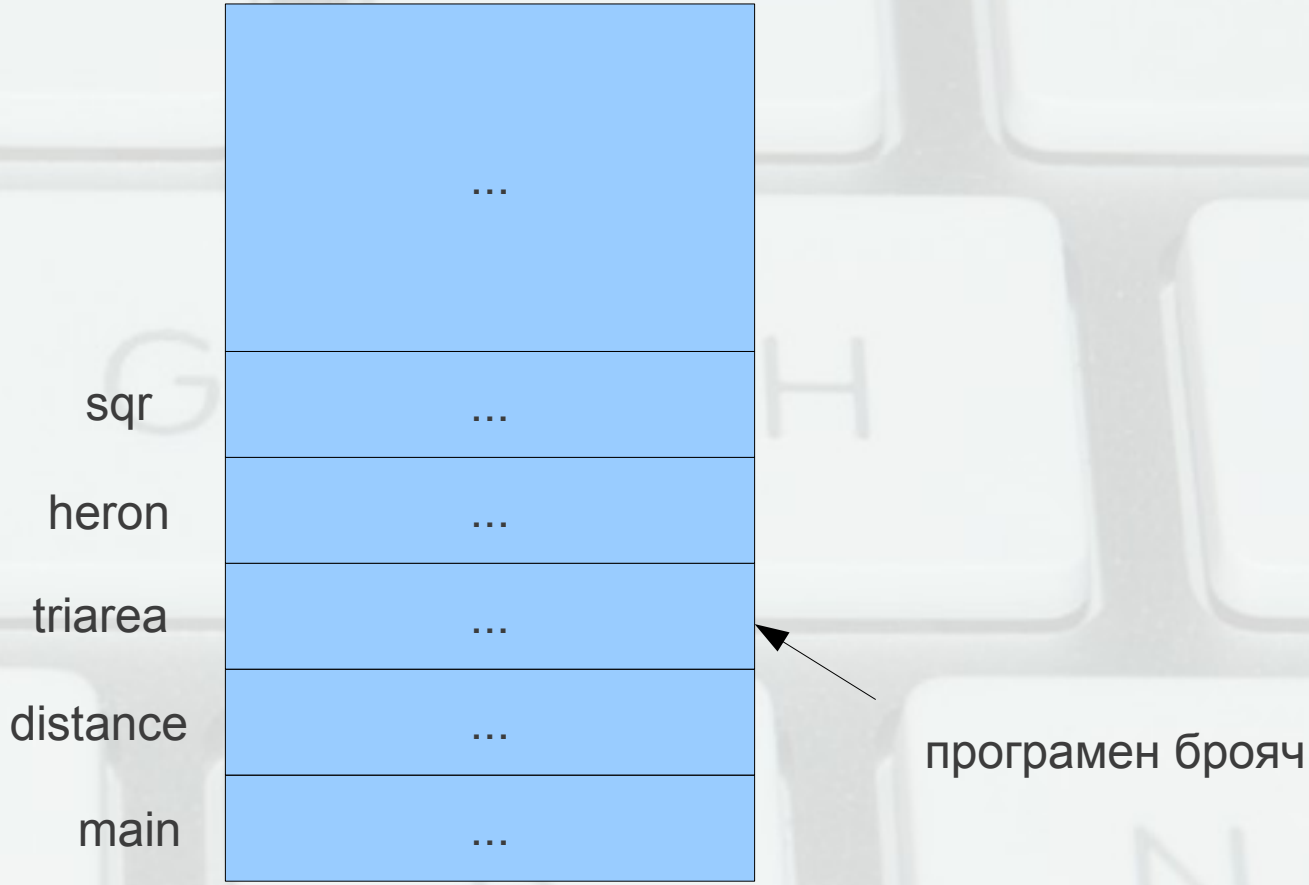
Програмен стек



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

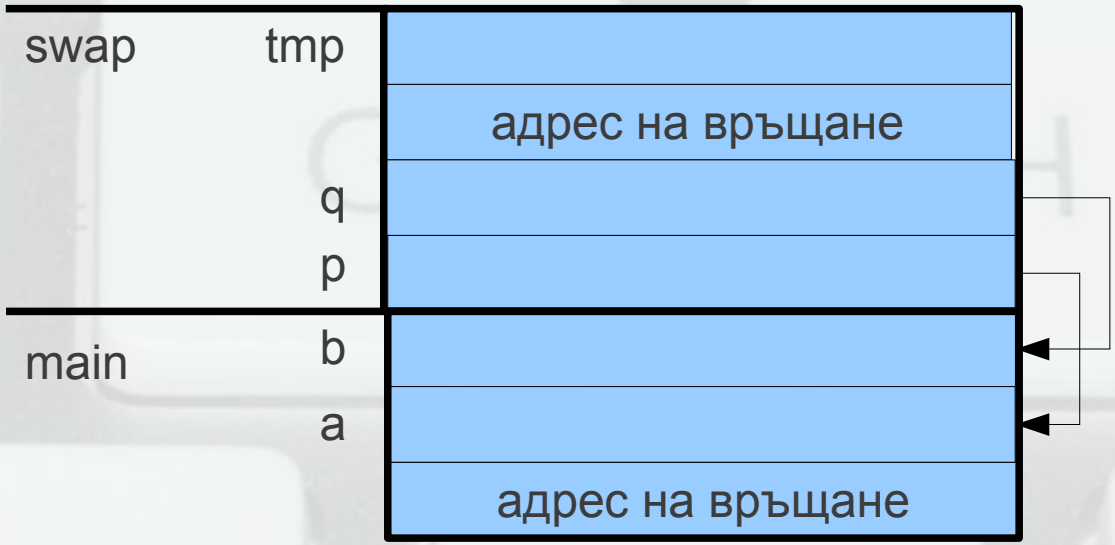
- Използване на глобални променливи
- Използване на статични променливи
  static <дефиниция_на_променлива>
- Работа с вход/изход

Предаване по указател (адрес) (call by pointer)

- 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

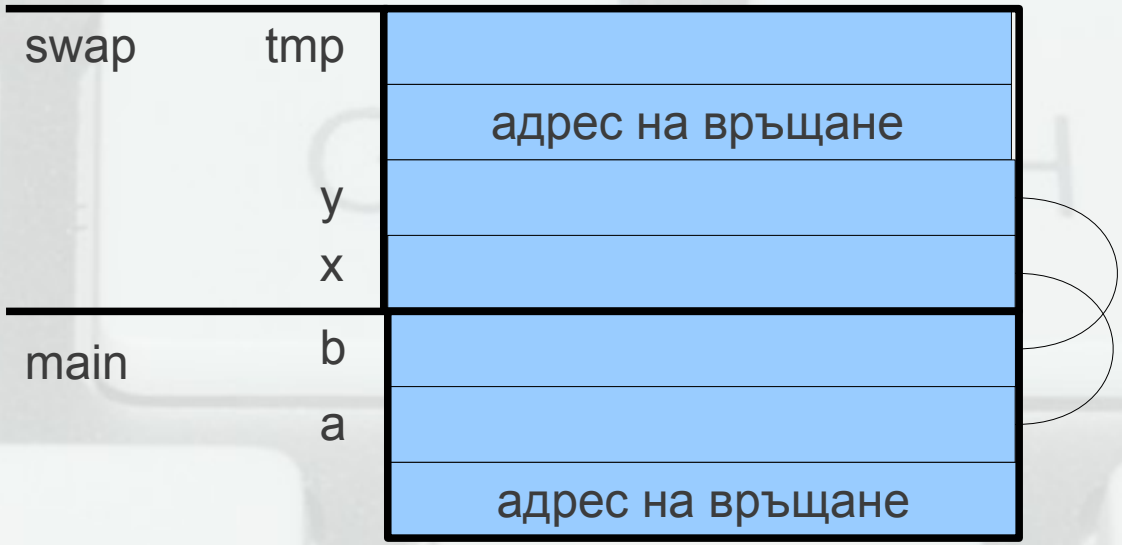


Предаване по псевдоним (референция) (call by reference)

- 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 неоднозначност при извикването

Предаване на масиви като параметри

- <параметър_масив> ::=
  <тип> <име>[[<константен_израз]] |
  <тип>* <име>
- изразът в скобите се игнорира!
- масивите се предават по указател
- 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);

Предаване на многомерни масиви като параметри

- <параметър_многомерен_масив> ::=
  <тип> <идентификатор>[[<константа>]] {[<константа>]} |
  <тип> (*<идентификатор>) {[<константа>]}
- Внимание! **int* a[10]** е различно от **int (*a)[10]**!
- константата в първите скоби се игнорира!
- многомерните масиви също се предават по указател

Предаване на многомерни масиви като параметри

- <параметър_многомерен_масив> ::=
  <тип> <идентификатор>[[<константа>]] {[<константа>]} |
  <тип> (*<идентификатор>) {[<константа>]}
- Внимание! **int* a[10]** е различно от **int (*a)[10]**!
- константата в първите скоби се игнорира!
- многомерните масиви също се предават по указател

Предаване на многомерни масиви като параметри

- <параметър_многомерен_масив> ::=
  <тип> <идентификатор>[[<константа>]] {[<константа>]} |
  <тип> (*<идентификатор>) {[<константа>]}
- първата размерност трябва да се подаде като допълнителен параметър
- другите размерности се предават, за да се пресмятат правилно позициите на елементите

Примерни функции

- Извеждане на матрица от числа
- Прочитане на масив от низове
- Проверка за срещане на дума в масив от низове
- Умножение на две правоъгълни матрици

Указателите като върнат резултат

- Внимание: връщат се указатели към обекти, които ще продължат да съществуват след като функцията приключи
-
```
int* pointMax(int* p, int* q) {
    if (*p > *q)
        return p;
    return q;
}
int* r = pointMax(&a, &b); (*r)--;
```

Псевдонимите като върнат резултат

- Важи същото правило като за указателите
- int& middle(int& x, int& y, int& z) {

```
    if (x <= y && y <= z || z <= x && x <= y)
        return y;
    if (y <= z && z <= x || x <= z && z <= y)
        return z;
    return x;
}
```

- middle(x, y, z) = 5;

Масивите като върнат резултат

- Функциите не могат да бъдат от тип масив
- Но могат да бъдат от тип указател
- По този начин може да се връщат едномерни масиви
- Внимание: връщат се само масиви, които ще продължат да съществуват след като функцията завърши

Примери

- Връщане на позицията на първото срещане на даден символ в низ
- Връщане на позицията на първото различие между два низа