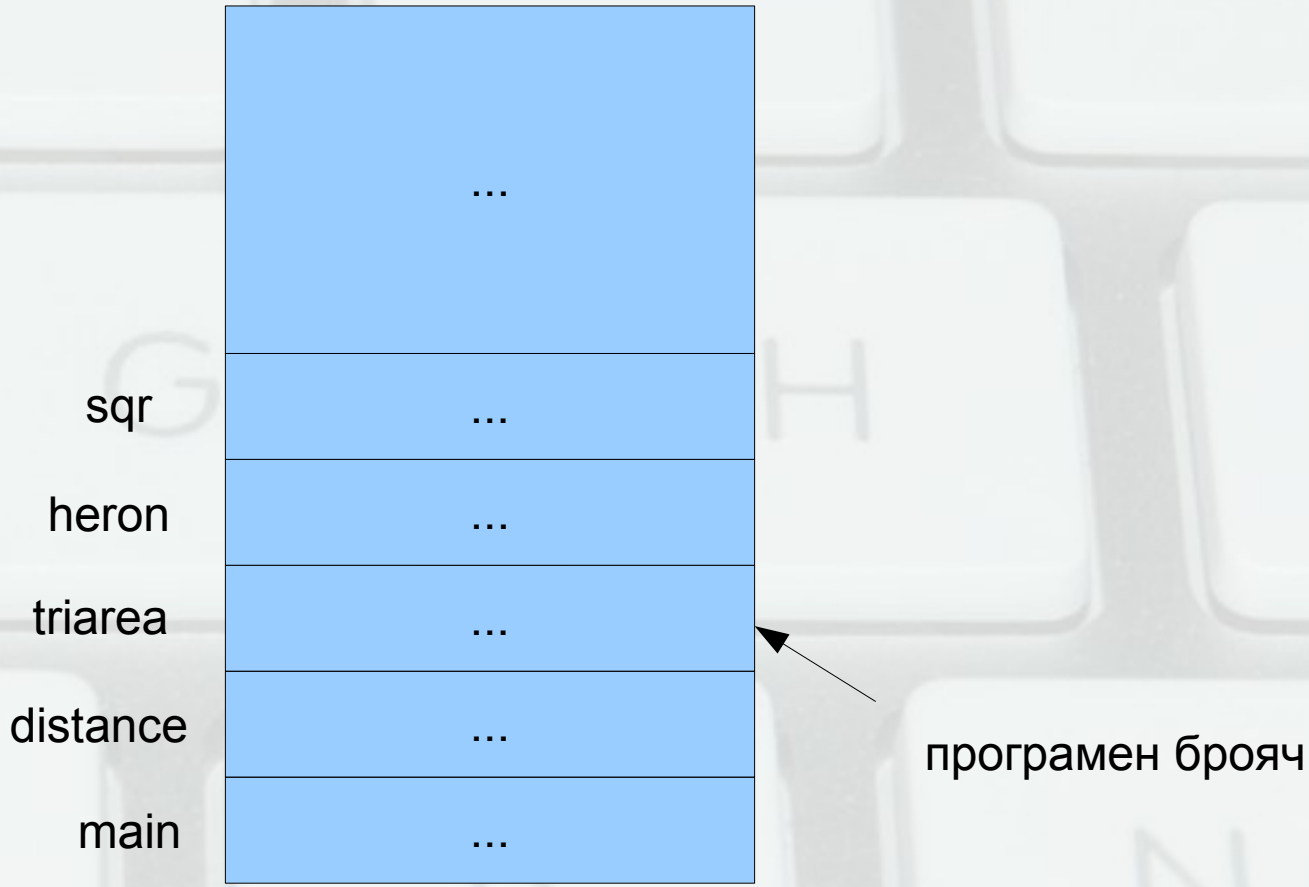
Функции от по-висок ред

Преговор: програмна памет

Област за програмен код



Указател към функция

- Машинният код за всяка функция са разположени в паметта
- Указател към функция е адресът на първата инструкция във функцията
- Името на функцията е константен указател към кода ѝ

Дефиниране на указатели към функции

- <тип> (*<идентификатор>)
  (<формални_параметри>) [= <указател>];
- имената на параметрите могат да се пропуснат
- Примери:
- void (*f)(int&, int&);
- f = swap;
- double (*op)(double) = sin; op = cos; op = NULL;
- ~~void (*p)(int, int&) = f; sin = op;~~

Извикване на функция чрез указател

- void (*f)(int&, int&) = swap;
- int x, y;
- swap(x,y); ↔ (*f)(x,y); ↔ f(x,y);

Потребителски типове

- typedef <декларация_на_променлива>
- Името на променливата означава името на потребителски дефиниран тип
- Примери:
- typedef int number;
- number x; ↔ int x;
- number f(number y) {...} ↔ int f(int y) {...}

Потребителски типове

- typedef double array_type[5][10];
- array_type a; ↔ double a[5][10];
- typedef int** pointer2;
- int x; int* p = &x; pointer2 q = &p;
- typedef int& ref;
- ref y = x;

Потребителски типове за функции

- typedef double (*math_fun)(double);
- math_fun p = exp; p = log;
- typedef void (*simple_function)();
- void h() { cout << “h()\n”; }
- simple_function q = h; q();
- void (*r)() = q;

Примерна сума 1

- sin(1) + sin(2) + sin(3) + ... + sin(n)
- double sum_sin(int n) {
  double s = 0;
  for(int i = 1; i <= n; i++) s += sin(i);
  return s;
  }

Примерна сума 2

- cos(1) + cos(2) + cos(4) + ... + cos(n)
- ```
  double sum_cos(int n) {
      double s = 0;
      for(int i = 1; i <= n; i *= 2) s += cos(i);
      return s;
  }
  ```
  ```

Открийте разликите!

- ```
  double sum_sin(int n) {
      double s = 0;
      for(int i = 1; i <= n; i++) s += sin(i);
      return s;
  }
  ```
- ```
  double sum_cos(int n) {
      double s = 0;
      for(int i = 1; i <= n; i *= 2) s += cos(i);
      return s;
  }
  ```

Открийте разликите!

- double sum_sin(int n) {
  double s = 0;
  for(int i = 1; i <= n; i++) s += sin(i);
  return s;
  }
- double sum_cos(int n) {
  double s = 0;
  for(int i = 1; i <= n; i *= 2) s += cos(i);
  return s;
  }

Общият шаблон

- double **<name>**(int n) {
  double s = 0;
  for(int i = 1; i <= n; i = **<next>**(i))
  s += **<f>**(i);
  return s;
  }

Функциите като параметри

- double sum(int n, double(*f)(double), int(*next)(int)) {
  double s = 0;
  for(int i = 1; i <= n; i = next(i))
  s += f(i);
  return s;
  }
- int plus1(int i) { return i + 1; }
- int mult2(int i) { return i * 2; }
- sum_sin(n) ↔ sum(n, sin, plus1)
- sum_cos(n) ↔ sum(n, cos, mult2)

Accumulate

- f(a) ○ (f(next(a)) ○ (f(next(next(a))) ○
  (... ○ (f(b) ○ ⌀)...)))
- typedef int (*next_function)(int);
- typedef double (*term_function)(double);
- typedef double (*operation)(double,double);
- double accumulate (operation op,
  double base_value,
  double a, double b,
  term_function f,
  next_function next);

Accumulate

- double accumulate (operation op,
  double base_value,
  double a, double b
  term_function f,
  next_function next) {
  double s = base_value;
  for(int i = a; i <= b; i = next(i))
  s = op(s, f(i));
  return s;
  }

Accumulate: примери

- double plus(int x, int y) { return x + y; }
- double mult(int x, int y) { return x * y; }
- sum_sin(n) ↔ accumulate(plus, 0, 1, n, sin, plus1)
- sum_cos(n) ↔ accumulate(plus, 0, 1, n, cos, mult2)
- prod_tan(a, b) ↔ accumulate(mult, 1, a, b, tan,plus1)
- sum(n, f, next) ↔ accumulate(plus, 0, 1, n, f, next)
- product(a, b, f, next) ↔
  accumulate(mult, 1, a, b, f, next)

Задачи

- Пресметнете

$$\sum_{i=0}^{n} \frac{x^i}{i!} \qquad \binom{n}{k}$$

$$x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \ldots + (-1)^n \frac{x^{2n+1}}{(2n+1)!}$$

Функциите като върнат резултат

- Да се напише функция, която по зададен номер връща математическа функция:
  - при 1 връща sin
  - при 2 връща cos
  - при 3 връща exp
  - при 4 връща log

Декларация на решението

- double (*choose_function(double))(int n);
- по-просто:
- typedef double (*math_fun)(double);
- math_fun choose_function(int n);

Решение

```
math_fun choose_function(int n) {
    switch(n) {
        case 1 : return sin;
        case 2 : return cos;
        case 3 : return exp;
        case 4 : return log;
        default : return NULL;
    }
```

Задача

- Да се напише функция, която по зададена едноаргументна функция f връща нейната производна
- math_fun derive(math_fun f);

Решение

- Използване на глобална променлива
- math_fun function = NULL;
- double derivative(double x) {
  double eps = 1E-10;
  return (function(x + eps) - function(x)) / eps;
  }
- math_fun derive(math_fun f) {
  function = f;
  return derivative;
  }

Пример за използване на derive

- math_fun mycos = derive(sin);
- cout << mycos(x) << ' ' << cos(x);
- cout << exp(x) << ' ' << derive(exp)(x);