# Programiranje 2

doc.dr.sc. Goranka Nogo

PMF – Matematički odsjek, Zagreb

# Kontakt

- ured: 228, drugi kat
- e-mail: `nogo@math.hr`
- konzultacije:
  - ponedjeljak, 14:00-15:00
  - četvrtak, 11:00-13:00
  - neki drugi termin, uz prethodni dogovor putem e-maila.
- Na konzultacije je potrebno donijeti vlastite zabilješke s predavanja.

# Osnovna pravila „lijepog” ponašanja

Molim da:

- ne ometate pričom izvođenje nastave
- ne kasnite na predavanja
- održavate red u predavaonici (ne ostavljajte iza sebe plastične boce, papire, …)
- utišate i pospremite mobitele
- isključite računala
- …

# Preddiplomski studij - računarski kolegiji

- Programiranje 1
- Programiranje 2
- Strukture podataka i algoritmi
- Računarski praktikum 1

# Ishodi učenja

- Po uspješnom završetku kolegija student/ica može:
  - opisati koncept rekurzije i navesti primjere korištenja
  - usporediti iterativna i rekurzivna rješenja osnovnih problema
  - opisati najčešće primjene i napisati jednostavne programe koji koriste sljedeće strukture podataka: polja, stringove i vezane liste
  - usporediti prednosti i mane statičkog i dinamičkog načina implementacije struktura podataka

# Ishodi učenja - nastavak

- Po uspješnom završetku kolegija student/ica može:
  - odabrati prikladnu strukturu podataka za modeliranje danog problema
  - oblikovati, analizirati i implementirati brze algoritme sortiranja (Quicksort, Mergesort).

# Pregled sadržaja kolegija

- Funkcije
- Struktura programa
- Dvodimenzionalna polja
- Pokazivači
- Strukture
- Datoteke
- Preprocesor
- Standardna C biblioteka

# Literatura

- Zabilješke s predavanja i vježbi te popratni materijali dostupni na službenim stranicama kolegija.
- B. W. Kernighan, D. M. Ritchie: *The C Programming Language*, 2nd Edition. Prentice Hall, 1988.
- B. S. Gottfried: *Theory and Problems of Programming with C*. Schaum's outline series, McGraw-Hill, (najnovije izdanje).

# Način polaganja ispita

- Elementi ocjenjivanja:
  - prvi kolokvij
  - drugi kolokvij.

- Ukupan broj bodova na prvom kolokviju je najmanje 40, a na drugom najmanje 60 (oba kolokvija mogu imati bonus bodove).

# Način polaganja ispita (2)

- Za prolaz kolegija je potrebno sakupiti barem 45 bodova **na redovitim kolokvijima** (prvi i drugi zajedno ili popravni).

- Pri tome na barem jednom zadatku, na prvom ili drugom kolokviju, treba sakupiti najmanje 80% mogućih bodova.

# Način polaganja ispita (3)

- Napomene.
  - Studenti/ce koji polože kolegij, a nisu zadovoljni ocjenom, mogu odgovarati usmeno.
  - Nastavnik ima pravo pozvati studenta/icu na ispit.

- Više detalja o načinu polaganja ispita možete naći na službenim web stranicama kolegija: http://degiorgi.math.hr/prog2/

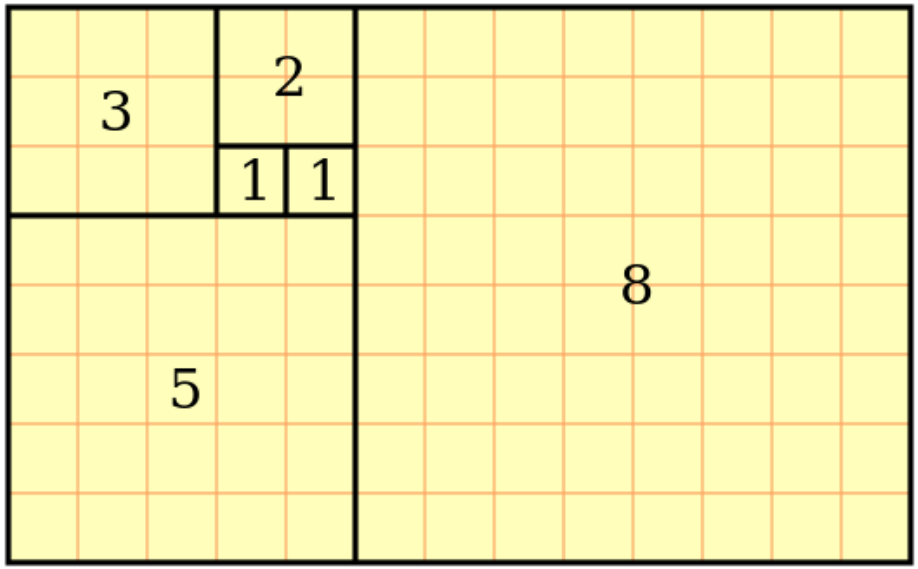
# Pitanja?

# Funkcije

# Primjer funkcije: Fibonaccijevi brojevi

$f_0 = 0, f_1 = 1,$

$f_n = f_{n-1} + f_{n-2}, n \geq 2$



# Fibonaccijevi brojevi: iterativna verzija

```
int fibonacci(int n)   /* n ≥ 0 */
{  int i;
   int f, f0 = 0, f1 = 1;
   if(n == 0) return 0;
   else if(n == 1) return 1;
        else
             for(i = 2; i <= n; i++){
                  f = f0 + f1;
                  f0 = f1;
                  f1 = f;}
   return f;
}

printf("%d\n", fibonacci(n));
```

# Načini prijenosa argumenata

- Načini prijenosa (predavanja) argumenata prilikom poziva funkcije:
  - prijenosom vrijednosti argumenata – *call by value*
  - prijenosom adresa argumenata – *call by reference*.

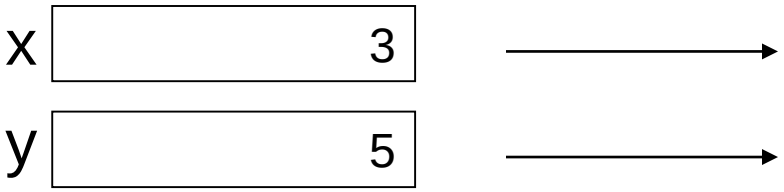
# Prijenos vrijednosti argumenata

```
#include <stdio.h>

void kvadrat(int x, int y)
{
    y = x * x;
    printf("Unutar funkcije: \
    x = %d, y = %d.\n", x, y);
}
```

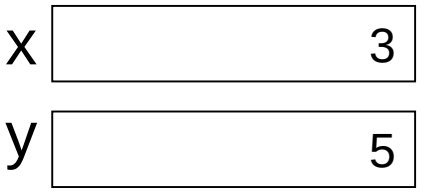
```
int main()
{
    int x = 3, y = 5;

    printf("Prije poziva:
    x = %d, y = %d.\n", x, y);
    kvadrat(x, y);
    printf("Nakon poziva: \
    x = %d, y = %d.\n", x, y);
    return 0;
}
```

# Prijenos vrijednosti argumenata (2)

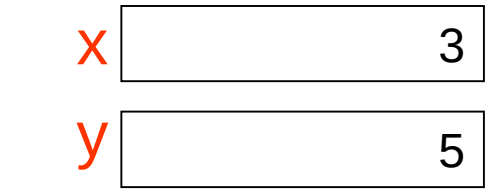
- Funkcija `main`:



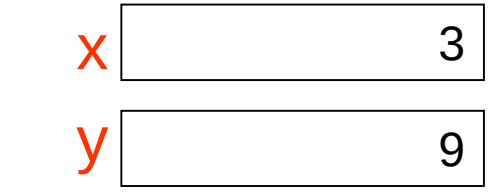

- Nakon poziva




- Poziv funkcije




- Izvođenje funkcije: y = x * x;



# Prijenos adrese argumenata

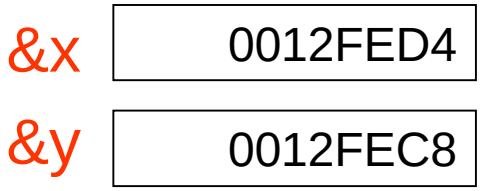
```
#include <stdio.h>

void kvadrat(int *x, int *y)
{
   *y = *x * *x;
   printf("Unutar funkcije: \
   x = %d, y = %d.\n", *x, *y);
}
```

```
int main()
{
   int x = 3, y = 5;

   printf("Prije poziva:
   x = %d, y = %d.\n", x, y);
   kvadrat(&x, &y);
   printf("Nakon poziva: \
   x = %d, y = %d.\n", x, y);
   return 0;
}
```

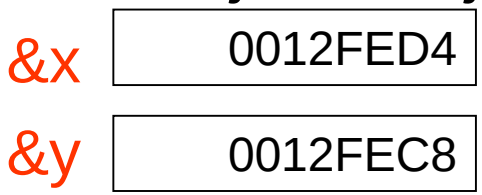
# Prijenos adrese argumenata (2)

- Funkcija `main`:

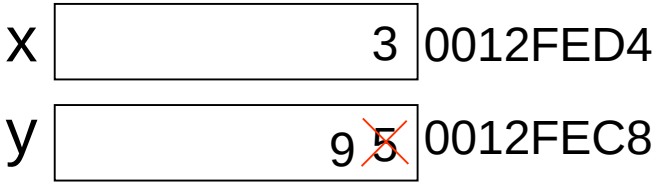

- Poziv funkcije




- Izvođenje funkcije: *y = *x * *x;




- Nakon poziva



# Rekurzivne funkcije

- Osnovna ideja: funkcija (procedura) poziva samu sebe ili, preciznije
  - u definiciji funkcije koristimo funkciju koju definiramo.
- Rekurzivna rješenja su u pravilu
  - kraća (imaju elegantniji zapis).
- Mogu biti
  - memorijski
  - vremenski

  zahtjevnija.
- Uvijek moraju postojati osnovni slučajevi koji se izvršavaju nerekurzivno (inicijalizacija rekurzije).

# Primjer: najveći element polja

```
int T(int a[ ], int l, int d){
    int max1, max2;

    if (l == d)
       return a[l];
    else{
       max1 = T(a, l, (d + l) / 2);
       max2 = T(a, (d + l) / 2+1, d);

     if (max1 > max2)
        return max1;
     else
        return max2; } }
```

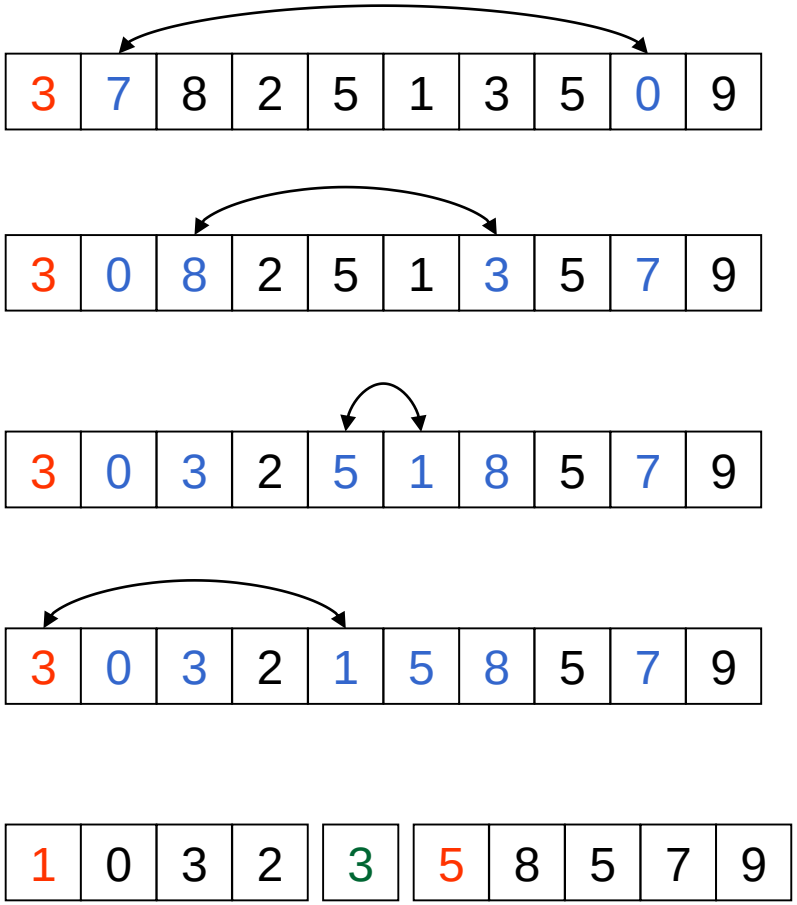
Primjer poziva funkcije T:

```
int najveci;
int T[100];
…
Najveci = T(a, 0, 99)
```

# Primjer: algoritam Quicksort

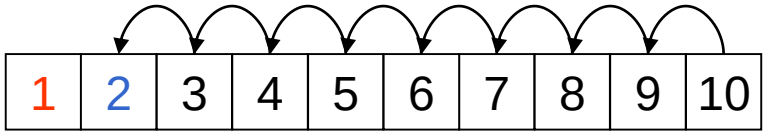
- Odabere se jedan element niza, tzv. **ključni element**.
- Preslože se elementi niza tako da
  - ključni element dođe na njegovo pravo mjesto u sortiranom nizu
  - se lijevo od ključnog elementa nalaze svi elementi niza koji su manji (ili jednaki) od njega
  - se desno nalaze svi elementi niza koji su veći od njega.
- Rekurzivno se sortiraju podniz manjih, odnosno podniz većih elemenata.

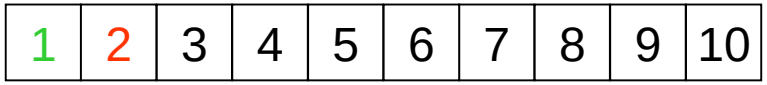
# Quicksort (2)


3 7 8 2 5 1 3 5 0 9
3 0 8 2 5 1 3 5 7 9
3 0 3 2 5 1 8 5 7 9
3 0 3 2 1 5 8 5 7 9
1 0 3 2 3 5 8 5 7 9

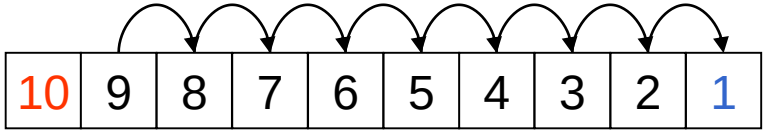
# Složenost

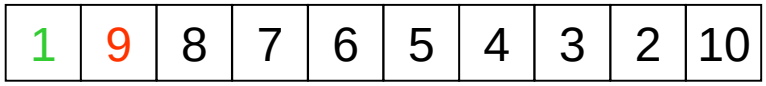
- Prosječna složenost: $\Theta(n\log_2 n)$.
- Složenost u najgorem slučaju: $O(n^2)$.




itd, ili




- Autor: C.A.R.Hoare, 1961.

# Funkcija `swap`

```
#include <stdio.h>

 /* Sortiranje algoritmom Quicksort.
    x[l] je ključni element - dovodimo ga na pravo mjesto u polju. */

void swap(int *a, int *b){
    int temp;

    temp = *a;
    *a = *b;
    *b = temp;
    return;
}
```

# Funkcija `quick_sort`

```
void quick_sort(int x[], int l, int d){
   int i, j;

   if (l < d) {
        i = l + 1;
        j = d;
        /* Mora se ispitati i slučaj i == j. */
        while (i <= j) {
                while (i <= d && x[i] <= x[l]) ++i;
                while (x[j] > x[l]) --j;
                if (i < j) swap(&x[i], &x[j]);
        }
```

# Funkcija `quick_sort` - nastavak

```
        if (l < j) swap(&x[j], &x[l]);
        quick_sort(x, l, j - 1);
        quick_sort(x, j + 1, d);
    }

    return;
}
```

# Funkcija `main`

```
int main(void) {
   int i, n;
   int x[ ] = {42, 12, 55, 94, 18, 44, -67};

   n = 7;
   quick_sort(x, 0, n - 1);

   printf("\n sortirano polje x\n");
   for (i = 0; i < n; ++i) {
        printf(" x[%d] = %d\n", i, x[i]);
   }
   return 0;
}
```