

---

# Znakovni nizovi (stringovi)

---

nastavak

---

# Primjer:

```
int main()
{
    char *v[2], q[80];

    gets(q);    /* učitano je: plava */
    v[0] = q;

    gets(q);    /* učitano je: crvena */
    v[1] = q;

    puts(v[0]); /* Što će se ispisati? */

    return 0;
}
```

---

---

# Primjer: Sortiranje rječnika

- Program se sastoji od sljedeća ti koraka:
  - unosa riječi sa standardnog ulaza
  - sortiranja unesenih riječi
  - ispisa sortiranih riječi.

```
#include <stdio.h>  
#include <string.h>  
#include <stdlib.h>
```

```
#define MAX 100
```

```
char *v[MAX];  
int broj_rijeci;
```

---

# Funkcija unos

```
int unos(char *v[]){
    char w[80*MAX], *q = w;
    int flag = 1, broj = 0;

    while(flag){
        if(broj > MAX) return -1;
        v[broj] = gets(q);

        if(strlen(v[broj]) == 0) return broj;
            q += strlen(q) + 1;
            broj++;
        }
    }
```

---

# Funkcija sort

```
void sort(char *v[], int n)
{
    char *temp,*min; int i,j,k;

    for(i = 0; i < n - 1; i++){
        min = v[i];
        k = i;
        for(j = i + 1; j < n; j++){
            if(strcmp(min, v[j]) >= 0){
                min = v[j];
                k = j;
            }
        }
    }
```

---

# Funkcija `ispis`

```
temp = v[i];
```

```
v[i] = v[k];
```

```
v[k] = temp;
```

```
}
```

```
}
```

```
void ispis(char *v[])
```

```
{
```

```
    int i;
```

```
    for(i = 0; i < broj_rijeci; i++)
```

```
        puts(v[i]);
```

```
}
```

---

# Funkcija main

```
int main( ){  
  
    if((broj_rijeci = unos(v)) > 0){  
        sort(v, broj_rijeci);  
        ispis(v);  
    }  
    else  
        printf("Učitano je previše riječi (ili niti jedna).");  
  
    return 0;  
}
```

---

# Argumenti komandne linije

- Program koji želi koristiti argumente komandne linije mora deklarirati funkciju `main` s dva argumenta:

```
int main(int argc, char *argv[])  
{ ..... }
```

- `argc` je broj argumenata komandne linije koji su utipkani pri pokretanju programa. Ako nema argumenata komandne linije onda je `argc=1`.
- `argv` je polje pokazivača na argumente komandne linije. `argv[0]` uvijek pokazuje na string koji sadrži ime programa koji se izvršava; `argv[argc] = NULL`.



# Primjeri:

\$ocjene.exe ulaz izlaz 50

argc=4

argv[0] → "ocjene.exe"

argv[1] → "ulaz"

argv[2] → "izlaz"

argv[3] → "50"

argv[4] = NULL

```
int main (int argc, char *argv[ ]){  
    int i;  
    printf ("argc = %d\n\n", argc);  
    for (i = 0; i < argc; i++)  
        printf ("%d %s \n", i, argv[i]);  
    .....
```

# Funkcije atoi i atof

```
#include <stdlib.h>
```

```
int atoi(const char *string);
```

```
printf("%d\n", atoi(" 123"));           /* 123 */
```

```
printf("%d\n", atoi(" 12 3"));         /* 12 */
```

```
printf("%d\n", atoi(" 12abc3"));       /* 12 */
```

```
printf("%d\n", atoi("abc12 3"));       /* 0 */
```

```
double atof(const char *string);
```

```
printf("%g\n", atof(" 12.5a4c 3"));    /* 12.5 */
```

```
printf("%g\n", atof(" 12.5e4c 3"));    /* 125000 */
```

# Primjer: Implementacija atoi

- Funkcija treba:
  - preskočiti sve početne bjeline,
  - učitati predznak (ako ga ima),
  - i redom pročitati sve znamenke te ih prevesti u broj tipa `int`.

```
#include <ctype.h>
```

```
int f_atoi(const char s[])
```

```
{
```

```
    int i, n, sign;
```

```
    /* Preskakanje bjelina. */
```

```
    for(i = 0; isspace(s[i]); ++i);
```

# Funkcija `f_atoi`

```
sign = (s[i] == '-')? -1: 1;
```

```
if(s[i] == '+' || s[i] == '-') ++i;
```

```
/* Hornerov algoritam za broj. */
```

```
for(n = 0; isdigit(s[i]); ++i)
```

```
    n = 10*n+(s[i]-'0');
```

```
return sign*n;
```

```
}
```

- Zadatak. Napišite implementaciju funkcije `itoa` koja pretvara cijeli broj u string:

```
void f_itoa(int n, char s[]){...}
```

# Primjer:

```
#include <stdio.h>
#include <stdlib.h>

main (int argc, char *argv[])
{
    int *a;
    int i, n, zbroj;

    n = atoi(argv[1]);

    if((a = (int*) calloc (n, sizeof(int))) == 0){
        printf ("Nema dovoljno memorije\n");
        exit(1); }

    .....
```

# Pokazivač na funkciju

- Pokazivač na funkciju deklarira se kao

```
tip_pod (*ime)(tip_1 arg_1, tip_2 arg_2, ..., tip_n arg_n);
```

gdje je `ime` pokazivač na funkciju koja uzima `n` argumenata tipa `tip_1`, `tip_2`, ..., `tip_n` i vraća vrijednost tipa `tip_pod`.

- Npr:

```
int (*pf)(char c, double a);
```

```
double (*fn_ptr)(int, int);
```

- Funkcija koja vraća pokazivač tipa `double`:

```
double *fn_ptr(int, int); ≡ double *(fn_ptr(int, int));
```

# Primjer:

```
int main(){
    double hipotenuza (double, double);
    double (*pf) (double, double);
    double c;

    pf = hipotenuza;
    c = pf(3., 4.);
    printf ("Rezultat iznosi: %g\n", c);
    return 0;
}

double hipotenuza(double a, double b){
    return sqrt(a*a+b*b);
}
```

---

# Primjer: Trapezna formula

```
#include <stdio.h>
#include <math.h>
double integracija(double, double, double (*)(double));

int main(){
    printf("Sinus: %f\n", integracija(0, 1, sin));
    printf("Kosinus: %f\n", integracija(0, 1, cos));
    return 0;
}

double integracija(double a, double b, double (*f)(double)){
    return 0.5*(b-a)*((*f)(a)+(*f)(b));
}
```



# Složene deklaracije

- Pri interpretaciji deklaracije uzimaju se u obzir prioriteti pojedinih operatora. Ti prioriteti **mogu se promijeniti** upotrebom zagrada.
- Primjeri: **p** je:
  - `int *p[10];` /\* polje od 10 ptr na `int` \*/
  - `int *p(void);` /\* funkcija koja nema arg i vraća pokazivač na `int` \*/
  - `int p(char *a);` /\* funkcija koja uzima ptr na `char` i vraća `int` \*/
  - `int *p(char *a);` /\* funkcija koja uzima ptr na `char` i vraća ptr na `int` \*/
  - `int (*p)(char *a);` /\* ptr na funkciju koja uzima ptr na `char` i vraća `int` \*/

# Složene deklaracije (2)

- `int (*p(char *a))[10];` /\* funk. uzima ptr na char i vraća ptr na polje od 10 elt tipa int \*/
- `int p(char (*a)[]);` /\* funk. uzima ptr na polje znakova i vraća int \*/
- `int (*p)(char (*a)[]);` /\* ptr na funk. koja uzima ptr na polje znakova i vraća int \*/
- `int *(*p)(char (*a)[]);` /\* ptr na funk. koja uzima ptr na polje znakova i vraća ptr na int \*/
- `int *(*p[10])(char *a);` /\* polje 10 ptr na funk. koja uzima ptr na char i vraća ptr na int \*/