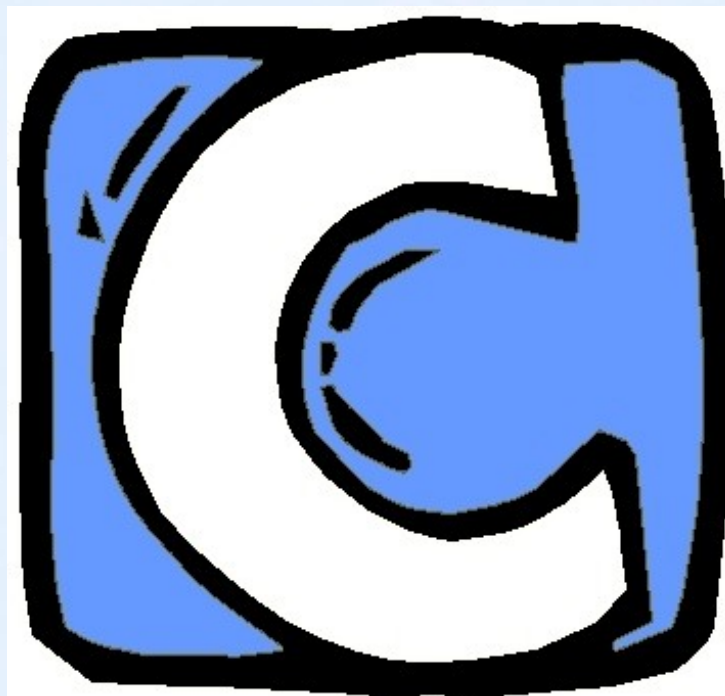


# ***Programski jezik C***



# ***Programski jezik C***

- Nizko nivojski jezik (blizu zbirnika).
- „Hipijevski“ jezik – vse je dovoljeno.
- Sintaksa podobna Javi (oz. obratno).
- Vhod in izhod.
- Kazalci.
- Delo s pomnilnikom.
- Funkcije in prenos parametrov.
- Strukture.

# ***Struktura programa***

zdravo.c

```
/*  
  Komentarji enaki kot v Javi.  
*/  
  
#include <stdio.h>  
  
// Glavni program v funkciji main().  
int main() {  
    printf("Zdravo OS\n");  
    return 0;  
}
```



# Prevajanje

- Prevajanje in povezovanje.

```
gcc -o zdravo zdravo.c
```

- Zagon.

```
./zdravo
```

Uporabna stikala

-std=c99

-c

-S

-O1

-O2

-O3

# ***Standardni izhod***


```
/*  
  Komentarji enaki kot v Javi.  
*/  
  
#include <stdio.h>  
  
// Glavni program v funkciji main().  
int main() {  
    printf("Zdravo OS\n");  
    return 0;  
}
```

# Standardni vhod

```
/*  
Namesto Scannerja imamo scanf(...)  
*/  
#include <stdio.h>
```

*Naslov kamor naj se število prebere.*

```
int main() {  
    int x;  
    scanf("%d", &x);  
    printf("Vpisali ste %d\n", x);  
    return 0;  
}
```





# Argumenti

```
int main(int argc, char* argv[]) {  
    printf("Stevilo argumentov je: %d\n", argc);  
    for (i=0; i<argc; i++) {  
        printf("%d: %s\n", i, argv[i]);  
    }  
    return 0;  
}
```

# Kazalci

- Kazalec je naslov v pomnilniku.
  - Odvisno od sistema: 32 oz. 64 bitni.
  - Vsebina: \*, naslov &.

```
#include <stdio.h>

int main() {
    int x = 5;
    int *y = &x;
    printf("Kazalec y: %p\n", y);
    printf("Vsebina y: %d\n", *y);
    return 0;
}
```



# ***Delo s pomnilnikom***

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

int main(int argc, char* argv[]) {
    int* a = malloc(2 * sizeof(int));
    a[0] = 1;
    a[1] = 2;
    printf("Tabela na naslovu %p\n", a);
    a++;
    printf("Drugi element na naslovu %p\n", a);
    a--;
    free(a);
    return 0;
}
```

# ***Delo s pomnilnikom***

- `void* malloc(size)`
  - Rezervira `size` bajtov, vrne naslov na rezervirani kos.
- `void* calloc(count, size)`
  - Kot `malloc(count * size)`, pomnilnik nastavi na 0.
- `void* realloc(ptr, size)`
  - Obstoječi prostor poveča oz. zmanjša, vsebina ostane nespremenjena.
- `void free(ptr)`
  - Sprosti pomnilnik na katerega kaže kazalec `ptr`.

# ***Funkcije***

- Sintaksa definicije funkcij podobna kot za Javanske metode.

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

```
int fib(int n) { ... }
```

```
int seštej(int x, int y) { ... }
```

```
int swap(int * x, int * y) { ... }
```



# ***Prenos parametrov***

- Prenos parametrov po vrednosti.

```
int fib(int n) {  
    if (n <= 0) return 0;  
    else if (n == 1) return 1;  
    int a = 1, b = 2;  
    while (n-- > 2) {  
        int c = a + b;  
        a = b;  
        b = c;  
    }  
    return a;  
}
```

# Prenos parametrov

- Prenos parametrov po *referenci*.

```
int swap(int * x, int * y) {  
    int t = *x;  
    *x = *y;  
    *y = t;  
}
```

- Pri klicu podamo naslov z &.

```
int fib(int n) {  
    if (n <= 0) return 0;  
    else if (n == 1) return 1;  
    int a = 1, b = 2;  
    while (n-- > 2) {  
        a += b;  
        swap(&a, &b);  
    }  
    return a;  
}
```

# Struktura

- V strukturo zapakiramo več podatkov.
- Podobno razredom v Javi.
- Vsi člani so *public*.

```
/*  
Struktura, ki hrani dve celi števili  
*/  
struct coord {  
    int x;  
    int y;  
};
```



# Struktura

```
struct coord {  
    int x;  
    int y;  
};  
  
int main(int argc, char* argv[]) {  
    // velikost strukture  
    printf("Velikost strukture je %d\\n",  
        sizeof(struct coord));  
  
    // uporaba strukture  
    struct coord c;  
    c.x = 1;  
    c.y = 2;  
  
    return 0;  
}
```

# *Struktura in tipi*

```
typedef struct coord {  
    int x;  
    int y;  
} coord;  
  
int main(int argc, char* argv[]) {  
    printf("Velikost strukture je %d\\n",  
        sizeof(coord));  
  
    coord c = { 1, 2 };  
  
    return 0;  
}
```

# Struktura

```
void obdelaj(coord * c) {  
    (*c).x = 123;  
    c->y = 456;  
}  
  
int main(int argc, char* argv[]) {  
    ...  
  
    coord c = { 1, 2 };  
    obdelaj(&c);  
  
    ...  
}
```



# ***Napredno prevajanje***

- Večji programi so seveda v več datotekah.
- Datoteke so medsebojno odvisne.
- Orodja za prevajanje (*build tools*).
- Najbolj znano je orodje `make`.
  - Java: npr. Ant, Maven.