Modularnost i enkapsulacija

- Implementacija na jeziku C: modul je jedan .c fajl

```
#define MaxStackSize 256
unsigned stack[MaxStackSize];  // Stack
unsigned sp = 0; // Stack pointer


int push (unsigned in) {
  if (sp==MaxStackSize) return -1; // Exception: stack full
  stack[sp++] = in;
  return 0;
}


int pop (unsigned* out) {
  if (sp==0) return -1; // Exception: stack empty
  *out = stack[--sp];
  return 0;
}
```

Modularnost i enkapsulacija

- Problem: interna implementacija steka (struktura, *stack* i *sp*) je dostupna ostalim delovima programa, pa stoga ti delovi programa mogu:
  - greškom da poremete tu strukturu i dovedu je u nekonzistentno stanje, npr:

```
sp = -5;
```

  - da se oslone na informaciju o načinu implementacije (postojanju niza *stack* i indeksa *sp*); ako iz bilo kog razloga imamo potrebu da promenimo tu implementaciju (npr. pređemo na neograničenu, dinamičku strukturu), promene će uticati na sve takve druge delove programa: oni se ili moraju menjati (teško i podložno greškama) ili neće raditi valjano
- Princip *sakrivanja informacija* (*information hiding*, David Parnas, 1972): svaka logička celina - modul programa, treba da ima jasno izdvojen
  - *interfejs* (*interface*): specifikaciju elemenata (struktura, tipova, operacija…) koje ostali delovi programa mogu da vide i pretpostavki na koje smeju da se oslone, i
  - *implementaciju* (*implementation*): interne delove (strukturu, ponašanje) koje drugi delovi programa ne smeju da vide, niti da se oslanjaju na pretpostavke o njoj
- Interfejsi treba da budu što opštiji, jednostavniji, kako bi sprege između delova softvera bile jednostavnije, labavije, lakše za kontrolu, a time ti delovi softvera nezavisniji i fleksibilniji
- *Enkapsulacija* (*encapsulation*) je programska tehnika koja podržava princip sakrivanja informacija (često se ova dva termina poistovećuju i koriste ravnopravno)