Modularnost i enkapsulacija

- Enkapsulacija na nivou modula u proceduralnom programiranju: razlika između
  - interfejsa modula; na jeziku C u fajlu zaglavlju (*header file, .h*):

```
/* File: stack.h */
int push (unsigned in);
int pop (unsigned* out);
```

  - implementacije tog modula; na jeziku C u fajlu zaglavlju (*.c file*):

```
/* File stack.c */
#include "stack.h"

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

int push (unsigned in) {
  …
}

int pop (unsigned* out) {
  …
}
```

- Da globalni identifikatori koji ne pripadaju interfejsu *ne bi bili dostupni* u drugim modulima, tj. da bi bili sakriveni od drugih modula, moraju se deklarisati tako da imaju tzv. *interno vezivanje* (*internal linking*):

```
static unsigned stack[MaxStackSize];  // Stack
static int sp = 0; // Stack pointer
```

Sa proceduralnog na OO programiranje: klase i objekti

- Nedostatak ovog rešenja: u programu imamo samo jedan ovakav stek, jednu instancu.
  Šta ako nam je potrebno više instanci ovakve strukture? Organizacija koja bi ovo omogućila:

```
/* File: stack.h */

#define MaxStackSize 256

struct Stack {
  unsigned stack[MaxStackSize];  // Stack
  int sp; // Stack pointer
};

void stack_init (Stack* this);
int stack_push (Stack* this, unsigned in);
int stack_pop (Stack* this, unsigned* out);

/* File stack.c */
#include “stack.h”

void stack_init (Stack* this) {
  this->sp = 0;
}

int stack_push (Stack* this, unsigned in) {
  if (this->sp==MaxStackSize) return -1; // Exception: stack full
  this->stack[this->sp++] = in;
  return 0;
}

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