Modularnost i enkapsulacija

- Rudiment enkapsulacije u proceduralnom programiranju: razlika između
  - deklaracije (potpisa, *signature*) nekog potprograma:

```
int push (unsigned in);
```

  - implementacije (potpune definicije tela) tog potprograma:

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

- Motivacija je izvorno banalna — nezavisno prevođenje modula: na mestu poziva, za generisanje koda pozivaoca, dovoljno je znati samo potpis potprograma, ne i celo telo; telo je potrebno na mestu gde se generiše kod za telo tog potprograma
- Opštije, ova ista rudimentarna tehnika predstavlja i podršku enkapsulaciji na nivou potprograma, kao osnovne gradivne jedinice proceduralnog programa
- Sličan princip potreban je i na krupnijem stepenu granularnosti - modulu

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
```