Sa proceduralnog na OO programiranje: klase i objekti

- Na jeziku C++, svaka (nestatička) funkcija članica klase poseduje lokalni, implicitno deklarisan pokazivač *this*; ovaj pokazivač je konstantan (ne može mu se promeniti vrednost)
- Tokom izvršavanja tela ovakve funkcije članice pokazivač *this* pokazuje na objekat za koga je ta funkcija pozvana
- Svaki neposredan pristup članu klase unutar ovakve funkcije smatra se implicitnim pristupom članu onog objekta na koga pokazuje *this* (uvek se može vršiti i eksplicitan pristup, što ponekad može poboljšati čitljivost ili razrešiti višeznačnost):

```
sp = 0;
```

je isto što i:

```
this->sp = 0;
```

- Privid da svaki objekat "poseduje svoju" funkciju članicu je zapravo posledica toga što se funkcija članica poziva za dati objekat, a članovima u tekstu tela funkcije može pristupati neposredno, bez navođenja objekta (podrazumeva se pristup članu onog objekta "čija" se funkcija izvršava). Ovaj privid stvara se jednostavnim opisanim mehanizmom implicitnog pristupa preko pokazivača *this*

Sa proceduralnog na OO programiranje: klase i objekti

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

#define MaxStackSize 256

struct Stack;

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

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


/* 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;
  this->stack[this->sp++] = in;
  return 0;
}

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

```c
// File: stack.h

const int MaxStackSize = 256;

class Stack {
public:
  Stack ();
  int push (unsigned in);
  int pop (unsigned* out);

private:
  unsigned stack[MaxStackSize];  // Stack
  int sp; // Stack pointer
};


// File stack.cpp
#include "stack.h"

Stack::Stack () {
  this->sp = 0;
}

int Stack::push (unsigned in) {
  if (this->sp==MaxStackSize) return -1;
  this->stack[this->sp++] = in;
  return 0;
}

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