THE “this” POINTER

Binding objects to member functions

Delroy A. Brinkerhoff

STACK EXAMPLE

STRUCT VERSION

```
const int SIZE = 100;

struct stack
{
        char    st[SIZE];
        int     sp;
};

stack   make_stack();
void    init_stack(stack* s);
void    push(stack* s, char data);
char    pop(stack* s);
int     size(stack* s);
char    peek(stack* s);
```

CLASS VERSION

```
class stack
{
    private:
        static const int SIZE = 100;

        char    st[SIZE];
        int     sp;

    public:
        stack() : sp(0) {}
        void    push(char data);
        char    pop();
        int     size();
        char    peek();
};
```

FUNCTION CALLS AND DEFINITIONS

PROGRAMMER WRITES

- `s.push('a');`

-
```
void push(stack* s, char data)
{
    if (s->sp < SIZE)
        s->st[s->sp++] = data;
    else
        cerr << "Overflow" << endl;
}
```

COMPILER GENERATES

- `s.push(&s, 'a');`

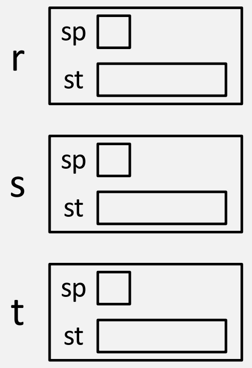
-
```
void stack::push(stack* this, char data)
{
    if (this->sp < SIZE)
        this->st[this->sp++] = data;
    else
        cerr << "Overflow" << endl;
}
```

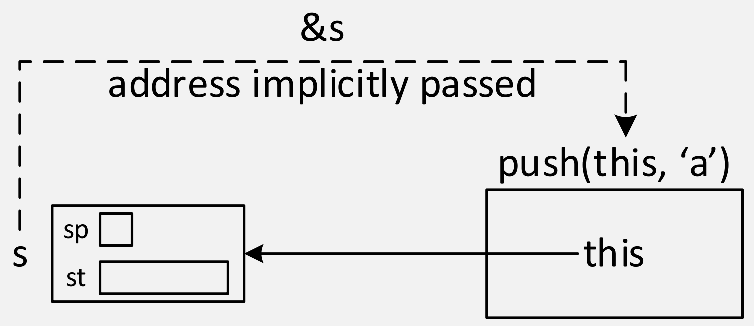
THE “this” POINTER

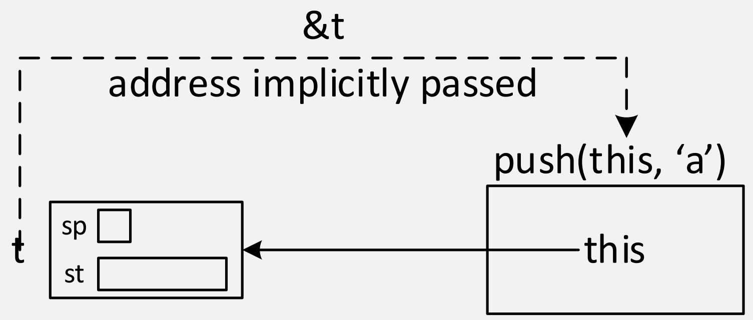
- The "this" pointer is an automatic local variable
- The compiler creates "this" in every non-static member function
- "this" stores the address of the object that calls the member function
- "this" implements the object-to-function binding

VISUALIZING THE this POINTER

```
stack  r;
stack  s;
stack  t;
s.push('a');
t.push('a');
```


r
sp
st
s
sp
st
t
sp
st


&s
address implicitly passed
push(this, ‘a’)
s
sp
st
this


&t
address implicitly passed
push(this, ‘a’)
t
sp
st
this