


Estruturas de Dados / Programação 2
Pilhas

Márcio Ribeiro
marcio@ic.ufal.br
twitter.com/marciomribeiro

Introduction

- When using an array, we can access any element we want

```
int items[] …

items[90];
items[22];
items[18];
```

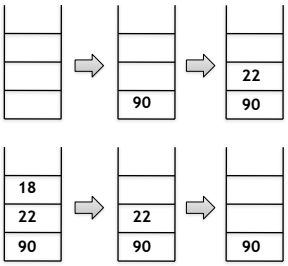
- Discussion: restricting access to some items make sense?

Yes! It does make sense!

Stack

Stack

- We can access only the last **added** element
- LIFO: "Last In, First Out"



Operations

- Push
  - Add an element to the stack top
- Pop
  - Remove the element from the stack top
- Peek
  - Returns (without removing) the top element

Abstract Data Type: Stack

Stack ADT

```
stack* create_stack();

void push(stack *stack, int item);

int pop(stack *stack);

int peek(stack *stack);

int is_empty(stack *stack);
```

Creating a Stack

```
#define MAX_STACK_SIZE 10

struct stack {
  int current_size;
  int items[MAX_STACK_SIZE];
};

stack* create_stack()
{
  stack *new_stack = (stack*) malloc(sizeof(stack));
  new_stack->current_size = 0;
  return new_stack;
}
```

Pushing items to the Stack

```
void push(stack *stack, int item)
{
  if (stack->current_size == MAX_STACK_SIZE) {
    printf("Stack overflow");
  } else {
    stack->items[stack->current_size++] = item;
  }
}
```

Stacks with Lists?!

```
struct node {
  int item;
  node *next;
}
```

```
struct stack {
  node *top;
}
```

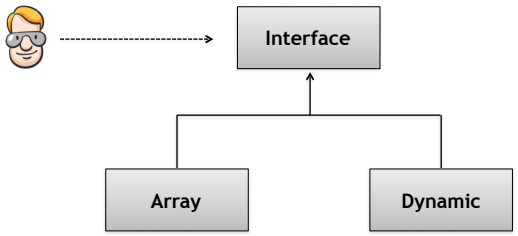
```
stack* create_stack()
{
  stack *new_stack = (stack*) malloc(sizeof(stack));
  new_stack->top = NULL;
  return new_stack;
}

void push(stack *stack, int item)
{
  node *new_top = (node*) malloc(sizeof(node));
  new_top->item = item;
  new_top->next = stack->top;
  stack->top = new_top;
}
```

What about these two following implementations?

```
void push(stack *stack, int item)
{
  if (stack->current_size == MAX_STACK_SIZE) {
    printf("Stack overflow");
  } else {
    stack->items[stack->current_size++] = item;
  }
}
```

```
void push(stack *stack, int item)
{
  node *new_top = (node*) malloc(sizeof(node));
  new_top->item = item;
  new_top->next = stack->top;
  stack->top = new_top;
}
```

Same contract! Clients access the interface (TAD)



Exercise: implement the pop and peek functions

```
int pop(stack *stack)
{
  if (is_empty(stack)) {
    printf("Stack underflow");
    return -1;
  } else {
    return stack->items[--stack->current_size];
  }
}

int peek(stack *stack)
{
  if (is_empty(stack)) {
    printf("Stack underflow");
    return -1;
  } else {
    return stack->items[stack->current_size - 1];
  }
}
```

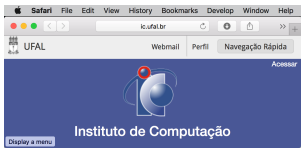
Exercise: decimal numbers to binary

```
while (number != 0)
{
  digit = number % 2;
  push(stack, digit);
  number = number / 2;
}

while (!is_empty(stack))
{
  printf("%d", pop(stack));
}
```

Exercise: palindrome

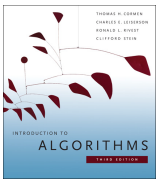
Applications

```
4 + 8 * (9 – 6)
 4 8 9 6 – * +
```




```
int check()
{
  if (m() && !n()) {
    …
  }
}
```

References



Chapter 10

Chapter 4