

Fundamental Concepts of Programming Languages

Abstract Data Types

Lecture 09

conf. dr. ing. Ciprian-Bogdan Chirilă

August 30, 2021

```
type stack(nr_max:integer)=class
  operations pop, push, top;
  var tab_st:array[1..nr_max] of integer;
  ind:integer;
```

```
function pop():integer;
begin
    if not empty() then
        ind:=ind-1;
        return tab_st[ind];
    else
        exception;
    end if;
end;
```

```
procedure push(x:integer);
begin
    if not overflow() then
        tab_st[ind]:=x;
        ind:=ind+1;
    else
        exception;
    end if;
end;
```

```
function top():integer;
begin
    if not empty() then
        return tab_st[ind-1];
    else
        exception;
    end if;
end;
```

```
function empty():boolean;  
begin  
    return ind=1;  
end;
```

```
function overflow():boolean;  
begin  
    return ind > nr_max;  
end;
```

```
begin
    ind:=1;
end;
```

```
var
    st1, st2 : stack(100);
    st3 : stack(55);
```

```
st1.push(10);
```

```
-----
```

```
st2.push(k);
```

```
-----
```

```
i:=st1.top();
```

```
-----
```

```
j:=st2.pop();
```



```
module st;
    export stack,pop,push,top,init;
type stack(nr_max:integer)=record
    tab_st:array[1..nr_max] of integer;
    ind:integer;
end record;
```

```
function pop(var s:stack):integer;
begin
    if not empty(s) then
        s.ind:=s.ind-1;
    return s.tab_st[s.ind];
    else exception;
    end if;
end;
```

```
procedure push(var s:stack;x:integer);
begin
    if not overflow(s) then
        s.tab_st[s.ind]:=x; s.ind:=s.ind+1;
    else exception;
    end if;
end;
```

```
function top(s:stack):integer;
begin
    if not empty(s) then
        return s.tab_st[s.ind-1]
    else exception;
    end if;
end;
```

```
procedure init(var s:stack);  
begin  
    s.ind:=1;  
end;
```

```
function empty(var s:stack):boolean;  
begin  
    return s.ind=1;  
end;
```

```
function overflow(var s:stack):boolean;
begin
    return s.ind>s.nr_max;
end;
```

```
var
    st1, st2:st.stack(100);
    st3:st.stack(50);
```

```
st.init(st1);
st.init(st2);
st.init(st3);

-----

st.push(st1,10);

-----

st.push(st2,k);

-----

i:=st.top(st1);

-----

j:=st.pop(st2);
```

```
type stack(type tip_el, nr_max:integer)=class
  operation pop, push, top;
  var var_st:array[1..nr_max]of tip_real;
  ind:integer;
```



```
function pop():tip_el;  
begin  
-----  
end;
```

```
procedure push(x:tip_el);  
begin  
-----  
end;
```

```
function top():tip_el;  
begin  
-----  
end;
```

```
function empty():boolean;  
begin  
-----  
end;
```

```
function overflow():boolean;
```

```
begin
```

```
-----
```

```
end;
```

```
begin
```

```
-----
```

```
end;
```

```
type t = _____
```

```
var
```

```
    st1 : stack(integer,100);
```

```
    st2 : stack(real,150);
```

```
    st3 : stack(t,50);
```

```
-----
```

```
st1.push(10);
```

```
st2.push(1.5);
```

```
module st(type tip_el);  
    export stack, pop, push, top, init;  
type stack(nr_max:integer) = record  
    tab_st:array[1..nr_max] of tip_el;  
    ind:integer;  
end record;
```

```
function pop(var s:stack) : tip_el;  
begin  
-----  
end;
```

```
procedure push(var s:stack; x:tip_el);  
begin  
-----  
end;
```

```
function top(var s:stack) : tip_el;  
begin  
-----  
end;
```

```
procedure init(var s:stack);  
begin  
-----  
end;
```

```
function empty(var s:stack) : boolean;  
begin  
-----  
end;
```

```
function overflow(var s:stack) : boolean;  
begin  
-----  
end;
```



```
begin
```

```
-----
```

```
end;
```

```
type t = _____;  
module st_int = st(integer);  
module st_real = st(real);  
module st_t = st(t);
```

```
var
```

```
    st1 : st_int.stack(100);  
    st2: st_real.stack(150);  
    st3: st_t.stack(50);
```

```
-----  
st_int.push(st1,10);
```

```
-----  
st_real.push(st2,1.5);  
-----
```

Bibliography