Report concerning:

**Stack Package**

By

**L. Dries**

dd. **16-08-2019**

Version: 1.00 Original version 16-08-2019 L. Dries

# Index

[1 Index 3](#_Toc16847902)

[2 Introduction 4](#_Toc16847903)

[3 General 5](#_Toc16847904)

[4 Internal structure 6](#_Toc16847905)

[5 Use of the package 7](#_Toc16847906)

[6 Listings 8](#_Toc16847907)

[6.1 Specification 8](#_Toc16847908)

[6.2 Body 10](#_Toc16847909)

# Introduction

A Stack is a first-in last-out memory structure. Such a structure is for instance meaningful in a case of error detection. By pushing the name of a routine on the stack at the start of a routine and removing it from the stack at the end of the routine any runtime error within the code routine can and not in a, from that routine called routine can be connected to that routine. In some operating systems such a stack is standard. Using such a device will use memory outside of the memory that is connected to the program by normal dimensions in your program. The standard garbage collection of ADA will clear that memory whenever not reachable anymore.

# General

The stack package is a package that creates s stack for one type of identifier. That can be an integer stack, an unbounded string stack, but also a stack of a type that you have created yourself. But you can only put identifiers of that type in it. More stacks of the same type or a different one are possible. To set a value in the stack is called pushing, getting one out is called popping. The last value pushed is always the first value popped.

# Internal structure

Internally every value pushed is automatically connected with a pointer to the one pushed before and the pointer to the value pushed is saved generally. When a value is popped you get the value pointed to by that external pointer, The value is then removed from the stack and the general pointer is than made to point to the value earlier pushed.

# Use of the package

Item is the type you use for the stack

procedure Push ( Typ : Item );

The procedure Push pushes an item on the stack

function Get return Item;

The function Get gets the top value of the stack without removing it from the stack

procedure Remove;

The procedure Remove removes the top value from the stack without reading it

function Pop return Item;

The function Pop gets the top value from the stack and removes it

procedure Clear;

The procedure Clear empties the complete stack

function Get\_Depth return integer;

The function Get\_Depth presents the number of levels within the stack at that moment.

To use the Stack package you have to declare it by:

package New\_Stack is new Stack(ID\_Line); to get a stack consisting od element of the type ID\_Line.

You can then use statement as New\_Stack.Pop(Line); where Line is of the type ID\_Line

# Listings

## Specification

-----------------------------------------------------------------------

-- Stack A package to create a general stack --

-- --

-- Copyright (C) 2019 L. Dries --

-- --

-- This library is free software; you can redistribute it and/or --

-- modify it under the terms of the GNU General Public --

-- License as published by the Free Software Foundation; either --

-- version 3 of the License, or (at your option) any later version. --

-- --

-- This library is distributed in the hope that it will be useful, --

-- but WITHOUT ANY WARRANTY; without even the implied warranty of --

-- MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU --

-- General Public License for more details. --

-- --

-- You should have received a copy of the GNU General Public --

-- License along with this library; if not, write to the --

-- Free Software Foundation, Inc., 59 Temple Place - Suite 330, --

-- Boston, MA 02111-1307, USA. --

-- --

-----------------------------------------------------------------------

generic

type Item is private;

Null\_Item : in Item;

package Stack is

type Block\_Stack is limited private;

type Stack\_Pointer is limited private;

-----------------------------------------------------------------------------

-- The procedure Pushes a value Typ of the type Item on the stack adding one

-- level to the stack

-----------------------------------------------------------------------------

procedure Push ( Typ : Item );

-----------------------------------------------------------------------------

-- The function reads the last pushed value from the stack but let the stack

-- intact.

-----------------------------------------------------------------------------

function Get return Item;

-----------------------------------------------------------------------------

-- The procedure reduces the level of the stack by one

-----------------------------------------------------------------------------

procedure Remove;

-----------------------------------------------------------------------------

-- The function pops a value of the type Item from the stack reducing the

-- level of the stack by one in the proces.

-----------------------------------------------------------------------------

function Pop return Item;

-----------------------------------------------------------------------------

-- The procedure Clears the stack completely

-----------------------------------------------------------------------------

procedure Clear;

-----------------------------------------------------------------------------

-- The function gets the Stack Depth

-----------------------------------------------------------------------------

function Get\_Depth return integer;

private

type Stack\_Pointer is access Block\_Stack;

type Block\_Stack is record

typ : Item;

previous : Stack\_Pointer := null;

next : Stack\_Pointer := null;

end record;

FirstSP : Stack\_Pointer := null;

LastSP : Stack\_Pointer := null;

Depth : integer := 0;

end Stack;

## Body

-----------------------------------------------------------------------

-- Stack A package to create a general stack --

-- --

-- Copyright (C) 2019 L. Dries --

-- --

-- This library is free software; you can redistribute it and/or --

-- modify it under the terms of the GNU General Public --

-- License as published by the Free Software Foundation; either --

-- version 3 of the License, or (at your option) any later version. --

-- --

-- This library is distributed in the hope that it will be useful, --

-- but WITHOUT ANY WARRANTY; without even the implied warranty of --

-- MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU --

-- General Public License for more details. --

-- --

-- You should have received a copy of the GNU General Public --

-- License along with this library; if not, write to the --

-- Free Software Foundation, Inc., 59 Temple Place - Suite 330, --

-- Boston, MA 02111-1307, USA. --

-- --

-----------------------------------------------------------------------

package body Stack is

procedure Push ( Typ : Item ) is

begin

if FirstSP = null then

FirstSP := new Block\_Stack;

LastSP := FirstSP;

else

declare

NewSP : Stack\_Pointer;

begin

NewSP := FirstSP;

while NewSP.next /= null loop

NewSP := NewSP.next;

end loop;

NewSP.next := new Block\_Stack;

NewSP.next.previous := NewSP;

LastSP := NewSP.next;

end;

end if;

Depth := Depth + 1;

LastSP.typ := Typ;

end Push;

function Get return Item is

begin

if LastSP = null then

return Null\_Item;

else

return LastSP.typ;

end if;

end Get;

procedure Remove is

OldSP : Stack\_Pointer;

begin

OldSP := LastSP;

if FirstSP = LastSP then

FirstSP := null;

end if;

if LastSP /= null then

LastSP := OldSP.previous;

if LastSP /= null then

LastSP.next := null;

end if;

Depth := Depth - 1;

if Depth < 0 then

Depth := 0;

end if;

end if;

end Remove;

function Pop return Item is

value : Item;

begin

value := Get;

Remove;

return value;

end Pop;

procedure Clear is

begin

while LastSP /= null loop

Remove;

end loop;

end Clear;

function Get\_Depth return integer is

begin

return Depth;

end;

end Stack;