Report concerning:

**Buffer Package**

By

**L. Dries**

dd. **16-08-2019**

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

Version: 1.10 Problem with Renumber solved. 16-08-2019 L. Dries

Insert\_Buffer now returns the value of the buffer 16-04-2020 L. Dries

pointer of the inserted buffer

# Index

[1 Index 3](#_Toc16848084)

[2 Introduction 4](#_Toc16848085)

[3 General 5](#_Toc16848086)

[4 Internal structure 6](#_Toc16848087)

[5 Use of the package 7](#_Toc16848088)

[6 Listings 8](#_Toc16848089)

[6.1 Specification 8](#_Toc16848090)

[6.2 Body 10](#_Toc16848091)

# Introduction

A Buffer is a random access memory structure. Such a structure is for instance meaningful in a case of need for unbounded array of some type. 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 buffer package is a package that creates buffer for one type of identifier. That can be an integer buffer, an unbounded string buffer, but also a buffer of a type that you have created yourself. But you can only put identifiers of that type in it. More buffers of the same type or a different one are possible. To set a value in the pipe is called pushing, getting one out is called popping. The first 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 if it is the first element in the pipe 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 pipe and the general pointer is than made to point to the value next pushed.

# Use of the package

Item is the generic type you use for the buffer

procedure Set\_Buffer ( inp : item; nr : integer := 0);

Set\_Buffer sets a value in the buffer at position nr. If nr = 0 the position will be the next element at the bottom of the buffer. If nr > 0 the element nr will be replaced. “0” is the default value for nr.

procedure Insert\_Buffer ( inp : item; bp: in out integer; Dir: Buffer\_Dir := after);

Insert\_Buffer will insert a new element before or after the bufferpointer bp. “after” is the default value for Dir. The buffers will be renumbered. Afterwards bp will contain the buffer nunber of the inserted buffer

function Get\_Buffer( nr : integer) return Item;

Get\_Buffer return the value of Item at bufferpointer nr. The buffers will be renumbered.

procedure Remove\_Buffer( nr : integer);

This Removes the buffer at that position. The buffers will be renumbered.

procedure Clear\_Buffer;

The complete buffer will be cleared

function Length return integer;

his reurns the number of elements in the buffer

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

package New\_Buffer is new Buffer(ID\_Line); to get a buffer consisting of elements of the type ID\_Line.

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

# Listings

## Specification

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

-- Package to Create Random Access Buffers for various types --

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

-- --

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

-- under terms of the GNU General Public License as published by the Free --

-- Software Foundation; either version 3, 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 MERCHAN- --

-- TABILITY or FITNESS FOR A PARTICULAR PURPOSE. --

-- --

-- As a special exception under Section 7 of GPL version 3, you are granted --

-- additional permissions described in the GCC Runtime Library Exception, --

-- version 3.1, as published by the Free Software Foundation. --

-- --

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

-- a copy of the GCC Runtime Library Exception along with this program; --

-- see the files COPYING3 and COPYING.RUNTIME respectively. If not, see --

-- <http://www.gnu.org/licenses/>. --

-- --

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

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

-- Buffer.ads Version 1.00 dd. 14-01-2019 Author L Dries --

------------------------------------------------------------------------------1

generic

type Item is private;

package Buffer is

type Block\_Buffer is limited private;

type Buffer\_Pointer is limited private;

type Buffer\_Dir is (after, before);

BUFFER\_ITEM\_OUT\_OF\_RANGE : exception;

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

-- Fill a Buffer Element After the mentioned pointer

-- where the standard is at the end of the buffer

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

procedure Set\_Buffer ( inp : item; nr:integer := 0);

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

-- Fill a Buffer Element before or after then mentioned element nr

-- where the standard is after the mentioned element

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

procedure Insert\_Buffer ( inp : item; bp: in out integer; Dir: Buffer\_Dir := after);

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

-- Get the value of the mentioned element

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

function Get\_Buffer( nr : integer) return Item;

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

-- Remove a Buffer Element

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

procedure Remove\_Buffer( nr : integer);

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

-- Clear the complete buffer

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

procedure Clear\_Buffer;

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

-- Find the nr of elemenets in the buffer

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

function Length return integer;

private

type Buffer\_Pointer is access Block\_Buffer;

type Block\_Buffer is record

nr : integer;

buf : Item;

previous : Buffer\_Pointer := null;

next : Buffer\_Pointer := null;

end record;

FirstBuffer : Buffer\_Pointer := null;

LastBuffer : Buffer\_Pointer := null;

Nr\_Items : integer := 0;

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

-- Renumber all the elements in the buffer

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

procedure Renumber;

end Buffer;

## Body

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

-- Package to Create Random Accress Buffers for various types --

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

-- --

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

-- under terms of the GNU General Public License as published by the Free --

-- Software Foundation; either version 3, 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 MERCHAN- --

-- TABILITY or FITNESS FOR A PARTICULAR PURPOSE. --

-- --

-- As a special exception under Section 7 of GPL version 3, you are granted --

-- additional permissions described in the GCC Runtime Library Exception, --

-- version 3.1, as published by the Free Software Foundation. --

-- --

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

-- a copy of the GCC Runtime Library Exception along with this program; --

-- see the files COPYING3 and COPYING.RUNTIME respectively. If not, see --

-- <http://www.gnu.org/licenses/>. --

-- --

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

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

-- Buffer.adb Version 1.00 dd. 14-01-2019 Author L Dries --

------------------------------------------------------------------------------1

package body Buffer is

procedure Set\_Buffer( inp : item; nr:integer := 0) is

El : Buffer\_Pointer := LastBuffer;

El1 : Buffer\_Pointer;

ok : boolean;

begin

if nr = 0 then

if El = null then

El := new Block\_Buffer;

FirstBuffer := El;

El.next := null;

LastBuffer := El;

El.previous := null;

else

El1 := El;

El := new Block\_Buffer;

El1.next := El;

LastBuffer := El;

El.next := null;

El.previous := El1;

end if;

El.buf := inp;

Renumber;

else

ok := false;

El1 := FirstBuffer;

while not ok loop

if El1.nr = nr then

ok := true;

else

El1 := El1.next;

if El1 = null then

raise BUFFER\_ITEM\_OUT\_OF\_RANGE;

end if;

end if;

end loop;

El1.buf := inp;

end if;

end;

procedure Insert\_Buffer ( inp : item; bp: in out integer; Dir: Buffer\_Dir := after) is

El : Buffer\_Pointer := FirstBuffer;

El1 : Buffer\_Pointer;

begin

if Nr\_Items < bp then

Set\_Buffer(inp);

else

while bp /= El.nr loop

El := El.next;

end loop;

El1 := new Block\_Buffer;

El1.buf := inp;

if Dir = after then

El1.previous := El;

El1.next := El.next;

El.next:= El1;

if El1.next /= null then

El1.next.previous := El1;

else

LastBuffer := El1;

end if;

else

El1.previous := El.previous;

El1.next := El;

El.previous := El1;

if El1.previous /= null then

El1.previous.next := El1;

else

FirstBuffer := El1;

end if;

end if;

Renumber;

end if;

end;

function Get\_Buffer( nr : integer) return Item is

El : Buffer\_Pointer := FirstBuffer;

X : item;

begin

if nr <= Nr\_Items then

while nr > El.nr loop

El := El.next;

end loop;

X := El.buf;

end if;

return X;

end;

procedure Remove\_Buffer( nr : integer) is

El : Buffer\_Pointer;

Elb : Buffer\_Pointer;

Ela : Buffer\_Pointer;

begin

if nr <= Nr\_Items then

El := FirstBuffer;

while El.nr /= nr loop

El := El.next;

end loop;

Elb := El.previous;

Ela := El.next;

if Elb /= null then

Elb.next := Ela;

else

Firstbuffer := El.next;

end if;

if Ela /= null then

Ela.previous := Elb;

else

LastBuffer := El.previous;

end if;

if Nr\_Items = 1 then

Nr\_Items := 0;

FirstBuffer := null;

LastBuffer := null;

else

Renumber;

end if;

end if;

end;

procedure Clear\_Buffer is

El : Buffer\_Pointer;

El1 : Buffer\_Pointer;

begin

El := FirstBuffer;

while El /= null loop

El1 := El.next;

El.previous := null;

El.next := null;

El := El1;

end loop;

El := null;

FirstBuffer := null;

LastBuffer := null;

Nr\_Items := 0;

end;

function Length return integer is

begin

return nr\_Items;

end;

procedure Renumber is

El : Buffer\_Pointer;

n : integer := 0;

begin

El := FirstBuffer;

while El /= null loop

n := n + 1;

El.nr := n;

El:= El.next;

end loop;

Nr\_Items := n;

end;

end Buffer;