西安交通大學
XI'AN JIAOTONG UNIVERSITY

2022-2023 学年第二学期

《编译器设计专题实验》

实验报告 2

学　　院：电信学部

班　　级：[redacted]

学　　号：[redacted]

姓　　名：[redacted]

二〇二三年 四月

目录

《实验 2-使用 makefile 和现成代码实现堆栈机》

一、实验内容（必做）

1. 使用 Cool 语言实现堆栈机，要求如下：

Int	将整数 int 压入栈顶
+	将字符‘+’压入栈顶
s	将字符‘s’压入栈顶
e	计算栈顶表达式的值（见下文详细介绍）
d	展示栈的内容，从栈顶开始，每个元素占一行
x	退出堆栈机程序

e 命令行为取决于堆栈的内容：

1）如果“+”在栈顶，那么“+”出栈，下面两个整数弹出且相加，结果被压入栈；

2）如果“s”在栈顶，那么“s”弹出，下面两个元素在栈内交换；

3）如果一个整数在栈顶，且栈为空，那么堆栈保持不变

不考虑错误输入的情况。

2. 编写 makefile 文件编译运行代码。

二、实验内容（选做）

使用 example 里面的 life.cl 完成一个可交互的游戏：首先，观察游戏输入输出，描述出细胞状态转换图；其次，修改代码使得游戏可以动态调整初始细胞数量；最后，发散式修改相应功能。

三、实验结果

(1) 堆栈机

```
GuoSongjian(Thu Apr 20 19:32:11):~/compiler_exp/exp2$ make
Compiling...
Compile OK!
Start testing...
SPIM Version 6.5 of January 4, 2003
Copyright 1990-2003 by James R. Larus (larus@cs.wisc.edu).
All Rights Reserved.
See the file README for a full copyright notice.
Loaded: ../cool/cool/lib/trap.handler
>1
>+
>2
>s
>d
s
2
+
1
>e
>e
>d
3
>x
COOL program successfully executed
Test finished!
Clean finished!
```

(2) 生命游戏

```
GuoSongjian(Fri Apr 21 16:30:17):~/compiler_exp/exp2_extend$ make
Compiling...
Compile OK!
Start testing...
SPIM Version 6.5 of January 4, 2003
Copyright 1990-2003 by James R. Larus (larus@cs.wisc.edu).
All Rights Reserved.
See the file README for a full copyright notice.
Loaded: ../cool/cool/lib/trap.handler
Welcome to the Game of Life.

Would you like to choose a background pattern?
Please use lowercase y or n for your answer [n]: y

Please input row: 5

Please input col: 6

#0>XXXXXX
#1>XX  XX
#2>  X
#3>X  X
#4> XX  X

XXXXXX
XX  XX
  X
X  X
 XX  X
```

```
Would you like to continue with the next generation?
Please use lowercase y or n for your answer [y]:

Increasing heap...

X-XX-X
X----X
X-XXX-
---X--
-XX---

Would you like to continue with the next generation?
Please use lowercase y or n for your answer [y]:

Increasing heap...

-X--X-
X----X
-XXXX-
----X-
--X---

Would you like to continue with the next generation?
Please use lowercase y or n for your answer [y]:

Increasing heap...

------
X----X
-XXXXX
-X--X-
------
```

```
Would you like to continue with the next generation?
Please use lowercase y or n for your answer [y]:

Increasing heap...

------
-XXX-X
XXXX-X
-X--XX
------

Would you like to continue with the next generation?
Please use lowercase y or n for your answer [y]: n


Would you like to choose a background pattern?
Please use lowercase y or n for your answer [n]: n
COOL program successfully executed
Test finished!
Clean finished!
GuoSongjian(Fri Apr 21 16:33:05):~/compiler_exp/exp2_extend$
```

状态转换：每个细胞的下一状态取决于其当前状态及周围 8 个细胞的状态。当细胞为 on 时，若周围有 2 或 3 个细胞为 on，则保持 on 不变，否则转换为 off；当细胞为 off 时，若周围有 3 个细胞为 on，则转换为 on，否则保持 off 不变。

对源代码的修改：原始代码只能从给定的初始状态中选择一个进行迭代，修改后用户可以自定义网格的行数与列数，并指定每一个网格位置的细胞状态是 on（输入 X）还是 off（输入空格）。

四、源代码

(1) A2I 类

```
-- A2I class implements the conversion between integers and strings

class A2I {

   -- c2i converts a single character to integers 0-9
   c2i(char : String) : Int {
      if char = "0" then 0 else
      if char = "1" then 1 else
      if char = "2" then 2 else
      if char = "3" then 3 else
      if char = "4" then 4 else
      if char = "5" then 5 else
      if char = "6" then 6 else
      if char = "7" then 7 else
      if char = "8" then 8 else
      if char = "9" then 9 else
      { abort(); 0; }  -- the 0 is needed to satisfy the
typechecker
      fi fi fi fi fi fi fi fi fi fi
   };

   -- i2c converts integers 0-9 to a single character
   i2c(i : Int) : String {
      if i = 0 then "0" else
      if i = 1 then "1" else
      if i = 2 then "2" else
      if i = 3 then "3" else
      if i = 4 then "4" else
      if i = 5 then "5" else
      if i = 6 then "6" else
      if i = 7 then "7" else
      if i = 8 then "8" else
      if i = 9 then "9" else
```

```
       { abort(); ""; }  -- the "" is needed to satisfy the
typechecker
       fi fi fi fi fi fi fi fi fi fi
   };

   (*
       a2i converts an ASCII string into an integer.
       The empty string is converted to 0.
       Signed and unsigned strings are handled.
       The method aborts if the string does not represent an
integer.
       Very long strings of digits produce strange answers
       because of arithmetic overflow.
   *)
   a2i(s : String) : Int {
       if s.length() = 0 then 0 else
       if s.substr(0,1) = "-" then ~a2i_aux(s.substr(1,s.length()-
1)) else
       if s.substr(0,1) = "+" then a2i_aux(s.substr(1,s.length()-
1)) else
          a2i_aux(s)
       fi fi fi
   };

   -- a2i_aux converts the usigned portion of the string
   a2i_aux(s : String) : Int {
       (let int : Int <- 0 in {
           (let j : Int <- s.length() in
               (let i : Int <- 0 in
                   while i < j loop {
                       int <- int * 10 + c2i(s.substr(i,1));
                       i <- i + 1;
                   } pool
               )
           );
           int;
           }
       )
   };

   (*
       i2a converts an integer to a string.
       Positive and negative numbers are handled correctly.
   *)
```

```
    i2a(i : Int) : String {
        if i = 0 then "0" else
        if 0 < i then i2a_aux(i) else
            "-".concat(i2a_aux(i * ~1))
        fi fi
    };

    -- i2a_aux converts the usigned integer using recursion
    i2a_aux(i : Int) : String {
        if i = 0 then "" else
            (let next : Int <- i / 10 in
                i2a_aux(next).concat(i2c(i - next * 10))
            )
        fi
    };

};
```

(2) List 类

```
-- List class implements the empty list

class List {

    isNil() : Bool { true };

    head() : String { { abort(); ""; } };

    tail() : List { { abort(); self; } };

    cons(i : String) : List { (new Cons).init(i, self) };

};
```

(3) Cons 类

```
-- Cons class implements the non-empty list

class Cons inherits List {

    first : String;
    rest : List;

    isNil() : Bool { false };

    head() : String { first };
```

```
    tail() : List { rest };

    init(head : String, next : List) : List {
        {
            first <- head;
            rest <- next;
            self;
        }
    };
};
```

(4) Main 类

```
class Main inherits IO {

    stack : List;

    newline() : Object { out_string("\n") };

    prompt() : String { { out_string(">"); in_string(); } };

    display_stack(s : List) : Object {
        if s.isNil() then out_string("") else {
            out_string(s.head());
            out_string("\n");
            display_stack(s.tail());
            }
        fi
    };

    main() : Object {
        (let z: A2I <- new A2I, stack: List <- new List, flag: Bool
<- true in
            while flag loop
                (let s : String <- prompt() in
                    if s = "x" then flag <- false else
                    if s = "d" then display_stack(stack) else
                    if s = "e" then {
                        if stack.isNil() then out_string("") else
                        if stack.head() = "+" then {
                            stack <- stack.tail();
                            (let a : Int <- new Int, b : Int <- new Int
in {
                                a <- z.a2i(stack.head());
```

```
                            stack <- stack.tail();
                            b <- z.a2i(stack.head());
                            stack <- stack.tail();
                            a <- a + b;
                            stack <- stack.cons(z.i2a(a));
                            }
                        );
                    } else
                    if stack.head() = "s" then {
                        stack <- stack.tail();
                        (let a : String <- new String, b : String
<- new String in {
                            a <- stack.head();
                            stack <- stack.tail();
                            b <- stack.head();
                            stack <- stack.tail();
                            stack <- stack.cons(a);
                            stack <- stack.cons(b);
                            }
                        );
                    } else out_string("")
                    fi fi fi;
                } else stack <- stack.cons(s)
                fi fi fi
            )
          pool
       )
   };

};
```

(5) makefile

```
CLASSDIR = ../cool/cool

FILES = $(wildcard *.cl)
TARGET = stack.s

all: source test clean

source: ${FILES}
   @echo "\e[32;1mCompiling...\e[33;0;m"
   @${CLASSDIR}/bin/coolc ${FILES} -o ${TARGET}
   @echo "\e[32;1mCompile OK!\e[33;0;m"
```

```
test: ${TARGET}
   @echo "\e[32;1mStart testing...\e[33;0m"
   @${CLASSDIR}/bin/spim -trap_file ${CLASSDIR}/lib/trap.handler -
file ${TARGET}
   @echo "\e[32;1mTest finished!\e[33;0m"

clean:
   @rm ${TARGET}
   @echo "\e[32;1mClean finished!\e[33;0m"
```

(6) life.cl

```
class Board inherits IO {

    rows: Int;
    columns: Int;
    board_size: Int;

    board_init(start: String, row: Int, col: Int): SELF_TYPE {
        {
            rows <- row;
            columns <- col;
            board_size <- start.length();
            self;
        }
    };

};


class CellularAutomaton inherits Board {

    population_map: String;

    init(map: String, row: Int, col: Int): SELF_TYPE {
        {
            population_map <- map;
            board_init(map, row, col);
            self;
        }
    };

    print(): SELF_TYPE {
        (let i: Int <- 0, num: Int <- board_size in {
            out_string("\n");
```

```
            while i < num loop {
                out_string(population_map.substr(i,columns));
                out_string("\n");
                i <- i + columns;
            } pool;
            out_string("\n");
            self;
        } )
    };

    num_cells(): Int { population_map.length() };

    cell(position: Int): String {
        if board_size - 1 < position then
         " "
        else
            population_map.substr(position, 1)
        fi
    };

    north(position: Int): String {
        if (position - columns) < 0 then
            " "
        else
            cell(position - columns)
        fi
    };

    south(position: Int): String {
        if board_size < (position + columns) then
            " "
        else
            cell(position + columns)
        fi
    };

    east(position: Int): String {
        if (((position + 1) / columns ) * columns) = (position + 1)
then
            " "
        else
            cell(position + 1)
        fi
    };
```

```
   west(position: Int): String {
      if position = 0 then
         " "
      else
         if ((position / columns) * columns) = position then
            " "
         else
            cell(position - 1)
      fi fi
   };

   northwest(position: Int): String {
      if (position - columns) < 0 then
         " "
      else
         if ((position / columns) * columns) = position then
            " "
         else
            north(position - 1)
      fi fi
   };

   northeast(position: Int): String {
      if (position - columns) < 0 then
         " "
      else
         if (((position + 1) / columns ) * columns) = (position +
1) then
            " "
         else
            north(position + 1)
      fi fi
   };

   southeast(position: Int): String {
      if board_size < (position + columns) then
         " "
      else
         if (((position + 1) / columns ) * columns) = (position +
1) then
            " "
         else
            south(position + 1)
```

```
        fi fi
    };

    southwest(position: Int): String {
        if board_size < (position + columns) then
            " "
        else
            if ((position / columns) * columns) = position then
                " "
            else
                south(position - 1)
        fi fi
    };

    neighbors(position: Int): Int {
        {
            if north(position) = "X" then 1 else 0 fi
            + if south(position) = "X" then 1 else 0 fi
            + if east(position) = "X" then 1 else 0 fi
            + if west(position) = "X" then 1 else 0 fi
            + if northeast(position) = "X" then 1 else 0 fi
            + if northwest(position) = "X" then 1 else 0 fi
            + if southeast(position) = "X" then 1 else 0 fi
            + if southwest(position) = "X" then 1 else 0 fi;
        }
    };

    (*  A cell will live if 2 or 3 of it's neighbors are alive. It
dies
    otherwise. A cell is born if only 3 of it's neighbors are
alive.  *)

    cell_at_next_evolution(position: Int): String {
        if neighbors(position) = 3 then
            "X"
        else
            if neighbors(position) = 2 then
                if cell(position) = "X" then
                    "X"
                else
                    "-"
                fi
            else
                "-"
```

```
       fi fi
   };

   evolve(): SELF_TYPE {
       (let position: Int <- 0, num: Int <- num_cells(), temp:
String in {
          while position < num loop {
             temp <-
temp.concat(cell_at_next_evolution(position));
             position <- position + 1;
          } pool;
          population_map <- temp;
          self;
       } )
   };

   (*  This is where the background pattern is detremined by the
user.  *)

   option(row: Int, col: Int): String {
       {
          out_string("\n");
          (let i: Int <- 0, string: String <- "" in {
             while i < row loop {
                out_string("#");
                out_int(i);
                out_string(">");
                string <- string.concat(in_string().substr(0,
col));
                i <- i + 1;
             } pool;
             string;
          } );
       }
   };

   choose_row(): Int {
       {
          out_string("\nPlease input row: ");
          in_int();
       }
   };

   choose_col(): Int {
```

```
       {
          out_string("\nPlease input col: ");
          in_int();
       }
   };

   prompt() : Bool {
       (let ans : String in {
          out_string("Would you like to continue with the next
generation? \n");
          out_string("Please use lowercase y or n for your answer
[y]: ");
          ans <- in_string();
          out_string("\n");
          if ans = "n" then false else true fi;
       } )
   };

   prompt2() : Bool {
       (let ans : String in {
          out_string("\n");
          out_string("Would you like to choose a background
pattern? \n");
          out_string("Please use lowercase y or n for your answer
[n]: ");
          ans <- in_string();
          if ans = "y" then true else false fi;
       } )
   };

};


class Main inherits CellularAutomaton {

   cells: CellularAutomaton;

   main(): SELF_TYPE {
       (let continue: Bool, choice: String in {
          out_string("Welcome to the Game of Life.\n");
          while prompt2() loop {
             continue <- true;
             (let row: Int <- choose_row(), col: Int <-
choose_col() in {
```

```
                choice <- option(row, col);
                cells <- (new CellularAutomaton).init(choice, row,
col);
                cells.print();
             } );
             while continue loop
                if prompt() then { cells.evolve();
cells.print(); }
                else continue <- false fi
             pool;
          } pool;
          self;
       } )
   };

};
```