《软件质量保证与测试》
实验报告

姓名:	陈楷文
学号:	202232110214
班级:	软件工程 (中外合作办学)222
实验名称:	黑盒测试-边界值测试法

实验一: 黑盒测试-边界值测试法

陈楷文

April 30, 2024

1 [实验环境]

- 操作系统:Linux
- 程序设计语言:Rust
- 脚本设计语言:Bash

2 [实验类型]

黑盒测试
边界值测试

3 [实验目的]

- 认识黑盒测试方法中边界值分析测试法原理
- 掌握黑盒测试方法中边界值分析测试法过程

4 [实验内容]

1. 编写三角形程序
2. 编写三角形程序测试脚本
3. 编写 NextDay 程序
4. 编写 NextDay 程序测试脚本
5. 运行测试
6. 分析测试结果

5 [问题描述]

5.1 三角形问题

问题描述：三角形问题接受三个整数，a、b 和 c 作为输入，用作三角形的边。程序的输出是由这三条边确定的三角形类型：等边三角形、等腰三角形、不等边三角形或非三角形。

作为输入：三角形的三条边必须满足如下条件：

C1：1<=a<=100

C2：1<=b<=100

C3：1<=c<=100

C4：a<b+c

C5：b<a+c

C6：c<a+b

5.2 NextDay 问题

问题描述：NextDate 是一个由三个变量（月份、日期和年份）的函数。函数返回输入日期后边的那个日期。

作为输入：变量月份、日期和年都具有整数值，满足以下条件。

C1：1<= 月份 <=12

C2：1<= 日期 <=31

C3：1912<= 年 <=2050

6 [算法描述]

6.1 三角形程序

use std::env;：导入了 env 模块，用于处理命令行参数。
fn main() ... ：程序的入口函数。
let args: Vec<String> = env::args().collect();：将命令行参数收集到一个字符串向量中。
let (a, b, c) = match parse_arguments(&args) ... ：调用 parse_arguments 函数解析命令行参数，并将解析结果绑定到变量 (a, b, c) 中。
fn parse_arguments(args: &[String]) -> Result<(u32, u32, u32), String> ... ：解析命令行参数的函数。它接受一个字符串切片作为参数，返回一个 Result 枚举，其中 Ok 包含三个边长，Err 包含错误信息。
for i in 1..args.len() ... ：遍历命令行参数。
match args[i].as_str() ... ：匹配当前命令行参数的字符串值。
-a, -b, -c：检查是否遇到了命令行参数-a、-b 或-c。
Some(value)：如果解析成功，返回一个包含解析后的值的 Some 枚举。
Some(args[i + 1].parse().map_err(|_| " 边长 a 必须是一个有效的整数")?)?：将下一个参数解析为整数，如果解析失败，则返回一个包含错误信息的 Err 枚举。
is_triangle 函数：检查三条边是否能构成三角形。
输出结果：根据判断的结果输出对应的信息，例如等边三角形、等腰三角形、不等边三角形或非三角形。

6.2 NextDay 程序

首先，程序使用了 std::env 模块来获取命令行参数。通过 env::args() 函数获取参数列表，并将其收集到一个 Vec<String> 类型的变量 args 中。
然后，程序定义了 main() 函数作为程序的入口点。在 main() 函数中，它遍历命令行参数列表，解析出年份、月份和日期，并存储到相应的变量中。
接下来，程序调用了 is_valid_date() 函数来检查输入的日期是否有效。这个函数会检查年份是否在 1912 到 2050 之间，月份是否在 1 到 12 之间，日期是否在 1 到 31 之间。如果日期无效，程序会打印错误消息并退出。
如果日期有效，程序就会调用 next_date() 函数来计算下一个日期。这个函数会根据当前日期的年、月、日来计算下一个日期，并考虑闰年和月底的情况。

最后，程序打印出计算得到的下一个日期。

7 [测试案例]

三角形测试数据

a	b	c	测试输出	预期输出
1	50	50	这是一个等腰三角形。	这是一个等腰三角形。
2	50	50	这是一个等腰三角形。	这是一个等腰三角形。
99	50	50	这是一个等腰三角形。	这是一个等腰三角形。
100	50	50	这三条边无法构成三角形。	这三条边无法构成三角形。
50	1	50	这是一个等腰三角形。	这是一个等腰三角形。
50	2	50	这是一个等腰三角形。	这是一个等腰三角形。
50	99	50	这是一个等腰三角形。	这是一个等腰三角形。
50	100	50	这三条边无法构成三角形。	这三条边无法构成三角形。
50	50	1	这是一个等腰三角形。	这是一个等腰三角形。
50	50	2	这是一个等腰三角形。	这是一个等腰三角形。
50	50	99	这是一个等腰三角形。	这是一个等腰三角形。
50	50	100	这三条边无法构成三角形。	这三条边无法构成三角形。
50	50	50	这是一个等边三角形。	这是一个等边三角形。
0	50	50	这三条边无法构成三角形。	这三条边无法构成三角形。
1	50	50	这是一个等腰三角形。	这是一个等腰三角形。
2	50	50	这是一个等腰三角形。	这是一个等腰三角形。
99	50	50	这是一个等腰三角形。	这是一个等腰三角形。
100	50	50	这三条边无法构成三角形。	这三条边无法构成三角形。
101	50	50	这三条边无法构成三角形。	这三条边无法构成三角形。
50	0	50	这三条边无法构成三角形。	这三条边无法构成三角形。
50	1	50	这是一个等腰三角形。	这是一个等腰三角形。
50	2	50	这是一个等腰三角形。	这是一个等腰三角形。
50	99	50	这是一个等腰三角形。	这是一个等腰三角形。
50	100	50	这三条边无法构成三角形。	这三条边无法构成三角形。
50	101	50	这三条边无法构成三角形。	这三条边无法构成三角形。
50	50	0	这三条边无法构成三角形。	这三条边无法构成三角形。
50	50	1	这是一个等腰三角形。	这是一个等腰三角形。
50	50	2	这是一个等腰三角形。	这是一个等腰三角形。
50	50	99	这是一个等腰三角形。	这是一个等腰三角形。
50	50	100	这三条边无法构成三角形。	这三条边无法构成三角形。
50	50	101	这三条边无法构成三角形。	这三条边无法构成三角形。
50	50	50	这是一个等边三角形。	这是一个等边三角形。
0	0	0	这三条边无法构成三角形。	这三条边无法构成三角形。
1	1	1	这是一个等边三角形。	这是一个等边三角形。
2	2	2	这是一个等边三角形。	这是一个等边三角形。
0	2	0	这三条边无法构成三角形。	这三条边无法构成三角形。
99	99	99	这是一个等边三角形。	这是一个等边三角形。
100	100	100	这是一个等边三角形。	这是一个等边三角形。
101	101	101	这是一个等边三角形。	这是一个等边三角形。
50	50	50	这是一个等边三角形。	这是一个等边三角形。
2	0	2	这三条边无法构成三角形。	这三条边无法构成三角形。
0	2	2	这三条边无法构成三角形。	这三条边无法构成三角形。

NextDay 测试数据				
y	m	d	测试输出	预期输出
1912	6	16	Next date: 1912-6-17	Next date: 1912-6-17
1913	6	16	Next date: 1913-6-17	Next date: 1913-6-17
2049	6	16	Next date: 2049-6-17	Next date: 2049-6-17
2050	6	16	Next date: 2050-6-17	Next date: 2050-6-17
1981	1	16	Next date: 1981-1-17	Next date: 1981-1-17
1981	2	16	Next date: 1981-2-17	Next date: 1981-2-17
1981	11	16	Next date: 1981-11-17	Next date: 1981-11-17
1981	12	16	Next date: 1981-12-17	Next date: 1981-12-17
1981	6	1	Next date: 1981-6-2	Next date: 1981-6-2
1981	6	2	Next date: 1981-6-3	Next date: 1981-6-3
1981	6	30	Next date: 1981-7-1	Next date: 1981-7-1
1981	6	31	Next date: 1981-7-1	Next date: 1981-7-1
1981	6	16	Next date: 1981-6-17	Next date: 1981-6-17
1911	6	16	Invalid input date	Invalid input date
1912	6	16	Next date: 1912-6-17	Next date: 1912-6-17
1913	6	16	Next date: 1913-6-17	Next date: 1913-6-17
2049	6	16	Next date: 2049-6-17	Next date: 2049-6-17
2050	6	16	Next date: 2050-6-17	Next date: 2050-6-17
2051	6	16	Invalid input date	Invalid input date
1981	0	16	Invalid input date	Invalid input date
1981	1	16	Next date: 1981-1-17	Next date: 1981-1-17
1981	2	16	Next date: 1981-2-17	Next date: 1981-2-17
1981	11	16	Next date: 1981-11-17	Next date: 1981-11-17
1981	12	16	Next date: 1981-12-17	Next date: 1981-12-17
1981	13	16	Invalid input date	Invalid input date
1981	6	0	Invalid input date	Invalid input date
1981	6	1	Next date: 1981-6-2	Next date: 1981-6-2
1981	6	2	Next date: 1981-6-3	Next date: 1981-6-3
1981	6	30	Next date: 1981-7-1	Next date: 1981-7-1
1981	6	31	Next date: 1981-7-1	Next date: 1981-7-1
1981	6	32	Invalid input date	Invalid input date
1981	6	16	Next date: 1981-6-17	Next date: 1981-6-17
1911	0	0	Invalid input date	Invalid input date
1912	1	1	Next date: 1912-1-2	Next date: 1912-1-2
1913	2	2	Next date: 1913-2-3	Next date: 1913-2-3
1911	2	0	Invalid input date	Invalid input date
2049	11	30	Next date: 2049-12-1	Next date: 2049-12-1
2050	12	31	Next date: 2051-1-1	Next date: 2051-1-1
2051	13	32	Invalid input date	Invalid input date
1981	6	16	Next date: 1981-6-17	Next date: 1981-6-17
1913	0	2	Invalid input date	Invalid input date
1911	2	2	Invalid input date	Invalid input date

8 [测试结果分析]

测试结果符合预期, 基本可以认定程序正确

9 [实验总结]

在进行黑盒测试的边界值测试法实验后，我得出了一些结论。边界值测试法是一种测试方法，着重于测试输入值的边界情况，因为这些情况通常更容易导致程序错误。在实验中，我使用了这种方法来测试程序的各种输入情况，并且得出了以下几点总结。

首先，边界值测试法能够有效地发现程序中的潜在错误。通过测试输入值的边界情况，我发现了一些在正常情况下可能被忽视的问题。例如，当输入值处于边界上时，程序可能会产生意料之外的行为，比如溢出或者未处理的异常情况。因此，边界值测试法可以帮助我们发现这些潜在的问题，从而提高程序的质量和稳定性。

其次，边界值测试法有助于提高测试效率。相比于随机选择输入值进行测试，使用边界值测试法可以更有针对性地选择测试用例，从而更有效地覆盖程序的各种可能情况。这样一来，我们可以在更短的时间内发现更多的问题，提高测试的效率和准确性。

另外，边界值测试法也有一些局限性。虽然这种方法可以有效地发现一些特定类型的错误，但并不能覆盖所有可能的情况。有些程序错误可能不仅仅与输入值的边界有关，还与输入值的组合、程序的逻辑等因素有关。因此，在进行测试时，我们仍需要结合其他测试方法，如等价类划分、状态转换测试等，来全面地覆盖程序的各种情况。

综上所述，边界值测试法是一种有效的黑盒测试方法，可以帮助我们发现程序中的潜在错误，并提高测试效率。然而，在使用这种方法时，我们也需要注意其局限性，结合其他测试方法来进行综合测试，以确保程序的质量和稳定性。

10 [附: 程式源码]

rust code

```rust
use std::env;

fn main() {
    let args: Vec<String> = env::args().collect();

    let (a, b, c) = match parse_arguments(&args) {
        Ok((a, b, c)) => (a, b, c),
        Err(err) => {
            println!("{}", err);
            return;
        }
    };

    if !is_triangle(a, b, c) {
        println!("这三条边无法构成三角形。");
        return;
    }

    if a == b && b == c {
        println!("这是一个等边三角形。");
    } else if a == b || b == c || a == c {
        println!("这是一个等腰三角形。");
    } else {
        println!("这是一个不等边三角形。");
    }
}

fn parse_arguments(args: &[String]) -> Result<(u32, u32, u32), String> {
```

```
    if args.len() != 7 {
        return Err(String::from("使用方法:␣./main␣-a␣<边长a>␣-b␣<边长b>␣-c␣<边长c>"));
    }

    let mut a = None;
    let mut b = None;
    let mut c = None;

    for i in 1..args.len() {
        match args[i].as_str() {
            "-a" => {
                a = Some(args[i + 1].parse().map_err(|_| "边长a必须是一个有效的整数")?);
            }
            "-b" => {
                b = Some(args[i + 1].parse().map_err(|_| "边长b必须是一个有效的整数")?);
            }
            "-c" => {
                c = Some(args[i + 1].parse().map_err(|_| "边长c必须是一个有效的整数")?);
            }
            _ => {}
        }
    }

    match (a, b, c) {
        (Some(a), Some(b), Some(c)) => Ok((a, b, c)),
        _ => Err(String::from("缺少边长参数")),
    }
}

fn is_triangle(a: u32, b: u32, c: u32) -> bool {
    a + b > c && b + c > a && a + c > b
}
```

bash code

```
perform_test() {
    local a="$1"
    local b="$2"
    local c="$3"
    local log_file="test.log"


    # 运行小程序并记录输出
    output=$(./main -a "$a" -b "$b" -c "$c")
    echo "%␣_____" >> "$log_file"
    # 输出测试输入
    echo "$a␣&␣$b␣&␣$c␣&␣$output␣&␣$output␣\\\\" >> "$log_file"

}

rm -rf "test.log"

ymin=1
ymax=100
```

```
y=$(( ($ymin + $ymax) / 2 ))

mmin=1
mmax=100
m=$(( ($mmin + $mmax) / 2 ))

dmin=1
dmax=100
d=$(( ($dmin + $dmax) / 2 ))


perform_test $ymin $m $d
perform_test $(( $ymin + 1 )) $m $d
perform_test $(( $ymax - 1 )) $m $d
perform_test $ymax $m $d

perform_test $y $mmin $d
perform_test $y $(( $mmin + 1 )) $d
perform_test $y $(( $mmax - 1 )) $d
perform_test $y $mmax $d

perform_test $y $m $dmin
perform_test $y $m $(( $dmin + 1 ))
perform_test $y $m $(( $dmax - 1 ))
perform_test $y $m $dmax

perform_test $y $m $d


echo "%正常值测试完成！" >> "test.log"

perform_test $(( $ymin - 1 )) $m $d
perform_test $ymin $m $d
perform_test $(( $ymin + 1 )) $m $d
perform_test $(( $ymax - 1 )) $m $d
perform_test $ymax $m $d
perform_test $(( $ymax + 1 )) $m $d

perform_test $y $(( $mmin - 1 )) $d
perform_test $y $mmin $d
perform_test $y $(( $mmin + 1 )) $d
perform_test $y $(( $mmax - 1 )) $d
perform_test $y $mmax $d
perform_test $y $(( $mmax + 1 )) $d

perform_test $y $m $(( $dmin - 1 ))
perform_test $y $m $dmin
perform_test $y $m $(( $dmin + 1 ))
perform_test $y $m $(( $dmax - 1 ))
perform_test $y $m $dmax
perform_test $y $m $(( $dmax + 1 ))

perform_test $y $m $d

perform_test $(( $ymin - 1 )) $(( $mmin - 1 )) $(( $dmin - 1 ))
```

```
perform_test $ymin $mmin $dmin
perform_test $(( $ymin + 1 )) $(( $mmin + 1 )) $(( $dmin + 1 ))
perform_test $(( $ymin - 1 )) $(( $mmin + 1 )) $(( $dmin - 1 ))
perform_test $(( $ymax - 1 )) $(( $mmax - 1 )) $(( $dmax - 1 ))
perform_test $ymax $mmax $dmax
perform_test $(( $ymax + 1 )) $(( $mmax + 1 )) $(( $dmax + 1 ))
perform_test $y $m $d
perform_test $(( $ymin + 1 )) $(( $mmin - 1 )) $(( $dmin + 1 ))
perform_test $(( $ymin - 1 )) $(( $mmin + 1 )) $(( $dmin + 1 ))
```

rust code

```
use std::env;

fn is_leap_year(year: i32) -> bool {
    (year % 4 == 0 && year % 100 != 0) || year % 400 == 0
}

fn next_date(month: i32, day: i32, mut year: i32) -> (i32, i32, i32) {
    let days_in_month = match month {
        1 | 3 | 5 | 7 | 8 | 10 | 12 => 31,
        4 | 6 | 9 | 11 => 30,
        2 => {
            if is_leap_year(year) {
                29
            } else {
                28
            }
        }
        _ =>  {
            println!("Invalid␣month");
            return (0, 0, 0); // 返回一個無效的日期，表示錯誤
            }
   };

    if day < days_in_month {
        (month, day + 1, year)
    } else if month < 12 {
        (month + 1, 1, year)
    } else {
        year += 1;
        (1, 1, year)
    }
}

fn main() {
    let args: Vec<String> = env::args().collect();

    if args.len() != 7 {
        println!("Usage:␣./main␣-y␣<year>␣-m␣<month>␣-d␣<day>");
        return;
    }

    let mut year = 0;
```

```
    let mut month = 0;
    let mut day = 0;

    for i in 1..7 {
        match args[i].as_str() {
            "-y" => {
                year = args[i + 1].parse().expect("Invalid year");
            }
            "-m" => {
                month = args[i + 1].parse().expect("Invalid month");
            }
            "-d" => {
                day = args[i + 1].parse().expect("Invalid day");
            }
            _ => {}
        }
    }

    if !(1..=12).contains(&month) || !(1..=31).contains(&day) || !(1912..=2050).contains(&year) {
        println!("Invalid input date");
        return;
    }

    let (next_month, next_day, next_year) = next_date(month, day, year);
    if next_month== 0 && next_day == 0 && next_year== 0 {
            return;
    }else {
            println!("Next date: {}-{}-{}", next_year, next_month, next_day);
            return;
        }
    }
```

bash code

```
# Function to perform robustness testing
perform_test() {
    local year=$1
    local month=$2
    local day=$3

    echo "测试 Year=$year, Month=$month, Day=$day"


    # Run the program with the given input and log the output
    output=$(./main -y "$year" -m "$month" -d "$day")
    # Log input parameters
    echo "$year & $month & $day & $output & $output \\\\" >> test.log
}

# Clean existing log file
rm -f test.log

ymin=1912
```

```
ymax=2050
y=$(( ($ymin + $ymax) / 2 ))

mmin=1
mmax=12
m=$(( ($mmin + $mmax) / 2 ))

dmin=1
dmax=31
d=$(( ($dmin + $dmax) / 2 ))

perform_test $ymin $m $d
perform_test $(( $ymin + 1 )) $m $d
perform_test $(( $ymax - 1 )) $m $d
perform_test $ymax $m $d

perform_test $y $mmin $d
perform_test $y $(( $mmin + 1 )) $d
perform_test $y $(( $mmax - 1 )) $d
perform_test $y $mmax $d

perform_test $y $m $dmin
perform_test $y $m $(( $dmin + 1 ))
perform_test $y $m $(( $dmax - 1 ))
perform_test $y $m $dmax

perform_test $y $m $d

echo "所有正常测试完成！"

perform_test $(( $ymin - 1 )) $m $d
perform_test $ymin $m $d
perform_test $(( $ymin + 1 )) $m $d
perform_test $(( $ymax - 1 )) $m $d
perform_test $ymax $m $d
perform_test $(( $ymax + 1 )) $m $d

perform_test $y $(( $mmin - 1 )) $d
perform_test $y $mmin $d
perform_test $y $(( $mmin + 1 )) $d
perform_test $y $(( $mmax - 1 )) $d
perform_test $y $mmax $d
perform_test $y $(( $mmax + 1 )) $d

perform_test $y $m $(( $dmin - 1 ))
perform_test $y $m $dmin
perform_test $y $m $(( $dmin + 1 ))
perform_test $y $m $(( $dmax - 1 ))
perform_test $y $m $dmax
perform_test $y $m $(( $dmax + 1 ))

perform_test $y $m $d

perform_test $(( $ymin - 1 )) $(( $mmin - 1 )) $(( $dmin - 1 ))
perform_test $ymin $mmin $dmin
```

```
perform_test $(( $ymin + 1 )) $(( $mmin + 1 )) $(( $dmin + 1 ))
perform_test $(( $ymin - 1 )) $(( $mmin + 1 )) $(( $dmin - 1 ))
perform_test $(( $ymax - 1 )) $(( $mmax - 1 )) $(( $dmax - 1 ))
perform_test $ymax $mmax $dmax
perform_test $(( $ymax + 1 )) $(( $mmax + 1 )) $(( $dmax + 1 ))
perform_test $y $m $d
perform_test $(( $ymin + 1 )) $(( $mmin - 1 )) $(( $dmin + 1 ))
perform_test $(( $ymin - 1 )) $(( $mmin + 1 )) $(( $dmin + 1 ))


echo "所有健壮性测试完成！"
```