**Министерство образования Республики Беларусь**

Учреждение образования

БелорусскиЙ государственный университет

информатики и радиоэлектроники

Факультет компьютерных систем и сетей

Кафедра программного обеспечения информационных технологий

Дисциплина: Метрология стандартизация и сертификация в информационных технологиях (МСиСвИнфТ)

|  |
| --- |
|  |
|  |

# оТЧЕТ

по лабораторной работе

на тему:

Тема работы: Расчет метрик Холстеда

|  |  |  |
| --- | --- | --- |
| Выполнил  Студенты гр. 051005 |  | К.Н. Волков  И.Д. Сидоренко |
| Проверил |  | Асс. C.В. Болтак |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

Минск, 2021

1. **Фрагмент анализируемого кода**

func BinHexToDec(num int) {

len := 7

base := 8

res := 0

temp := 0

num1 := 0

if base == 2 {

for a := 0; a < 3; a += 1 {

for i := 0; i <= len-1; i += 1 {

if num == '1' {

temp = 1

for j := 0; j <= len-1-i; j += 1 {

temp = temp \* base

}

res = res + temp

}

}

}

} else if base == 8 {

for i := 0; i <= len-1; i += 1 {

num1 = 0

switch num {

case 000:

{num1 = 0}

case 001:

{num1 = 1}

case 010:

{num1 = 2}

case 011:

{num1 = 3}

case 100:

{num1 = 4}

case 101:

{num1 = 5}

case 110:

{num1 = 6}

case 111:

{num1 = 7}

}

temp = 1

for j := 0; j <= len-i; j += 1 {

temp = temp \* base

}

res = res + temp\*num1

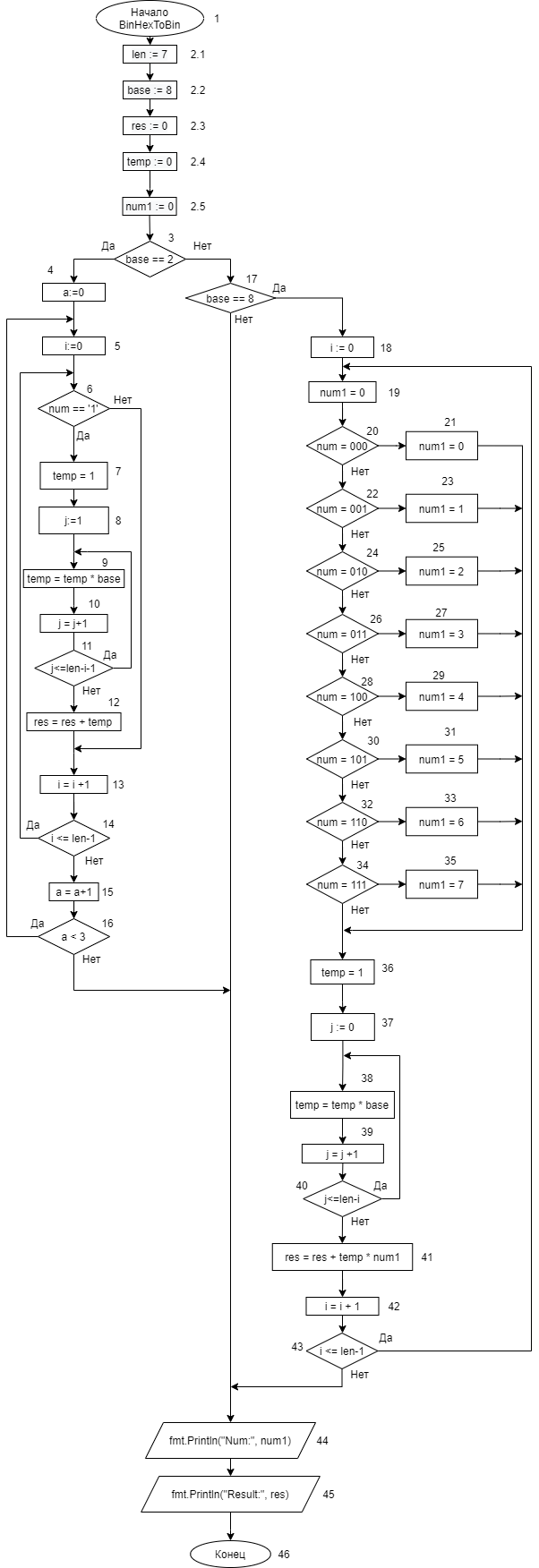
}

}

fmt.Println("Num:", num1)

fmt.Println("Result:", res)

}



*Рис 1 — Графическая схема алгоритма работы программы*

**2. Метрика граничных значений фрагмента кода**

**Метрика граничных значений** базируется на определении скорректированной сложности вершин графа программы.

*Абсолютная граничная сложность* программы Sa определяется как сумма скорректированных сложностей всех вершин графа G.

*Относительная граничная* сложность программы So определяется по формуле:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Номер вершины выбора | Номер вершины перехода | Скорректированная сложность вершины выбора | Номера вершин подграфа | Номер нижней границы подграфа |
| 3 | 4, 17 | 41 | 4-43 | 44 |
| 6 | 7, 13 | 7 | 7-12 | 13 |
| 11 | 9, 12 | 4 | 9-11 | 12 |
| 14 | 6, 15 | 10 | 6-14 | 15 |
| 16 | 5, 44 | 11 | 5-16 | 44 |
| 17 | 18, 44 | 27 | 18-43 | 44 |
| 20 | 21, 22 | 16 | 21-35 | 36 |
| 22 | 23, 24 | 14 | 23-35 | 36 |
| 24 | 25, 26 | 12 | 25-35 | 36 |
| 26 | 27, 28 | 10 | 27-35 | 36 |
| 28 | 29, 30 | 8 | 29-35 | 36 |
| 30 | 31, 32 | 6 | 31-35 | 36 |
| 32 | 33, 34 | 4 | 33-35 | 36 |
| 34 | 35, 36 | 2 | 35 | 36 |
| 40 | 38, 41 | 4 | 38-40 | 41 |
| 43 | 19, 44 | 26 | 19-43 | 44 |

*Таблица 2.1 — Свойства подграфов фрагмента программы*

|  |  |
| --- | --- |
| Вершина | ССВ |
| 1 | 1 |
| 2.1 | 1 |
| 2.2 | 1 |
| 2.3 | 1 |
| 2.4 | 1 |
| 2.5 | 1 |
| 3 | 41 |
| 4 | 1 |
| 5 | 1 |
| 6 | 7 |
| 7 | 1 |
| 8 | 1 |
| 9 | 1 |
| 10 | 1 |
| 11 | 4 |
| 12 | 1 |
| 13 | 1 |
| 14 | 10 |
| 15 | 1 |
| 16 | 11 |
| 17 | 27 |
| 18 | 1 |
| 19 | 1 |
| 20 | 16 |
| 21 | 1 |
| 22 | 14 |
| 23 | 1 |
| 24 | 12 |
| 25 | 1 |
| 26 | 10 |
| 27 | 1 |
| 28 | 8 |
| 29 | 1 |
| 30 | 6 |
| 31 | 1 |
| 32 | 4 |
| 33 | 1 |
| 34 | 2 |
| 35 | 1 |
| 36 | 1 |
| 37 | 1 |
| 38 | 1 |
| 39 | 1 |
| 40 | 4 |
| 41 | 1 |
| 42 | 1 |
| 43 | 26 |
| 44 | 1 |
| 45 | 1 |
| 46 | 0 |

*Таблица 2.2 — Скорректированные сложности вершин графа фрагмента программы*

**Метрики Маккейба для фрагмента программы:**

Абсолютная граничная сложность Sa = 236

Относительная граничная сложность So = 1 - (51-1)/236 =0,788

Цикломатическое число Маккейба Z(G) = 66-51+2\*1=17

**Метрики Джилба для фрагмента программы:**

Абсолютная сложность программы CL = 16

Относительная сложность программы cl = 0,14

Максимальный уровень вложенности условного оператора CLI = 10

**Метрики Джилба для полной программы:**

Абсолютная сложность программы CL = 63

Относительная сложность программы cl = 0,16

Максимальный уровень вложенности условного оператора CLI = 11

**3. Код анализируемой программы**

package main

import (

"fmt"

"time"

)

func PosOrNeg(n int) {

if n > 0 {

fmt.Println("+")

} else if n == 0 {

fmt.Println("NULL")

} else if n < 0 {

fmt.Println("-")

}

}

func LeapYear(year int) string {

leapFlag := 0

if year%4 == 0 {

if year%100 == 0 {

if year%400 == 0 {

leapFlag = 1

} else {

leapFlag = 0

}

} else {

leapFlag = 1

}

} else {

leapFlag = 0

}

if leapFlag == 1 {

return "Yes"

} else {

return "No"

}

}

func WriteNumAsFrom0To4(num int) (str string) {

switch num {

case 0:

str = "Zero"

case 1:

str = "One"

case 2:

str = "Two"

case 3:

str = "Three"

case 4:

str = "Four"

default:

str = "It's more than Four"

}

return

}

func BinHexToDec(num int) {

len := 7

base := 8

res := 0

temp := 0

num1 := 0

if base == 2 {

for a := 0; a < 3; a += 1 {

for i := 0; i <= len-1; i += 1 {

if num == '1' {

temp = 1

for j := 0; j <= len-1-i; j += 1 {

temp = temp \* base

}

res = res + temp

}

}

}

} else if base == 8 {

for i := 0; i <= len-1; i += 1 {

num1 = 0

switch num {

case 000:

{

num1 = 0

}

case 001:

{

num1 = 1

}

case 010:

{

num1 = 2

}

case 011:

{

num1 = 3

}

case 100:

{

num1 = 4

}

case 101:

{

num1 = 5

}

case 110:

{

num1 = 6

}

case 111:

{

num1 = 7

}

}

temp = 1

for j := 0; j <= len-i; j += 1 {

temp = temp \* base

}

res = res + temp\*num1

}

}

fmt.Println("Num:", num1)

fmt.Println("Result:", res)

}

func NoonTime() (str string) {

t := time.Now()

switch {

case t.Hour() < 12:

str = "It's before noon"

case t.Hour() == 12:

str = "It's noon"

default:

str = "It's after noon"

}

return

}

func Useless(n int) (sum int) {

sum = 0

switch n {

case 1:

{

for i := 0; i <= n; i++ {

sum = sum + n\*50\*i

}

}

case 2:

{

for i := 0; i <= n; i++ {

sum = sum + n\*40\*i

}

}

case 3:

{

for i := 0; i <= n; i++ {

sum = sum + n\*30\*i

}

}

case 4:

{

for i := 0; i <= n; i++ {

sum = sum + n\*20\*i

}

}

}

return

}

func Last(num int) (str string) {

if num < 0 {

str = "is negative"

} else if num < 10 {

str = "has 1 digit"

} else {

str = "has multiple digits"

}

return

}

func calcn(n int) (x int) {

if n < 0 {

x = n

} else if n == 0 {

x = 1

} else if n < 10 {

x = n \* 2

} else if n < 20 {

x = n \* 4

}

return

}

func itisnumber(n int) (str string) {

var i int

str = "Error"

for i = 0; i < 10; i++ {

if i == n {

switch n {

case 0:

{

str = "Zero"

break

}

case 1:

{

str = "One"

break

}

case 2:

{

str = "Two"

break

}

case 3:

{

str = "Three"

break

}

case 4:

{

str = "Four"

break

}

case 5:

{

str = "Five"

break

}

case 6:

{

str = "Six"

break

}

case 7:

{

str = "Seven"

break

}

case 8:

{

str = "Eight"

break

}

case 9:

{

str = "Nine"

break

}

}

}

}

return

}

func gg(str string) {

sum := 0

for i := 0; i < len(str); i++ {

if str[i] == 'a' {

fmt.Printf("%c", str[i])

sum = sum + 1

}

}

fmt.Printf("\n%d\n", sum)

}

func ff(n int) int {

if n < 0 {

fmt.Println("Error")

return n

}

var digit int

sum := 0

for n > 0 {

digit = n % 10

sum += digit

n = n / 10

}

return sum

}

func main() {

PosOrNeg(0)

fmt.Println("1912 Leap? ", LeapYear(1912))

var num int = 4

fmt.Printf("Write %d as ", num)

fmt.Println(WriteNumAsFrom0To4(num))

BinHexToDec(010)

fmt.Println(NoonTime())

fmt.Println(Useless(3))

fmt.Println("120", Last(120))

fmt.Println(calcn(15))

fmt.Println(itisnumber(1))

gg("aagfgfannaaansdffsnaatta")

sum := ff(123)

if sum > 0 {

fmt.Println(sum)

}

}