软件测试基础与实践

实验报告

实验名称：白盒测试实验三

实验地点：软件学院机房

实验日期：2020 年 11 月 27 日

学生姓名：叶宏庭

学生学号：71118415

东南大学 软件学院 制

一、实验目的

（1）巩固白盒测试知识，能熟练应用数据流覆盖方法设计测试用例；
（2）学习测试用例的书写。

二、实验内容

（一）题目 1：数据量测试技术实验

1. 运用数据流测试方法，对用 C/C++语言实现的 CgiDecode 程序中的 decode()方法进行测试。

要求：

（1） 测试要考虑 decode()中 encoded, decoded, *eptr, eptr, *dptr, dptr, ok, c, digit_high, digit_low 变量；

（2） 给出每个变量对应的 du-path 和 dc-path;

（3） 根据变量的 dc-path 设计测试用例，完成对 decode()的测试；

实验过程注意要点：

（1） 变量*eptr 和变量*dptr：
由于这种变量涉及到对指针进行*操作，因此非声明位置出现*eptr 和*dptr 的时候都视为是相应指针 eptr 和 dptr 的使用节点。

（2） 难点 1：正确分析变量的定义节点和使用节点；

（3） 难点 2：变量的定义节点不要求变量一定出现。
在指针发生变化的时候，会影响到相应的指针变量的值。因此，语句 22 虽然没有出现*eptr，但却是*eptr 的一个定义节点。

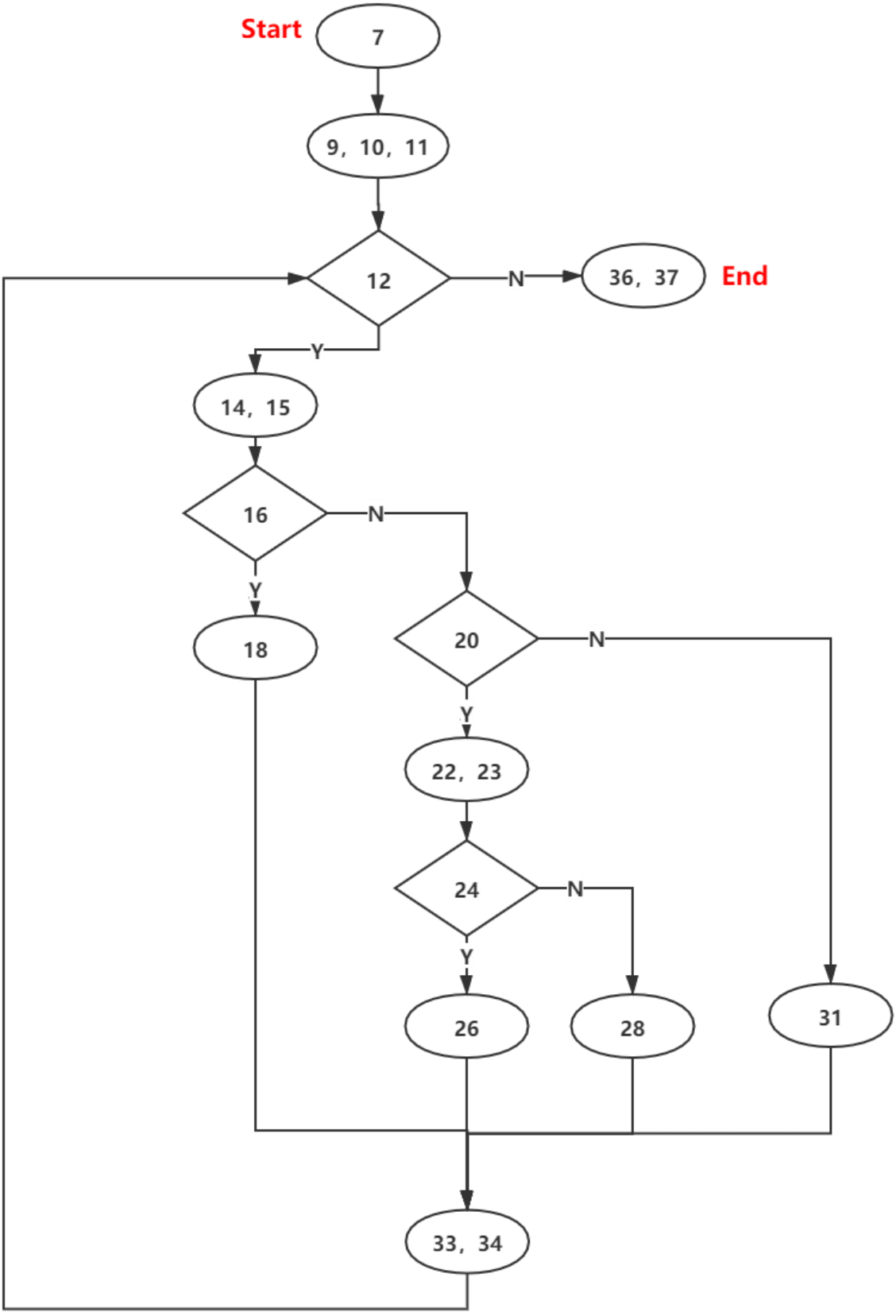
（4） 提供一个 CGI 解码的程序供理解和测试过程中参考，其中 getHexValue()的作用是取对应字符串的十六进制值。

（5） DU-PATH/DC-PATH 的数量及其确定：定义节点 A 到使用节点 B 之间可能有多条 DU-PATH/DC-PATH，理论上这些 DU-PATH/DC-PATH 都需要进行测试，如果这些路径可能有无穷多条，请指出来。

Decode()函数的语句及其编号如下：

1	/** Translate a string from the CGI encoding to plain ascii text.
2	* '+' becomes space, %xx becomes byte with hex value xx,
3	* other alphanumeric characters map to themselves.
4	* Returns 0 for success, positive for erroneous input
5	* 1 = bad hexadecimal digit
6	*/
7	int decode(char *encoded, char *decoded)
8	{
9	char *eptr = encoded;
10	char *dptr = decoded;
11	int ok=0;
12	while (*eptr)
13	{
14	char c;
15	c = *eptr;
16	if (c == '+')
17	{ /* Case 1: '+' maps to blank */
18	*dptr = ' ';
19	}
20	else if (c == '%')
21	{ /* Case 2: '%xx' is hex for character xx */
22	int digit_high = getHexValue(*(++eptr));
23	int digit_low = getHexValue(*(++eptr));
24	if (digit_high == -1 || digit_low==-1) {
25	/* *dptr='?'; */
26	ok=1; /* Bad return code */
27	} else {
28	*dptr = 16* digit_high + digit_low;
29	}
30	} else {/* Case 3: All other characters map to themselves */
31	*dptr = *eptr;
32	}
33	++dptr;
34	++eptr;
35	}
36	*dptr = '\0'; /* Null terminator for string */
37	return ok;
38	}

Decode()函数流程图如下：



2. 解答第（1）问

下面给出各个变量的 DEF、USE 节点：

变量：encoded		
Node	Type	Code
7	DEF	int decode(char *encoded, char *decoded)
9	USE	char *eptr = encoded;

变量：decoded		
Node	Type	Code
7	DEF	int decode(char *encoded, char *decoded)
10	USE	char *dptr = decoded;

变量：*eptr		
Node	Type	Code
9	DEF	char *eptr = encoded;
22	DEF	int digit_high = getHexValue(*(++eptr));
23	DEF	int digit_low = getHexValue(*(++eptr));
34	DEF	++eptr;
12	USE	while (*eptr)
15	USE	c = *eptr;
22	USE	int digit_high = getHexValue(*(++eptr));
23	USE	int digit_low = getHexValue(*(++eptr));
31	USE	*dptr = *eptr;

变量：eptr		
Node	Type	Code
9	DEF	char *eptr = encoded;
22	DEF	int digit_high = getHexValue(*(++eptr));
23	DEF	int digit_low = getHexValue(*(++eptr));
34	DEF	++eptr;
12	USE	while (*eptr)
15	USE	c = *eptr;
22	USE	int digit_high = getHexValue(*(++eptr));
23	USE	int digit_low = getHexValue(*(++eptr));
31	USE	*dptr = *eptr;
34	USE	++eptr;

变量：*dptr		
Node	Type	Code
10	DEF	char *dptr = decoded;
18	DEF	*dptr = ' ';

28	DEF	*dptr = 16* digit_high + digit_low;
31	DEF	*dptr = *eptr;
33	DEF	++dptr;
36	DEF	*dptr = '\0';

变量：dptr		
Node	Type	Code
10	DEF	char *dptr = decoded;
33	DEF	++dptr;
18	USE	*dptr = ' ';
28	USE	*dptr = 16* digit_high + digit_low;
31	USE	*dptr = *eptr;
33	USE	++dptr;
36	USE	*dptr = '\0';

变量：ok		
Node	Type	Code
11	DEF	int ok=0;
26	DEF	ok=1; /* Bad return code */
37	USE	return ok;

变量：c		
Node	Type	Code
14	DEF	char c;
15	DEF	c = *eptr;
16	USE	if (c == '+')
20	USE	else if (c == '%')

变量：digit_high		
Node	Type	Code
22	DEF	int digit_high = getHexValue(*(++eptr));
24	USE	if (digit_high == -1 || digit_low==-1) {
28	USE	*dptr = 16* digit_high + digit_low;

变量：digit_low		
Node	Type	Code
23	DEF	int digit_low = getHexValue(*(++eptr));
24	USE	if (digit_high == -1 || digit_low==-1) {
28	USE	*dptr = 16* digit_high + digit_low;

3. 解答第 2 问

变量：encoded		
Du-path		
编号	Type	Path
P1	Du-path	7-8-9
Dc-path		
编号	Type	Path
P1	Dc-path	7-8-9

变量：decoded		
Du-path		
编号	Type	Path
P1	Du-path	7-8-9-10
Dc-path		
编号	Type	Path
P1	Dc-path	7-8-9-10

变量：*eptr		
Du-path		
编号	Type	Path
P1	Du-path	9-10-11-12
P2	Du-path	9-10-11-12-13-14-15
P3	Du-path	9-10-11-12-13-14-15-16-20-21-22
P4	Du-path	9-10-11-12-13-14-15-16-20-21-22-23
P5	Du-path	9-10-11-12-13-14-15-16-20-30-31
P6	Du-path	22
P7	Du-path	22-23
P8	Du-path	23
Dc-path		
编号	Type	Path
P1	Dc-path	9-10-11-12
P2	Dc-path	9-10-11-12-13-14-15
P3	Dc-path	9-10-11-12-13-14-15-16-20-30-31
P4	Dc-path	22
P5	Dc-path	23

变量：eptr		
Du-path		
编号	Type	Path
P1	Du-path	9-10-11-12
P2	Du-path	9-10-11-12-13-14-15

P3	Du-path	9-10-11-12-13-14-15-16-20-21-22
P4	Du-path	9-10-11-12-13-14-15-16-20-21-22-23
P5	Du-path	9-10-11-12-13-14-15-16-20-30-31
P6	Du-path	9-10-11-12-13-14-15-16-17-18-33-34 或 9-10-11-12-13-14-15-16-20-21-22-23-24-25-26-33-34 或 9-10-11-12-13-14-15-16-20-21-22-23-24-27-28-33-34 或 9-10-11-12-13-14-15-16-20-30-31-33-34
Dc-path		
编号	Type	Path
P1	Dc-path	9-10-11-12
P2	Dc-path	9-10-11-12-13-14-15
P3	Dc-path	9-10-11-12-13-14-15-16-20-30-31

变量：*dptr		
Du-path		
编号	Type	Path
无		
Dc-path		
编号	Type	Path
无		

变量：dptr		
Du-path		
编号	Type	Path
P1	Du-path	10-11-12-13-14-15-16-17-18
P2	Du-path	10-11-12-13-14-15-16-20-21-22-23-24-28
P3	Du-path	10-11-12-13-14-15-16-20-30-31
P4	Du-path	10-11-12-13-14-15-16-17-18-33 或 10-11-12-13-14-15-16-20-21-22-23-24-25-26-33 或 10-11-12-13-14-15-16-20-21-22-23-24-28-33 或 10-11-12-13-14-15-16-20-30-31-33
P5	Du-path	10-11-12-36 或 10-11-12-13-14-15-16-17-18-33-34-12-36 或 10-11-12-13-14-15-16-20-21-22-23-24-25-26-33-34-12-36 或 10-11-12-13-14-15-16-20-21-22-23-24-28-33-34-12-36 或 10-11-12-13-14-15-16-20-30-31-33-34-12-36
Dc-path		
编号	Type	Path
P1	Dc-path	10-11-12-13-14-15-16-17-18
P2	Dc-path	10-11-12-13-14-15-16-20-21-22-23-24-28
P3	Dc-path	10-11-12-13-14-15-16-20-30-31
P4	Dc-path	10-11-12-36

变量：ok		
Du-path		
编号	Type	Path
P1	Du-path	11-12-36-37
P2	Du-path	26-33-34-12-36-37
Dc-path		
编号	Type	Path
P1	Dc-path	11-12-36-37
P2	Dc-path	26-33-34-12-36-37

变量：c		
Du-path		
编号	Type	Path
P1	Du-path	14-15-16
P2	Du-path	14-15-16-20
P3	Du-path	15-16
P4	Du-path	15-16-20
Dc-path		
编号	Type	Path
P1	Dc-path	15-16
P2	Dc-path	15-16-20

变量：digit_high		
Du-path		
编号	Type	Path
P1	Du-path	22-23-24
P2	Du-path	22-23-24-28
Dc-path		
编号	Type	Path
P1	Dc-path	22-23-24
P2	Dc-path	22-23-24-28

变量：digit_low		
Du-path		
编号	Type	Path
P1	Du-path	23-24
P2	Du-path	23-24-28
Dc-path		
编号	Type	Path
P1	Dc-path	23-24

P2	Dc-path	23-24-28

4. 解答第 3 问

针对各个变量的测试用例如下：

变量：encoded					
Dc-path					
编号	Type	Path			
P1	Dc-path	7-8-9			
测试用例					
编号	输入 encode	输入 decode	预期输出 ok	实际输出 ok	覆盖路径
1	SoftwareTest	NULL	0	0	P1

变量：decoded					
Dc-path					
编号	Type	Path			
P1	Du-path	7-8-9-10			
测试用例					
编号	输入 encode	输入 decode	预期输出 ok	实际输出 ok	覆盖路径
1	SoftwareTest	NULL	0	0	P1

变量：*eptr					
Dc-path					
编号	Type	Path			
P1	Dc-path	9-10-11-12			
P2	Dc-path	9-10-11-12-13-14-15			
P3	Dc-path	9-10-11-12-13-14-15-16-20-30-31			
P4	Dc-path	22			
P5	Dc-path	23			
测试用例					
编号	输入 encode	输入 decode	预期输出 ok	实际输出 ok	覆盖路径
1	SoftwareTest	NULL	0	0	P1,P2,P3
2	Software%Test	NULL	0	0	P1,P2,P3,P4,P5

变量：eptr		
Dc-path		
编号	Type	Path
P1	Dc-path	9-10-11-12
P2	Dc-path	9-10-11-12-13-14-15

P3	Dc-path	9-10-11-12-13-14-15-16-20-30-31			
测试用例					
编号	输入 encode	输入 decode	预期输出 ok	实际输出 ok	覆盖路径
1	SoftwareTest	NULL	0	0	P1,P2,P3

变量：*dptr					
Dc-path					
编号	Type	Path			
无					
测试用例					
编号	输入 encode	输入 decode	预期输出 ok	实际输出 ok	覆盖路径
无					

变量：dptr					
Dc-path					
编号	Type	Path			
P1	Dc-path	10-11-12-13-14-15-16-17-18			
P2	Dc-path	10-11-12-13-14-15-16-20-21-22-23-24-28			
P3	Dc-path	10-11-12-13-14-15-16-20-30-31			
P4	Dc-path	10-11-12-36			
测试用例					
编号	输入 encode	输入 decode	预期输出 ok	实际输出 ok	覆盖路径
1	Software+Test%Y	NULL	0	0	P1,P2,P3
2	NULL	NULL	0	0	P4

变量：ok					
Dc-path					
编号	Type	Path			
P1	Dc-path	11-12-36-37			
P2	Dc-path	26-33-34-12-36-37			
测试用例					
编号	输入 encode	输入 decode	预期输出 ok	实际输出 ok	覆盖路径
1	NULL	NULL	0	0	P1
2	Software%!Test	NULL	1	1	P2

变量：c					
Dc-path					
编号	Type	Path			
P1	Dc-path	15-16			
P2	Dc-path	15-16-20			
测试用例					
编号	输入 encode	输入 decode	预期输出 ok	实际输出 ok	覆盖路径
1	Software%!Test	NULL	1	1	P1,P2

变量：digit_high					
Dc-path					
编号	Type	Path			
P1	Dc-path	22-23-24			
P2	Dc-path	22-23-24-28			
测试用例					
编号	输入 encode	输入 decode	预期输出 ok	实际输出 ok	覆盖路径
1	Software%Test	NULL	0	0	P1,P2

变量：digit_low					
Dc-path					
编号	Type	Path			
P1	Dc-path	23-24			
P2	Dc-path	23-24-28			
测试用例					
编号	输入 encode	输入 decode	预期输出 ok	实际输出 ok	覆盖路径
1	Software%Test	NULL	0	0	P1,P2

三、实验思考

1. 通过测试，是否发现程序中存在的缺陷？

答：没有发现错误缺陷，但是程序可以进行优化，降低时间开销。

2. 谈谈数据流测试和控制流测试的区别和联系？

答：两者都需要画出程序流程图，关注程序中的节点，都需要寻找路径。区别在于，控制流测试关注于程序整体的路径覆盖与条件覆盖等，而数据流测试关注于各个变量的定义与使用路径，关注数据的变化是否符合预期输出。

3. 如何用工具来替代手工的白盒测试，你觉得这样的工具应该如何设计？设计的技术中可能的技术难点在于哪里？

答：需要考虑多种测试方案，包括控制流中的路径覆盖、条件覆盖等，基本路径覆盖和数据流测试，同时能够根据程序代码，自动生成测试用例。

难点在于如何自动生成测试用例，自动生成预期输出等。

四、实验体会

通过本次实验，我更加深入理解了数据流测试的目的与意义，动手实践过程中，对数据流测试的方法、过程有了初步了解与掌握，希望在未来的学习与工作中，能够继续学习，深入掌握，设计出更好的测试用例，做一个合格的测试工程师。