



东南大学国家示范性软件学院

College of Software Engineering
Southeast University

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： 计算机楼 268

实验日期： 2018.11.8

学生姓名： 杨昱昊

学生学号： 71116216

东南大学 软件学院 制



一、实验目的

(1) 巩固白盒测试只是，能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：控制流测试技术实验

1. 题目内容

运用数据流测试方法，对 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；

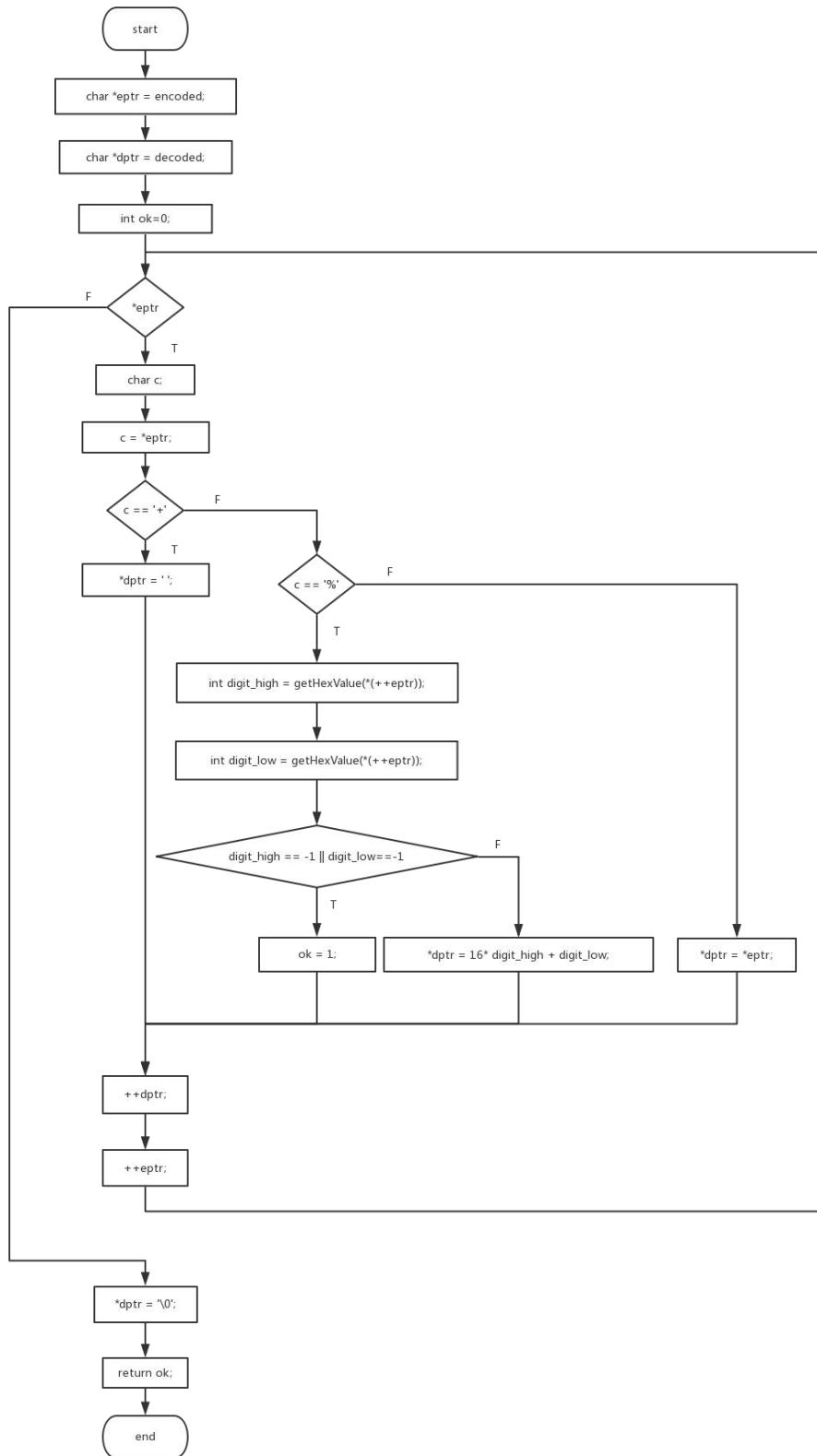


decode()函数的语句及编号如下(摘自实验指导书)

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	}

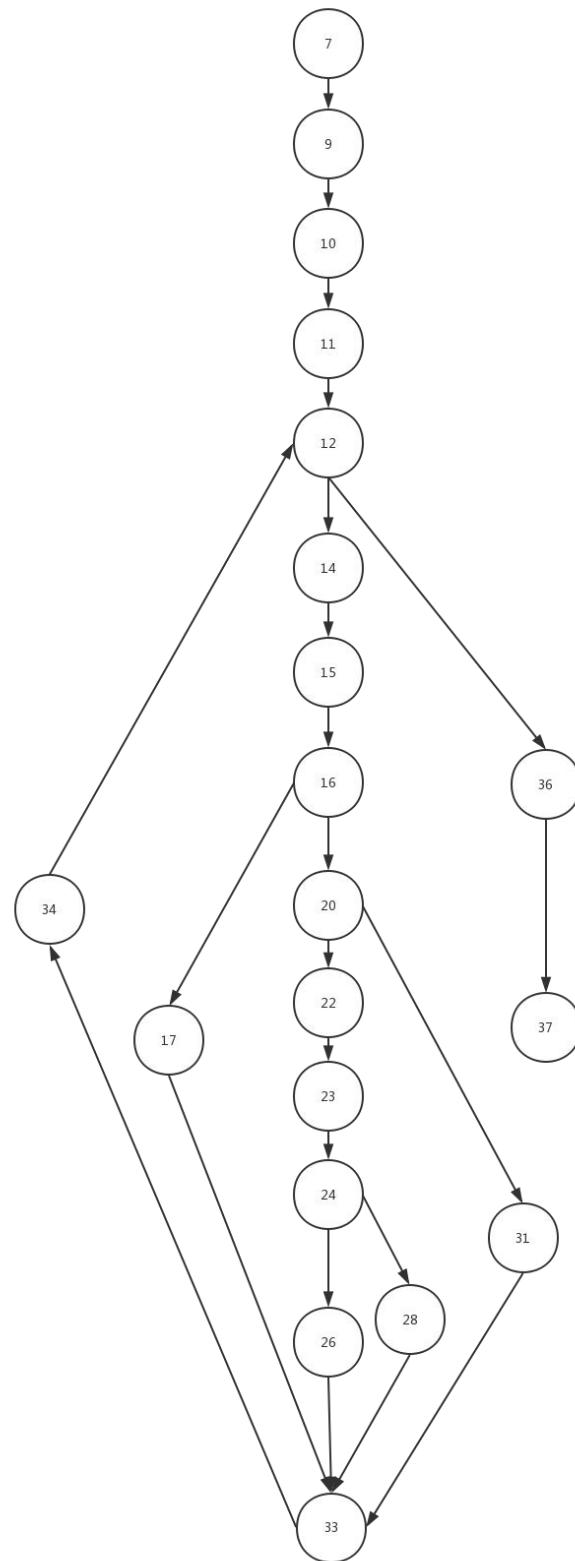


程序流程图如下





程序流图如下





2. 题目解答

encoded:

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

du-path:7-9

dc-path: 7-9

测试用例：

编号	执行条件	输入	期望输出	实际输出	路径
P1	数据流测试	(‘+’, ‘=’)	0	0	7-9

decoded:

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

du-path:7-9-10

dc-path: 7-9-10

测试用例：

编号	执行条件	输入	期望输出	实际输出	路径
P1	数据流测试	(‘+’, ‘=’)	0	0	7-9-10

*eptr:

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

du-path :

P1	9-10-11-12
P2	9-10-11-12-14-15



P3	9-10-11-12-14-15-16-20-22
P4	9-10-11-12-14-15-16-20-31
P5	9-10-11-12-14-15-16-20-22-23
P6	22
P7	22-23
P8	22-23-24-28-33-34-12
P9	22-23-24-28-33-34-12-14-15
P10	22-23-24-28-33-34-12-14-15-16-20-31
P11	23
P12	23-24-28-33-34-12
P13	23-24-28-33-34-12-14-15
P14	23-24-28-33-34-12-14-15-16-20-22
P15	23-24-28-33-34-12-14-15-16-20-31
P16	34
P17	34-12
P18	34-12-14-15
P19	34-12-14-15-16-20-22
P20	34-12-14-15-16-20-31
P21	34-12-14-15-16-20-22-23

约简：

P4	9-10-11-12-14-15-16-20-31
P5	9-10-11-12-14-15-16-20-22-23
P10	22-23-24-28-33-34-12-14-15-16-20-31
P14	23-24-28-33-34-12-14-15-16-20-22

测试用例：

编号	执行条件	输入	期望输出	实际输出	路径
P4	数据流测试	('--','=')	0	0	9-10-11-12-14-15-16-20-31
P5	数据流测试	('%-','=')	0	0	9-10-11-12-14-15-16-20-22-23
P10	数据流测试	('%-=',' =')	0	0	22-23-24-28-33-34-12-14-15-16-20-31
P14	数据流测试	('%%','=') '')	1	1	23-24-28-33-34-12-14-15-16-20-22

dc-path:

P1	9-10-11-12
P2	9-10-11-12-14-15
P3	9-10-11-12-14-15-16-20-31
P4	22



P5	23
P6	34-12
P7	34-12-14-15

约简后保留 P3, P4, P5, P7

测试用例：

编号	执行条件	输入	期望输出	实际输出	路径
P3	数据流测试	('-' , '=')	0	0	9-10-11-12-14-15-16-20-31
P4	数据流测试	('%' , '=')	1	1	22
P5	数据流测试	('+' , '=')	0	0	23
P7	数据流测试	('+' , '=')	0	0	34-12-14-15

epr:

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

du-path :

P1	9-10-11-12
P2	9-10-11-12-14-15
P3	9-10-11-12-14-15-16-20-22
P4	9-10-11-12-14-15-16-20-31
P5	9-10-11-12-14-15-16-20-22-23
P6	9-10-11-12-14-15-16-18-33-34
P7	22
P8	22-23
P9	22-23-24-28-33-34
P10	22-23-24-28-33-34-12
P11	22-23-24-28-33-34-12-14-15
P12	22-23-24-28-33-34-12-14-15-16-20-31
P13	23



P14	23-24-28-33-34
P15	23-24-28-33-34-12
P16	23-24-28-33-34-12-14-15
P17	23-24-28-33-34-12-14-15-16-20-31
P18	23-24-28-33-34-12-14-15-16-20-22
P19	34
P20	34-12
P21	34-12-14-15
P22	34-12-14-15-16-20-31
P23	34-12-14-15-16-20-22
P24	34-12-14-15-16-20-22-23

约简：

P4	9-10-11-12-14-15-16-20-31
P5	9-10-11-12-14-15-16-20-22-23
P6	9-10-11-12-14-15-16-18-33-34
P12	22-23-24-28-33-34-12-14-15-16-20-31
P18	23-24-28-33-34-12-14-15-16-20-22

测试用例：

编号	执行条件	输入	期望输出	实际输出	路径
P4	数据流测试	('--','=')	0	0	9-10-11-12-14-15-16-20-31
P5	数据流测试	('%', '=')	1	1	9-10-11-12-14-15-16-20-22-23
P6	数据流测试	('+','=')	0	0	9-10-11-12-14-15-16-18-33-34
P12	数据流测试	('%-','='))	0	0	22-23-24-28-33-34-12-14-15-16-20-31
P18	数据流测试	('%-%','=')	1	1	23-24-28-33-34-12-14-15-16-20-22

dc-path:

P1	9-10-11-12
P2	9-10-11-12-14-15
P3	9-10-11-12-14-15-16-20-31
P4	22
P5	23
P6	34

约简后得：P3,P4,P5,P6

测试用例：



编号	执行条件	输入	期望输出	实际输出	路径
P3	数据流测试	('-' , '=')	0	0	9-10-11-12-14-15-16-20-31
P4	数据流测试	('%' , '=')	1	1	22
P5	数据流测试	('%-', '=')	1	1	23
P6	数据流测试	('-' , '=')	0	0	34

*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';

由于没有 USE 节点所以没有 du-path 和 dc-path

dptr:

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

du-path :

P1	10-11-12-14-15-16-18
P2	10-11-12-14-15-16-20-22-23-24-28
P3	10-11-12-14-15-16-20-31
P4	10-11-12-14-15-16-20-31-33
P5	10-11-12-14-15-16-20-31-33-34-12-36
P6	33
P7	33-34-12-14-15-16-18
P8	33-34-12-14-15-16-20-22-23-24-28
P9	33-34-12-14-15-16-20-31
P10	33-34-12-36

约简：



P1	10-11-12-14-15-16-18
P2	10-11-12-14-15-16-20-22-23-24-28
P5	10-11-12-14-15-16-20-31-33-34-12-36
P7	33-34-12-14-15-16-18
P8	33-34-12-14-15-16-20-22-23-24-28
P9	33-34-12-14-15-16-20-31

测试用例：

编号	执行条件	输入	期望输出	实际输出	路径
P1	数据流测试	('+','=')	0	0	10-11-12-14-15-16-18
P2	数据流测试	('%-','='))	0	0	10-11-12-14-15-16-20-22-23-24-28
P5	数据流测试	('--','=')	0	0	10-11-12-14-15-16-20-31-33-34-12-36
P7	数据流测试	('-+','=')	0	0	33-34-12-14-15-16-18
P8	数据流测试	('-%-','='))	1	1	33-34-12-14-15-16-20-22-23-24-28
P9	数据流测试	('--','=')	0	0	33-34-12-14-15-16-20-31

dc-path:

P1	10-11-12-14-15-16-18
P2	10-11-12-14-15-16-20-22-23-24-28
P3	10-11-12-14-15-16-20-31
P4	33
P5	33-34-12-14-15-16-18
P6	33-34-12-14-15-16-20-22-23-24-28
P7	33-34-12-14-15-16-20-31
P8	33-34-12-36

约简得：P1,P2,P3,P5,P6,P7,P8

测试用例：

编号	执行条件	输入	期望输出	实际输出	路径
P1	数据流测试	('+','=')	0	0	10-11-12-14-15-16-18
P2	数据流测试	('%-','='))	0	0	10-11-12-14-15-16-20-22-23-24-28
P3	数据流测试	('--','=')	0	0	10-11-12-14-15-16-20-31



P5	数据流测试	('-+','=')	0	0	33-34-12-14-15-16-18
P6	数据流测试	('-%--','=')	0	0	33-34-12-14-15-16-20-22-23-24-28
P7	数据流测试	('--','=')	0	0	33-34-12-14-15-16-20-31
P8	数据流测试	(' ','=')	0	0	33-34-12-36

ok:

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

du-path :

P1	11-12-36-37
P2	26-33-34-12-36-37

测试用例：

编号	执行条件	输入	期望输出	实际输出	路径
P1	数据流测试	(' ','=')	0	0	11-12-36-37
P2	数据流测试	('%-','=')	0	0	26-33-34-12-36-37

dc-path:

P1	11-12-36-37
P2	26-33-34-12-36-37

测试用例：

编号	执行条件	输入	期望输出	实际输出	路径
P1	数据流测试	(' ','=')	0	0	11-12-36-37
P2	数据流测试	('%-','=')	0	0	26-33-34-12-36-37

C:

Node	Type	Code
14	DEF	char c;
15	DEF	c = *eptr;



16	USE	if (c == '+')
20	USE	else if (c == '%')

du-path :

P1	14-15-16
P2	14-15-16-20
P3	15-16
P4	15-16-20

约简得 : P2

测试用例 :

编号	执行条件	输入	期望输出	实际输出	路径
P2	数据流测试	('%-','=')	0	0	14-15-16-20

dc-path:

P1	15-16
P2	15-16-20

约简得 : P2

测试用例 :

编号	执行条件	输入	期望输出	实际输出	路径
P2	数据流测试	('%-','=')	0	0	15-16-20

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;

du-path :

P1	22-23-24
P2	22-23-24-28

约简得 : P2

测试用例 :

编号	执行条件	输入	期望输出	实际输出	路径



P2	数据流测试	(‘%--’, ‘=’)	0	0	22-23-24-28
----	-------	---------------	---	---	-------------

dc-path:

P1	22-23-24
P2	22-23-24-28

约简得：P2

测试用例：

编号	执行条件	输入	期望输出	实际输出	路径
P2	数据流测试	(‘%--’, ‘=’)	0	0	22-23-24-28

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;

du-path :

P1	23-24
P2	23-24-28

约简得：P2

测试用例：

编号	执行条件	输入	期望输出	实际输出	路径
P2	数据流测试	(‘%--’, ‘=’)	0	0	23-24-28

dc-path:

P1	23-24
P2	23-24-28

约简得：P2

测试用例：

编号	执行条件	输入	期望输出	实际输出	路径
P2	数据流测试	(‘%--’, ‘=’)	0	0	23-24-28



三、实验体会

谈谈数据流测试和控制流测试的区别和联系。

区别：数据流测试主要是从变量的计算和使用来判定程序是否正确，主要关注数据的定义和使用。控制流测试是根据程序的执行路径来判定程序是否正确，旨在测试程序中大量路径中的一部分路径。

联系：控制流测试是数据流测试的基础。