Test

Engineering Report 95000xxxx95000xxxx

Draft 195000xxxx Draft 1Draft 1

Date: 2018-07-06

Copyright © 2018

All Rights Reserved

**PROPRIETARY NOTICE AND DISCLAIMER**

Unless noted otherwise, this document and the information herein disclosed are proprietary to MCCI Corporation, 3520 Krums Corners Road, Ithaca, New York 14850 (“MCCI”). Any person or entity to whom this document is furnished or having possession thereof, by acceptance, assumes custody thereof and agrees that the document is given in confidence and will not be copied or reproduced in whole or in part, nor used or revealed to any person in any manner except to meet the purposes for which it was delivered. Additional rights and obligations regarding this document and its contents may be defined by a separate written agreement with MCCI, and if so, such separate written agreement shall be controlling.

The information in this document is subject to change without notice, and should not be construed as a commitment by MCCI. Although MCCI will make every effort to inform users of substantive errors, MCCI disclaims all liability for any loss or damage resulting from the use of this manual or any software described herein, including without limitation contingent, special, or incidental liability.

MCCI, TrueCard, TrueTask, MCCI Catena, and MCCI USB DataPump are registered trademarks of MCCI Corporation.

MCCI Instant RS-232, MCCI Wombat and InstallRight Pro are trademarks of MCCI Corporation.

All other trademarks and registered trademarks are owned by the respective holders of the trademarks or registered trademarks.

Copyright © 2017 by MCCI Corporation.

Document Release History

|  |  |  |
| --- | --- | --- |
| Draft 1 | 2018-07-06 | First draft |

**Table of Contents**

1 Introduction 3

1.1 Relevant Documents 3

**List of Tables**

Table 1. References, IP Blocks 3

**List of Figures**

**No table of figures entries found.**

**List of SeQUENCE Diagrams**

**No table of figures entries found.**

## Program 1

#include "stdio.h"

int main()

{

    char a[] = { 'A', 'B', 'C', 'D' };

    char\* ppp = &a[0];

    \*ppp++; // Line 1

    printf("%c %c ", \*++ppp, --\*ppp); // Line 2

}

Ans:CA

Ppp is preincrement and Ppp points to the index 1.The printf statement is executed from right to left.—ppp decrements the value at index1.And \*++ppp points to the next location C,prints it C A

## Program 2

main()

{

if(1,0)

{

printf("True");

}

Else

{

printf("False");

}

}

Ans: false

If statement evaluates the value at the right side because ()have left to right associative .if(0) so else part is executed and false is printed.

## Program 3

#### #include<stdio.h>

#### int main()

#### {

#### int a = 320;

#### char \*ptr;

#### ptr =( char \*)&a;

#### printf("%d ",\*ptr);

#### return 0;

}

Ans :64

Char pointer keep the address one byte at a time.binary value of 320 is 00000001 01000000.so pointer is pointing to first 8 bit and the decimal value is 64.

## Program 4

int f(int x, int \*py, int \*\*ppz)

{

  int y, z;

  \*\*ppz += 1;

   z  = \*\*ppz;

  \*py += 2;

   y = \*py;

   x += 3;

   return x + y + z;

}

 void main()

{

   int c, \*b, \*\*a;

   c = 4;

   b = &c;

   a = &b;

   printf( "%d", f(c,b,a));

   getchar();

}

Ans: z is changed to 5(\*\*ppz+=1). changes c to 7(\*py+=2), x is not changed. y is changed to 7. x is incremented by 3. return 7 + 7 + 5.

## Program 5

#include<stdio.h>

int main()

{ union a

{

int i;

char ch[2];

};

union a u;

u.ch[0]=3;

u.ch[1]=2;

printf("%d, %d, %d\n", u.ch[0], u.ch[1], u.i);

return 0;

}

Ans: 3,2,515

It prints the value of u.ch[0] = 3, u.ch[1] = 2 and it prints the value of u.i means the value of entire union size.

## Program 6

#include <stdio.h>

#include <string.h>

int main(void)

{

    char\* p = "geeks";

    printf("%lu %lu %lu ", sizeof(p), sizeof(\*p), sizeof("geeks"));

    printf("%lu %lu", strlen(p), strlen("geeks"));

    return 0;

}

Ans:8 1 6 5 5

p is a pointer so sizeof returns [sizeof](http://www.geeksforgeeks.org/sizeof-operator-c/)(char\*), \*p is of type char, sizeof(“geeks”) returns the number of characters including the null character, strlen(p) returns length of string pointed by p without null character and strlen returns number of characters without null character.

## Program 7

Int main()

{

int a=5,\*b,c;

b=&a;

printf(“%d”,a\* \*b \* a+ \*b);

return 0;

}

Ans:130

25\*5+5

## Program 8

int main()

{

    int i, j = 3;

    float k = 7;

    i = k % j;

    printf("%d", i);

    return (0);

}

Ans:error . % cannot be applied to floating point integer.

## Program 9

int main()

{

    int x = 5;

    int \* const ptr = &x;

    ++(\*ptr);

    printf("%d", x);

    getchar();

    return 0;

}

Ans:6

++(\*ptr )increments the value of x.hence,6

## Program 10

void fun(char\*\* str\_ref)

{

    str\_ref++;

}

int main()

{

    char \*str = (void \*)malloc(100\*sizeof(char));

    strcpy(str, "GeeksQuiz");

    fun(&str);

    puts(str);

    free(str);

    return 0;

}

Ans:GeeksQuiz

str\_ref is a local variable to fun(). When we do str\_ref++, it only changes the local variable str\_ref..

## Program 11

int main()

{

    int x = 2;

    switch (x) {

        x--;

        switch (x) {

        case 1:

            printf("Hello");

            break;

        case 2:

            printf("GFG");

            break;

        case 3:

            printf("Welcome");

            break;

        default:

            printf("BYE");

        }

    }

    return (0);

}

Ans:No output. Since there is no outer switch case.no output.

## Program 12

int main()

{

    int x = 2;

    switch (x) {

    case 1:

        printf("Hello");

        break;

    case 2:

        printf("Hello");

        break;

    case 3:

        printf("GFG");

        break;

    default:

        printf("BYE");

    }

    return (0);

}

Ans:Error.Since the switch case for both the cases are same,it shows error.

## Program 13

int main()

{

    static int i=5;

    if(--i){

        main();

        printf("%d ",i);

    }

}

Ans: 0 0 0 0

Since i is a static variable and is stored in Data Section, all calls to main share same i.

## Program 14

int fun()

{

  static int num = 16;

  return num--;

}

int main()

{

  for(fun(); fun(); fun())

    printf("%d ", fun());

  return 0;

}

Ans:14,11,8,5,2

Initially num value is 16. When initialization num value becomes 15.in the condition part num value is checked for non zero value and it is decremented to 14.When printf is called further decremented to 13 after printing 14.Likewise the loop continues till the num becomes zero.

## Program15:

void func()

{

printf("Hi");

}

int main()

{

int a=10;

printf("%d",sizeof(a=(3.4+8)));

printf("%d",a);

printf("%d",sizeof(3.8+4));

printf("%d",sizeof((char)(float)(int)(3.8-4)));

printf("%d",sizeof(func()));

printf("%d",sizeof(FILE));

return 0;

}

Ans:

Sizeof(a=(3.4+8)) is 4

A=10

Sizeof(3.8+4) is 8

Sizeof((char)(float)(int) (3.8-4)) is 1

Sizeof(func() is error in visual studio but other compiler give 1 as the output.

Sizeof(FILE) is 4.

## Program16:

union foo

{

double a;

char \*b;

union data

{

char name[10];

int val;

}d;

};

int main()

{

union foo a;

printf("%d",sizeof(a));

printf("%d",sizeof(a.d));

return 0;

}

Ans:

Sizeof(a) is 12

Sizeof(a.d) is 16

Padding occurs in this program.

## Program17:

int main()

{

int a = 10;

printf("%d %d %d", a, ++(a)\*a++/a\*++a, a \*= a + 5);

return 0;

}

Ans: 153 23104 153

a \*=a + 5 is 150 then ++a \*a++/a \* ++a is executed.

((152 \*151)/151)\*152 so the answer is 23104.

a is 153.

## Program18:

int main()

{

int a = 300;

char \*b = (char \*)&a;

\*++b = 2;

printf("%d ",a);

return 0;

}

Ans: it store 00010110 0000001

b is point to 00010110 pointer position is increament and store the value as 2

so it is replace 00010110 00000010

a=556

## Program19:

int main()

{

int a=10;

int b=5;

a>b?a=4:b;

a<b?a:b=20;

printf("%d %d",a,b);

return 0;

}

Ans: Error in line 6.

a>b ? a=4: b; a=4;

a<b a:b=20; it execute in (a<b a:b)=20; It show lvalue required error

a<b ? a : (b = 20); change the statement like this.it is not show error.

## Program20:

int main()

{

int a=528;

printf("%d",a>>1<<2>>4<<1);

return 0;

}

Ans: 132

00000010 00010000>>1 = 00000001 00001000

00000001 00001000<<2 = 00000100 00100000

00000100 00100000>>4 = 00000000 01000010

00000000 01000010<<1 = 00000000 10000100

## Program21:

int main()

{

int a=5;

a=(a,--a,a--,a=a\*10,++(a--));

printf("%d",a);

return 0;

}

Ans: Error

++(a--) in this statement ++(constant) so it give modifiable lvalue expression error.

## Program22:

int main()

{

struct node

{

int a;

int b;

union foo

{

double c;

int c1;

}e;

struct node \*data;

};

struct node fir = { 1,2,4 };

struct node sec = { 2,3,5 };

struct node thir = { 4,5,6 };

fir.data = &thir;

sec.data = &fir;

thir.data = &thir;

printf("%d %d %d", thir.data->a, sec.data->data->data->data->a++,(int) fir.data->e.c);

return 0;

}

Ans:5 4 6

## Program23:

int main()

{

int a = 10;

int b1 = 20;

int \*a1;

int \*\*b;

a1 = &a;

b = &a1;

\*b = (&b1);

printf("%d %d %d", a, \*a1, \*\*b);

}

Ans: 10 20 20

a=10 b=20

a1 point to address of a;

b point to address of a1;

\*b is point to address of b1.

## Program24:

int main()

{

int a = 10;

int \* const b;

const int \*c=12;

\*b = a;

b = &a;

c = &a;

\*c=a;

printf("%d %d",\*b,\*c);

}

Ans : Error

Line 7 :can’t change the address b.

Line 9:can’t change the value of c.

## Program 25

int main()

{

char s[10]="Sun";

char a1[10];

a1=s;

printf("%s",s1);

}

Ans:Error :expression must be modifiable lvalue.

Line 5: address of array value modifiable.

## Program26

int f(int x, int \*py, int \*\*ppz)

{

int y, z;

\*\*ppz += 1;

z = \*\*ppz;

\*py += 2;

y = \*py;

x += 3;

return x + y + z;

}

int main()

{

int c, \*b, \*\*a;

c = 4;

b = &c;

a = &b;

printf("%d ", f(c, b, a));

return 0;

}

Ans: 19

In function f

\*\*ppz += 1; so \*\*ppz=5

Z=5

\*pz =5;

\*pz += 2; so,\*pz=7;

Y=7

X=4;

X+=3;

X=7;

So the function return 19

## Program 27

void f(char \*\*p)

{

char \*t;

t = (p += sizeof(int))[-1];

printf("%sn", t);

}

int main()

{

char \*argv[] = { "ab", "cd", "ef", "gh", "ij", "kl" };

f(argv);

return 0;

}

Ans: gh

T=(p+=sizeof(int))[-1]

P is point to first element of array

P+=sizeof(int) then point to 5th element of array

P[-1] is decrease the array element by 1.

t is point to 4th element of array.

## Program 28

int main()

{

int a=5;

int b=a++&&--a||a++;

printf("%d %d",a,b);

return 0;

}

Ans:5 1

(a++ && --a) is 5&&4 it give the 1

1||a++ is return 1

b=1;

## Program 29

void foo(int a)

{

char \*s = "012345678";

int b;

b = a % 8;

a = a / 8;

if (a > 0)

foo(a);

printf("%c", s[b]);

}

int main()

{

int a = 12897;

foo(a);

return 0;

}

Ans:31141

It give octal number of 12897.