Instituto Tecnológico de Costa Rica

Escuela de Computación

Lenguajes de Programación

Grupo 40

Tarea Programada 3 - Problemas C

Profesor (a):

Jose Rafael Castro Mora

Estudiante (s):

José Adrián Amador Ávila - 2016101574

San José, I Semestre 2023

Pregunta 1

```
10
11  void pregunta1() { // Aritmética básica
12      int x;
13      x = - 3 + 4 * 5 - 6; PRINTX;
14      x = 3 + 4 % 5 - 6; PRINTX;
15      x = - 3 * 4 % - 6 / 5; PRINTX;
16      x = ( 7 + 6 ) % 5 / 2; PRINTX;
17  }
18
```

Salida

```
estudiante@Latitude-5491:~/Escritorio/Computacion/2023/ISemestre/Lenguajes/Tarea3$ ./problemas
11
1
0
1
```

1. Para la primera operación $x = -3 + 4 * 5 - 6$, siguiendo la precedencia de operadores, se realizaría primero la multiplicación $4 * 5 = 20$, luego $-3 + 20 = 17$, y por útlimo $17 - 6 = 11$, dando como resultado $x = 11$..
2. La segunda operación $x = 3 + 4 \,\%\, 5 - 6$, primero el operador **%** tiene mayor precedencia por lo tanto $4 \,\%\, 5 = 4$, luego $3 + 4 = 7$, y por último, $7 - 6 = 1$, dando como resultado $x = 1$..
3. La tercera operación $x = -3 * 4 \,\%\, -6 / 5$, para esta operación los operadores **%**, * y / tienen la misma precedencia y se ejecutan de izquierda a derecha, por lo tanto $-3 * 4 = -12$, luego $-12 \,\%\, -6 = 0$, y por último, $0 / 5 = 0$, dando como resultado $x = 0$.
4. La cuarta operación $x = (7 + 6) \,\%\, 5 / 2$, primero los paréntesis **()** tienen mayor precedencia, por lo tanto $(7 + 6) = 13$, luego $13 \,\%\, 5 = 3$, y por último, $3 / 2 = 1$, dando como resultado $x = 1$.

Pregunta 2

```
19  void pregunta2() { // Asignación
20      int x = 2, y, z;
21
22      x *= 3 + 2; PRINTX;
23      x *= y = z = 4; PRINTX;
24      x = y == z; PRINTX;
25      x == (y = z); PRINTX;
26  }
27
```

Salida

```
estudiante@Latitude-5491:~/Escritorio/Computacion/2023/ISemestre/Lenguajes/Tarea3$ ./problemas
10
40
1
1
```

1. Para la primera asignación x *= $3 + 2$, los operadores de asignación simple y compuesta tiene la prioridad más baja, luego de la evaluación secuencial, por lo tanto primero tenemos que $3 + 2 = 5$, y luego tenemos x *= 5, sabemos que $x = 2$, por lo tanto primero se multiplican 5 y 2, dando como resultado 10 y se asigna a la variable x.
2. Segunda asignación x *= $y = z = 4$, todos los operadores de asignación tienen la misma precedencia y van de derecha a izquierda, por lo tanto en este caso, primero se asigna 4 a la variable z, luego se le asigna el valor de z a la variable y, y por último se multiplica el valor de la variable y con el valor de la variable x, $y = 4$ y $x = 10$, por lo tanto $x = 40$.
3. Tercera asignación $x = y == z$, acá el operador de igualdad tiene mayor precedencia por lo tanto se resuelve primero antes de asignar el valor a x. La expresión $y == z$ da como resultado "true" que es representado como un 1, luego ese 1 es asignado a la variable x, por lo tanto $x = 1$.
4. Cuarta asignación $x == (y = z)$, acá los paréntesis tienen mayor precedencia que el operador de igualdad, por lo tanto se resuelve primero. Dentro de los paréntesis se asigna el valor de z a la variable y, por lo cual tendríamos que $y = 4$, luego se compara si x es igual a y, tenemos que $x = 4$, por lo tanto es "true", dando como resultado el 1 que se imprime en pantalla.

Pregunta 3

```
29
30  void pregunta3() { // Lógica y operadores de incremento
31      int x, y, z;
32
33      x = 2; y = 1; z = 0;
34      x = x && y || z; PRINT(x);
35
36      x = y = 1;
37      z = x ++ - 1; PRINT(x), PRINT(z);
38      z += - x ++ + ++ y; PRINT(x); PRINT(z);
39      z = x / ++x; PRINT(z);
40  }
41
```

Salida

```
emas
estudiante@Latitude-5491:~/Escritorio/Computacion/2023/ISemestre/Lenguajes/Tarea3$ ./problemas
int = 1
int = 2
int = 0
int = 3
int = 0
int = 1
```

1. Para la primera operación $x = x \,\&\&\, y \,||\, z$, acá los operadores lógicos tienen mayor precedencia que el operador de asignación, además, el AND lógico (&&) tiene mayor precedencia que el OR lógico (||). Primero tenemos la expresión $x \,\&\&\, y$, $x = 2$ e $y = 1$, el resultado de esta operación es 1, luego ese 1 se resuelve con el OR lógico $1 \,||\, z$, como $z = 0$, el resultado de esa operación también da 1, y ese 1 se asigna a la variable x, $x = 1$, lo cual es lo que se imprime en pantalla.
2. Ahora se asigna $x = y = 1$, por lo tanto x e y tienen un valor de 1.
3. La segunda operación $z = x \;++\; - 1$, acá el operador postfijo ++ tiene mayor precedencia, no obstante, como es un post-incremento, la variable x no se ve afecta hasta después de ser evaluada dentro de la expresión, por lo tanto seguimos teniendo que $x = 1$ hasta que la expresión sea evaluada, ahora se procede a realizar la resta dando como resultado $z = 0$. Ahora bien, ya que la expresión fue evaluada y el valor de dicha expresión asignado, entonces se procede a incrementar el valor de x en 1, por lo tanto tenemos que $x = 2$, es por eso que se imprime int = 2 e int = 0, siendo los valores de x y z respectivamente.
4. La tercera operación $z \mathrel{+}= \; - x \;++ \; + \; ++ y$, primero el operador prefijo ++ incrementa el valor de y antes de ser usado dentro de la expresión, así, tenemos que $y = 2$, luego se evalúa la

expresión $-\ x\ +\ y$ (el operador postfijo ++ lo ignoramos por ahora ya que no se aplica hasta que la operación sea resuelta) con $x\ =\ 2$ e $y\ =\ 2$, dando como resultado 0, el cual es evaluado en $z\ +=\ 0$, como $z\ =\ 0$, la suma de $0\ +\ 0\ =\ 0$ y 0 se asigna a la variable z, por último, se incrementa en 1 la variable x, por lo tanto tenemos que $x\ =\ 3$. Los valores que se imprimen de 3 y 0 son la variable x y la variable z respectivamente.

5. Para la última expresión tenemos $z\ =\ x/\ ++\ x$, primero se evalúa el operador prefijo ++ lo cual suma 1 a la variable x, ahora tenemos que $x\ =\ 4$, luego se evalúa $x\ /\ x$ lo cual da como resultado 1, y dicho valor se asigna a la variable z. El valor de z es imprimido en pantalla, siendo este el valor 1.

Pregunta 4

```
42  void pregunta4() { // Operadores de bits
43      int x, y, z;
44
45      x = 03; y = 02; z = 01;
46      PRINT( x | y & z );
47      PRINT( x | y & - z );
48      PRINT( x ^ y & - z );
49      PRINT( x & y && z);
50
51      x = 1; y = -1;
52      PRINT( ! x | x );
53      PRINT( - x | x );
54      PRINT( x ^ x );
55      x <<= 3; PRINT(x);
56      y <<= 3; PRINT(y);
57      y >>= 3; PRINT(y);
58  }
```

Salida

```
estudiante@Latitude-5491:~/Escritorio/Computacion/2023/ISemestre/Lenguajes/Tarea3$ ./problemas
int = 3
int = 3
int = 1
int = 1
int = 1
int = -1
int = 0
int = 8
int = -8
int = -1
estudiante@Latitude-5491:~/Escritorio/Computacion/2023/ISemestre/Lenguajes/Tarea3$
```

1. Para la primera operación $x \mid y \,\&\, z$, el operador "&" tiene mayor precedencia por lo cual es evaluado primero, tenemos que $y = 02$ y $z = 01$, por lo cual $02 \,\&\, 01 = 00$, ahora $x \mid 00 = 03$ ya que $x = 03$. Se imprime en pantalla 3.
2. La segunda operación $x \mid y \,\&\, -z$, tenemos que $y = 02$ y $z = 01$, al negar z obtenemos $z = 1111\ 1111$ en forma binaria, al aplicar AND bit a bit (&) obtenemos como resultado $y \,\&\, -z = 02$, ahora pasamos con $x \mid 02$, tenemos que $x = 03$, entonces $03 \mid 02 = 03$. Se imprime en pantalla 3.
3. La tercera operación $x \,\hat{}\, y \,\&\, -z$, el AND bit a bit (&) tiene mayor precedencia que el OR exclusivo (^), entonces resolvemos primero $y \,\&\, -z$, que sabemos que dá como resultado 02, ahora pasamos a $x \,\hat{}\, 02$, con $x = 03$, por lo tanto tenemos como resultado 01. Se imprime en pantalla 1.
4. La cuarta operación $x \,\&\, y \,\&\&\, z$, el AND bit a bit (&) tiene mayor precedencia que el AND lógico (&&), por lo tanto se resuelve primero $x \,\&\, y$ con $x = 03$ e $y = 02$, dando como resultado 03, ahora tenemos 03 && z con $z = 01$, dando como resultado 1. Se imprime 1 en pantalla.
5. La quinta operación ! $x \mid x$, el operador NOT lógico (!) tiene mayor precedencia, tenemos que $x = 1$, por lo tanto al negarlo se obtiene 0, ahora $0 \mid 1 = 1$. Se imprime 1.
6. La sexta operación $-\ x \mid x$, tenemos que $x = 1$, entonces $-1 \mid 1 = -1$, ya que al aplicar el OR inclusivo bit a bit ("|") a cada bit entre los dos números tenemos como resultado 1111, el cual representa al -1. Se imprime -1.
7. La séptima operación $x \,\hat{}\, x$, tenemos que $x = 1$, entonces se imprime 0, ya que el operador OR exclusivo bit a bit (^) solo da 1 cuando ambos datos son diferentes, al tener x como 0001 y aplicar la el OR exclusivo, todos sus bits da 0.
8. La octava operación $x <<= 3$, acá se realiza primero el desplazamiento a la izquierda de bits y luego se asigna el resultado a la variable x, tenemos que $x = 1$, en su forma binaria sería $x = 0001$, si desplazamos a la izquierda 3 bits entonces obtenemos 1000, se asigna a x y ahora tenemos que $x = 8$. Se imprime 8 en pantalla.
9. La novena operación $y <<= 3$, se realiza un desplazamiento de bits a la izquierda, tenemos que $y = -1$ o en su forma binaria $y = 1111$, si desplazamos 3 bits a la izquierda, tendríamos …1111 1000, lo que equivale a -8, se asigna a y, por lo tanto $y = -8$. Se imprime -8 en pantalla.
10. La última operación $y >>= 3$, realiza un desplazamiento de bits a la derecha, tenemos que $y = -8$ o en su forma binaria $y = \ldots 1111\ 1000$, desplazamos 3 bits a la derecha, entonces tenemos …1111, lo cual equivale a -1, se asigna a la variable y. Se imprime -1.

Pregunta 5

```
60  void pregunta5() { // Operadores relacionales y condicionales
61      int x=1, y=1, z=1;
62
63      x += y += z;
64      PRINT( x < y ? y : x );
65
66      PRINT( x < y ? x ++ : y ++ );
67      PRINT(x); PRINT(y);
68
69      PRINT( z += x < y ? x ++ : y ++ );
70      PRINT(y); PRINT(z);
71
72      x=3; y=z=4;
73      PRINT( (z >= y >= x) ? 1 : 0);
74      PRINT( z >= y && y >= x );
75  }
76
```

Salida

```
estudiante@Latitude-5491:~/Escritorio/Computacion/2023/ISemestre/Lenguajes/Tar
estudiante@Latitude-5491:~/Escritorio/Computacion/2023/ISemestre/Lenguajes/Tar
int = 3
int = 2
int = 3
int = 3
int = 4
int = 4
int = 4
int = 0
int = 1
estudiante@Latitude-5491:~/Escritorio/Computacion/2023/ISemestre/Lenguajes/Tar
```

1. Primero tenemos la operación $x \mathrel{+}= y \mathrel{+}= z$, lo cual primero suma y con z, asigna el valor a la variable y, y luego suma x con y, y guarda el valor en x. Entonces $y \mathrel{+}= z$ con $y = 1$ y $z = 1$ da como resultado $y = 2$. Ahora tenemos $x \mathrel{+}= y$ con $x = 1$ e $y = 2$ da como resultado 3, el cual se asigna a la variable x. De esto se obtiene lo siguiente: $x = 3,\ y = 2,\ z = 1$.
2. La segunda operación $x < y\ ?\ y : x$, es una expresión condicional que evalúa si la variable x es menor a la variable y, de ser verdadero devuelve y, de lo contrario devuelve x. Como $x = 3$ e $y = 2$, x no es menor a y, por lo tanto devuelve x y se imprime un 3.

3. La tercera operación $x < y\,?\,x\ ++\ :\ y\ ++$, evalúa si la variable x es menor a la variable y, $x = 3$ e $y = 2$, por lo tanto es falso, entonces devuelve y ++. Se imprime 2 y luego suma 1 a la variable y, quedando que $y = 3$.
4. Se imprime la variable x y la variable y con $x = 3$ e $y = 3$.
5. La cuarta operación $z\ +=\ x < y\,?\,x\ ++\ :\ y\ ++$, se evalúa primero la expresión condicional, la variable x no es menor a la variable y, por lo tanto es falso y se devuelve y ++. Ahora tenemos $z\ +=\ y\ ++$ con $z = 1$ e $y = 3$, se suma la variable z con la variable y, y se asigna el resultado en z, la suma da 4, por lo tanto $z = 4$. Se imprime 4 y luego, se suma 1 a y, por lo tanto $y = 4$.
6. Se imprime la variable y con $y = 4$ y la variable z con $z = 4$.
7. Ahora tenemos que $x = 3$ e y las variables $y = z = 4$.
8. La quinta operación $(z\ >=\ y\ >=\ x)\,?\,1 : 0$, la condición primero evalúa si z es mayor o igual a y, el cual da verdadero o un 1, ahora tenemos que $1\ >= x$, el cual da falso porque 1 no es mayor o igual a 3, por lo tanto se imprime 0.
9. La sexta operación $z\ >=\ y\ \&\&\ y\ >=\ x$, como el operador de mayor igual tiene mayor precedencia tendríamos algo como lo siguiente $(z\ >=\ y)\ \&\&\ (y\ >=\ x)$, primero z es mayor o igual a y lo cual es cierto (1), segundo y es mayor a igual a x lo cual también es cierto (1), entonces tenemos 1 && 1 que dá como resultado 1. Se imprime 1 en pantalla.

Pregunta 6

```
79  void pregunta6() { // Precedencia de operadores y evaluación
80      int x, y, z;
81
82      x = y = z = 1;
83      ++x || ++y && ++z; PRINT3(x,y,z);
84
85      x = y = z = 1;
86      ++x && ++y || ++z; PRINT3(x,y,z);
87
88      x = y = z = 1;
89      ++x && ++y && ++z; PRINT3(x,y,z);
90
91      x = y = z = -1;
92      ++x && ++y || ++z; PRINT3(x,y,z);
93
94      x = y = z = -1;
95      ++x || ++y && ++z; PRINT3(x,y,z);
96
97      x = y = z = -1;
98      ++x && ++y && ++z; PRINT3(x,y,z);
99  }
```

Salida

```
x=2     y=1     z=1
x=2     y=2     z=1
x=2     y=2     z=2
x=0     y=-1    z=0
x=0     y=0     z=-1
x=0     y=-1    z=-1
```

1. Primero tenemos que tanto x, y, z tiene un valor de 1.
2. La primera operación $++x \,||\, ++y \,\&\&\, ++z$, primero evalúa el && el cual devuelve verdadero o 1, luego evalúa el || el cual devuelve el ++x, por lo tanto se imprime x=2, y=1 y z=1.
3. Volvemos a dejar todas la variables igualadas a 1.
4. La segunda operación $++x \,\&\&\, ++y \,||\, ++z$, primero evalúa el && el cual da verdadero y luego evalúa el || el cual devuelve la primera expresión, dando como resultado x=2, y=2 y z=1.
5. Se vuelve a igual a 1 todas las variables.
6. La tercera operación $++x \,\&\&\, ++y \,\&\&\, ++z$, evalúa todas las expresiones las cuales dan verdadero y las devuelve, por lo tanto se imprime x=2, y=2 y z=2.

7. Ahora igualamos todas las variables a -1.
8. La cuarta operación $++x \;\&\&\; ++y \;||\; ++z$, primero hace ++x entonces x=0, trata de evaluar la expresión $x \;\&\&\; ++y$ la cual falla ya que da falso porque x=0, luego evalúa $0 \;||\; ++z$, como z=-1 entonces devuelve ++z, lo que deja z=0, y se imprime x=0, y=-1 y z=0.
9. Volvemos a dejar las variables en -1.
10. La quinta operación $++x \;||\; ++y \;\&\&\; ++z$, primero evalúa ++y entonces y=0, ahora $y \;\&\&\; ++z$ lo cual falla ya que y=0, entonces ahora evalúa $++x \;||\; 0$ lo cual devuelve ++x entonces x=0, por lo tanto imprime x=0, y=0 y z=-1.
11. Volvemos a dejar las variables en -1.
12. La sexta operación $++x \;\&\&\; ++y \;\&\&\; ++z$, primero evalúa ++x entonces x=0, ahora evalúa $x \;\&\&\; ++y$ lo cual falla ya que x=0, ahora evalúa $0 \;||\; ++z$ lo cual falla igualmente, por lo tanto se imprime x=0, y=-1 y z=-1.

Pregunta 7

```
108  char input7[] = "SSSWILTECH1\1\11W\1WALLMP1";
109
110  void pregunta7() { // switch, break continue
111      int i, c;
112
113      for ( i=2; (c=input7[i])!='\0'; i++) {
114          switch (c) {
115              case 'a' : putchar('i'); continue;
116              case '1' : break;
117              case  1  : while( (c=input7[++i])!='\1' && c!='\0') ;
118              case  9  : putchar('S');
119              case 'E' : case 'L': continue;
120              default: putchar(c); continue;
121          }
122          putchar(' ');
123      }
124      putchar('\n');
125  }
```

Salida

```
estudiante@Latitude-5491:~/Escritorio/Computacion/2023/I
SWITCH SWAMP
estudiante@Latitude-5491:~/Escritorio/Computacion/2023/I
```

Este código realiza un ciclo que comienza en la posición 2 del char input7[], es decir, comenzaría en la tercera S de la cadena de caracteres. Dentro del ciclo se realiza un switch que compara el valor que toma la variable c con respecto a evaluar input7[] en el índice i. Estos valores son enteros. Primero tenemos que i=2, entonces $c = input7[2] => c = 83$. Los casos evalúan chars y tenemos que $'a' = 97$, $'1' = 49$, $'E' = 69$, $'L' = 76$, entonces el primer caso es si c==97, segundo si c==49, tercero si c==1, cuarto si c==9, quinto si c==69 o si c==76 y por último el caso por defecto.

- Para el primer ciclo el switch entra en el caso por defecto y hace un putchar(c) lo que coloca una S en pantalla y continua con el siguiente ciclo.
- El segundo ciclo $i = 3$, $c = input7[3] => c = 87$, entonces entra en el caso por defecto, hace un putchar(c) que coloca una W en pantalla y continua con el siguiente ciclo.
- El tercer ciclo $i = 4$, $c = input7[4] => c = 73$, entonces entra en el caso por defecto, hace un putchar(c) que coloca una I en pantalla y continua con el siguiente ciclo.
- El cuarto ciclo $i = 5$, $c = input7[5] => c = 76$, entonces entra en el caso de que si c==69 o c==76, y continúa con el siguiente ciclo.
- El quinto ciclo $i = 6$, $c = input7[6] => c = 84$, entonces entra en el caso por defecto, hace un putchar(c) que coloca una T en pantalla y continua con el siguiente ciclo.
- El sexto ciclo $i = 7$, $c = input7[7] => c = 69$, entonces entra en el caso c==69 o c==76 y continua con el siguiente ciclo.
- El séptimo ciclo $i = 8$, $c = input7[8] => c = 67$, entonces entra en el caso por defecto, hace un putchar(c) que coloca una C en pantalla y continua con el siguiente ciclo.
- El octavo ciclo $i = 9$, $c = input7[9] => c = 72$, entonces entra en el caso por defecto, hace un putchar(c) que coloca una H en pantalla y continua con el siguiente ciclo.
- El noveno ciclo $i = 10$, $c = input7[10] => c = 49$, entonces entra en el caso de si c==49 el cual hace un break del switch, siguiendo con la siguiente sentencia dentro del for la cual es putchar(' ').
- El décimo ciclo $i = 11$, $c = input7[11] => c = 1$, entonces entra en el caso c==1, el cual ejecuta el loop $while((c = input7[++ i])! = '\1' \&\& c! = '\0')$. Este ciclo evalúa:
    - Con $i = 12$, $c = input7[12] => c = 9$ el cual es diferente de '\1' y diferente de '\0'.
    - Con $i = 13$, $c = input7[13] => c = 87$ que es diferente de las condiciones.
    - Con $i = 14$, $c = input7[14] => c = 1$ el cual no cumple que sea diferente de '\1', por lo tanto el ciclo termina.

    - Continua las comparaciones con los demás casos del switch, entra el caso cuando es 9, hace un putchar('S') el cual coloca una S en pantalla y luego entra en el caso 'E' or 'L' el cual continúa con el siguiente ciclo del for.

- El onceavo ciclo $i = 15$, $c = input7[15] \Rightarrow c = 87$, entonces entra en el caso por defecto, hace un putchar(c) el cual coloca un W en pantalla y continua con el siguiente ciclo.
- El doceavo ciclo $i = 16$, $c = input7[16] \Rightarrow c = 65$, entonces entra en el caso por defecto, hace un putchar(c) cual cual coloca una A en la pantalla y continua con el siguiente ciclo.
- El treceavo ciclo $i = 17$, $c = input7[17] \Rightarrow c = 76$, entonces entra en c==69 or c==76 y continua con el siguiente ciclo.
- El 14° ciclo $i = 18$, $c = input7[18] \Rightarrow c = 87$, vuelve a entrar en c==69 or c==76 y continua con el siguiente ciclo.
- El 15° ciclo $i = 19$, $c = input7[19] \Rightarrow c = 77$, entra en el caso por defecto, coloca una M en pantalla y continua con el siguiente ciclo.
- El 16° ciclo $i = 20$, $c = input7[20] \Rightarrow c = 80$, entra en el caso por defecto, coloca una P en pantalla y continua con el siguiente ciclo.
- El 17° ciclo $i = 21$, $c = input7[21] \Rightarrow c = 49$, entra en el caso c==49, hace un break del switch, luego se hace putchar(' ') y se sale del for ya que se llegó al final de la cadena de caracteres.

Pregunta 8

```
127  int a8[] = {0,1,2,3,4};
128
129  void pregunta8() { // Vectores y punteros simples
130      int i, *p;
131
132      for( i=0; i <=4; i++ ) PR(d,a8[i]);
133      NL;
134      for( p= &a8[0]; p<=&a8[4]; p++)
135          PR(d,*p);
136      NL; NL;
137
138      for ( p=&a8[0],i=1; i <=5; i++ )
139          PR(d,p[i]);
140      NL;
141      for( p=a8,i=0; p+1<=a8+4; p++,i++ )
142          PR(d,*(p+i));
143      NL; NL;
144
145      for ( p=a8+4; p>=a8; p--) PR(d,*p);
146      NL;
147      for ( p=a8+4,i=0; i<=4; i++ ) PR(d,p[-i]);
148      NL;
149      for ( p=a8+4; p >=a8; p-- ) PR(d,a8[p-a8]);
150      NL;
151  }
152
```

1. Primero tenemos $for(\ i\ =\ 0;\ i\ <=\ 4;\ i\ ++\ )\ PR(d, a8[i])$ esto imprime todos los valores dentro de la variable a8[] evaluada en i, la cual tiene como rango {0, 1, 2, 3, 4}.
2. Segundo tenemos $for(\ p\ =\ \&a8[0];\ p\ <=\ \&a8[4];\ p\ ++)\quad PR(d, *\ p)$ esto imprime el valor que está almacenado en la dirección de memoria que apunta p en cada uno de los ciclos, el ciclo for itera a través de los elementos de la variable a8[] accediendo a ellos por medio de sus direcciones de memoria.
3. Tercero tenemos $for\ (\ p\ =\ \&a8[0], i\ =\ 1;\ i\ <=\ 5;\ i\ ++\ )\quad PR(d, p[i])$ la variable p se le asigna la primera dirección de memoria del vector a8[], al realizar $p[i]$ es equivalente a $*\ (p\ +\ i)$ en donde vamos a ir accediendo a los elementos del vector en la posición i. Entonces, se imprime en pantalla los elementos desde el 1 hasta el 4 del vector a8[] pero al tratar de acceder a $p[5]$ cuando i=5 se imprime un 0 ya que las demás posiciones fueron inicializadas en 0 dentro del vector.

4. Cuarto tenemos $for(\ p = a8, i = 0;\ p + 1 <= a8 + 4;\ p\ ++, i\ ++\ )\quad PR(d, *\ (p + i))$, acá se le asigna la dirección de memoria a la variable p del vector a8, este ciclo for termina cuando p+1 sea mayor al final del vector (a8+4), tanto p como la variable i se incrementan en 1 por cada ciclo. Entonces primero se imprime * $(p + i)$ con i=0 que vendría siendo la primera posición del vector, es decir, se imprime un 0, la variable i va tomando los valores de 1, 2, y 3 lo que hace que se imprima 2, 4 y 0 respectivamente.
5. Quinto tenemos $for\ (\ p = a8 + 4;\ p >= a8;\ p --)\ PR(d, *\ p)$, en este ciclo for se recorre los valores del vector por medio de la dirección de memoria en reversa, comenzando con $p = a8 + 4$ que equivale a decir que p apunta a la dirección de memoria del cuarto valor del vector a8, este ciclo se repite hasta que p apunte la dirección de memoria del primer valor del vector a8, entonces se imprime 4, 3, 2, 1 y 0 respectivamente.
6. Sexto tenemos $for\ (\ p = a8 + 4, i = 0;\ i <= 4;\ i\ ++\ )\ PR(d, p[-\ i])$, acá se le asigna a la variable el vector a8 en la última posición del mismo, se inicializa la variable i en 0 y se aumenta en 1 hasta que i sea 4. Al hacer p[-i] en cada iteración es lo mismo que decir *(p-i), entonces se va accediendo a los valores del vector de derecha a izquierda, por lo tanto se imprime 4, 3, 2, 1 y 0 respectivamente.
7. Séptimo tenemos $for\ (\ p = a8 + 4;\ p\ >= a8;\ p\ --\ )\ PR(d, a8[p\ -\ a8])$, acá se le asigna al puntero p la dirección del último elemento de a8 con a8+4, se hace una iteración hasta que p apunte a la primera posición del vector a8, con a8[p-a8] se accede a la posición del vector dado por la diferencia entre el valor p y el valor a8, el cual sería 4, 3, 2, 1, 0 respectivamente, por lo tanto se imprime 4, 3, 2, 1, 0, lo que significa que se recorre el vector a8 y se imprime cada uno de sus valores de derecha a izquierda.

Pregunta 9

```
152
153  int a9[]={0,1,2,3,4};
154  int *p9[] = {a9,a9+1,a9+2,a9+3,a9+4};
155  int **pp9=p9;
156
157  void pregunta9() { // vectores de punteros
158      PR2(d,a9,*a9);
159      PR3(d,p9,*p9,**p9);
160      PR3(d,pp9,*pp9,**pp9);
161      NL;
162
163      pp9++; PR3(d,pp9-p9,*pp9-a9,**pp9);
164      *pp9++; PR3(d,pp9-p9,*pp9-a9,**pp9);
165      *++pp9; PR3(d,pp9-p9,*pp9-a9,**pp9);
166      ++*pp9; PR3(d,pp9-p9,*pp9-a9,**pp9);
167
168      pp9=p9;
169      **pp9++; PR3(d,pp9-p9,*pp9-a9,**pp9);
170      *++*pp9; PR3(d,pp9-p9,*pp9-a9,**pp9);
171      ++**pp9; PR3(d,pp9-p9,*pp9-a9,**pp9);
172  }
```

1. Primero tenemos $PR2(d, a9, * a9)$, el cual imprime la dirección en memoria del primer elemento del vector a9 pero en su versión decimal y luego el valor al que apunta esa dirección de memoria, el cual sería el primer elemento, es decir, 0.
2. Segundo tenemos $PR3(d, p9, * p9, ** p9)$, esto imprime primero la dirección en memoria del primer elemento del vector de punteros p9, luego imprime el valor del primer elemento en el vector de punteros p9, el cuál sería la dirección en memoria del vector a9, luego imprime el valor contenido en la primer dirección de memoria del vector p9, que sería el primer elemento del vector a9, es decir, 0.
3. Tercero tenemos $PR3(d, pp9, * pp9, ** pp9)$, esto imprime primero el valor contenido en el puntero de puntero pp9, el cual sería la dirección en memoria de p9, luego imprime el valor del primer elemento de p9, el cual sería la dirección en memoria de a9, luego imprime el valor de la dirección en memoria del primer elemento contenido en p9, el cual sería la dirección en memoria de a9, siendo este el primer elemento de ese vector y dando como resultado 0.
4. Cuarto tenemos $pp9\ ++;\ PR3(d, pp9 - p9, * pp9 - a9, ** pp9)$, primero con pp9++ se cambia a la siguiente dirección de memoria que apunta el vector de punteros p9, es decir, la

dirección en memoria del segundo elemento en el vector de punteros p9. Entonces, primero imprime 1, ya que es la diferencia entre pp9 y p9, luego imprime 1 ya que es la diferencia entre *pp9, que sería la dirección en memoria del segundo elemento del vector p9, es decir a9+1, y la dirección en memoria de a9, es decir que se tendría algo como a9+1-a9, y por último imprime el valor contenido dentro del segundo elemento del vector p9, el cual es a9+1, siendo su valor 1.

5. Quinto tenemos * $pp9$ ++; $PR3(d, pp9 - p9, * pp9 - a9, ** pp9)$, primero con *pp9++ se modifica la dirección de memoria a la que apunta pp9, entonces ahora apunta al tercer elemento del vector p9. Ahora se imprime la diferencia que hay entre la dirección en memoria que contiene pp9 y la dirección en memoria de p9, si p9 apunta a la dirección de memoria del primer elemento en su vector y pp9 a la tercera, entonces la diferencia es 2, por lo tanto se imprime un 2. Luego, se imprime la diferencia entre el tercer elemento de p9, el cual sería la dirección de memoria a9+2, y a9, que da como resultado 2, por lo tanto se imprime un 2. Por último, se imprime el valor contenido en la dirección de memoria a9+2, el cual sería un 2, por lo tanto se imprime un 2.
6. Sexto tenemos *++ $pp9$; $PR3(d, pp9 - p9, * pp9 - a9, ** pp9)$, con *++pp9 hacemos que la variable pp9 apunte al cuarto elemento del vector de punteros p9. Ahora, se imprime la diferencia entre la dirección de memoria contenido en pp9, la cual sería el cuarto elemento del vector p9, y la dirección de memoria del primer elemento del vector p9, se imprime 3. Luego, la diferencia entre *pp9 que sería el cuarto elemento en p9, es decir a9+3, y a9, se imprime 3. Por último, se imprime el valor que contiene la dirección en memoria del cuarto elemento en el vector p9, es decir el valor en a9+3, el cual sería 3.
7. Séptimo tenemos ++* $pp9$; $PR3(d, pp9 - p9, * pp9 - a9, ** pp9)$, con ++*pp9 apuntamos al quinto elemento del vector p9, es decir, a a9+4, pero no modificamos el valor de pp9. Ahora, se imprime la diferencia entre pp9 y p9, la cual es 3. Luego, se imprime la diferencia entre lo que apunta *pp9 y a9, la cual es 4 y luego se imprime el valor en a9+4, el cual es 4.
8. Se le asigna p9 a pp9.
9. Octavo tenemos ** $pp9$ ++; $PR3(d, pp9 - p9, * pp9 - a9, ** pp9)$, con **pp9++ hacemos que pp9 apunte a la siguiente dirección en memoria, es decir, que pp9 ahora apunta a la dirección de memoria del segundo elemento de p9. Ahora, se imprime la diferencia entre pp9 y p9, la cual es 1. Luego, se imprime la diferencia entre el valor contenido en la dirección de memoria del segundo elemento de p9, es decir, a9+1 y a9, el cual es 1. Por último, se imprime el valor de a9+1, el cual es 1.
10. Noveno tenemos *++* $pp9$; $PR3(d, pp9 - p9, * pp9 - a9, ** pp9)$, con *++*pp9 hacemos que *pp9 apunte a la tercera posición del vector p9, pero el valor de pp9 no cambia, es decir, que sigue apuntado a la dirección del segundo elemento del vector p9. Ahora, se imprime la diferencia

entre pp9 y p9, el cual es 1. Luego, se imprime la diferencia entre *pp9 y a9, el cual es 2, ya que *pp9 sería a9+2. Por último, se imprime el valor en a9+2 el cual es un 2.

11. Último tenemos $++{**}\ pp9;\ PR3(d, pp9 - p9, *\ pp9 - a9, **\ pp9)$, con ++**pp9 modifica el valor del tercer elemento en a9 dando como resultado 3. Ahora, se imprime la diferencia entre pp9 y p9, el cual es 1. Luego, la diferencia entre *pp9 y a9, el cual es 2. Por último, imprime el valor en la dirección de memoria **pp9 el cual es 3.

Pregunta 10

```
173
174  int a10[3][3] = {
175      { 1, 2, 3 },
176      { 4, 5, 6 },
177      { 7, 8, 9 }
178  };
179
180  int *pa[3] = { a10[0], a10[1], a10[2] };
181  int *p10 = a10[0];
182
183  void pregunta10() { // multiples dimensiones
184      int i;
185
186      for( i=0; i<3; i++ )
187          PR3(d, a10[i][2-i], *a10[i], *(*(a10+i)+i) );
188      NL;
189
190      for ( i=0; i<3; i++ )
191          PR2(d, *pa[i], p10[i]);
192  }
```

1. Primero se realiza un ciclo con 3 iteraciones las cuales ejecutan $PR3(d,\ a10[i][2 - i],\ *\ a10[i],\ *\ (*\ (a10 + i) + i)\ )$
    a. Para la primera iteración, $i = 0$, entonces primero se imprime $a10[0][2]$, el cual es un 3. Luego, se imprime el valor de la dirección en memoria del primer vector dentro de a10, el cual es 1. Por último, como $i = 0$ se accede al valor del puntero que apunta al primer vector de a10, el cual es 1.
    b. Para la segunda iteración, $i = 1$, entonces primero se imprime $a10[1][1]$, el cual es 5. Luego, se imprime el valor de la dirección de memoria del segundo vector dentro de a10, es decir, 4. Por último, *(*(a10+1)+1) con a10+1 se mueve al segundo vector de a10,

luego se le suma 1 al puntero, entonces ahora se apunta al segundo elemento dentro del segundo vector de a10 y por último accedemos a ese elemento, es decir, 5.

c. Para la última iteración, $i = 2$, entonces primero se imprime $a10[2][0]$, el cual es un 7. Luego, se accede al valor del puntero en *a10[2], es decir, el primer elemento del tercer vector en a10, se imprime un 7. Por último, *(*(a10+2)+2) esto mueve el puntero hacia el tercer vector dentro de a10, luego mueve el puntero a la tercera posición del vector y accede al valor, entonces imprime un 9.

2. Luego, se realiza un ciclo con 3 iteraciones que ejecuta $PR2(d,\ *\ pa[i],\ p10[i])$:
   a. Para la primera iteración, $i = 0$, con *pa[0] accedemos a la primera posición del vector pa, el cual es a10[0] y luego obtenemos el valor de a10[0] el cual es 1. Luego, con p10[0] accedemos a la primera dirección del vector a10[0] el cual apunta al primer elemento, es decir, 1.
   b. Para la segunda iteración, $i = 1$, con *pa[1] accedemos a la segunda posición del vector pa, el cual es a10[1] y luego obtenemos el valor de a10[1], el cual es 4. Luego, con p10[1] obtenemos el segundo elemento del primer vector de a10, es decir, 2.
   c. Para la tercera iteración, $i = 2$, con *pa[2] accedemos a la tercera posición del vector pa, el cual es a10[2], y luego obtenemos el primer elemento del tercer vector en a10, es decir, 7. Por último, con p10[2], accedemos al tercer elemento del primer vector en a10, es decir, 3.

Pregunta 11

```
194  char *c[] = {
195      "ENTER",
196      "NEW",
197      "POINT",
198      "FIRST"
199  };
200  char **cp[] = { c+3, c+2, c+1, c };
201  char ***cpp = cp;
202
203  void pregunta11() { // sopa de punteros
204      printf("%s", **++cpp );
205      printf("%s ", *--*++cpp+3 );
206      printf("%s", *cpp[-2]+3 );
207      printf("%s\n", cpp[-1][-1]+1 );
208  }
209
```

1. Primero con **++cpp le suma 1 a cpp, entonces cpp pasa a apuntar a la segunda posición de cp, luego obtiene el valor de cp en esa dirección el cual es c+2, eso devuelve POINT y se imprime en pantalla.
2. Segundo con *--*++cpp+3, primero mueve el puntero al tercer elemento de cp, luego le resta 1 a la dirección de memoria lo que hace que apunte al primer string de c, luego le suma 3 a esa posición de memoria, lo que hace que apunte a ‘E’ dentro del string ‘ENTER’, se imprime ‘ER’.
3. Tercero con *cpp[-2]+3, primero nos posicionamos al primer elemento de cp y accedemos al último elemento de c que es esa misma dirección de memoria, ahora le sumamos 3, entonces ahora se apunta a ‘S’, se imprime ‘ST’.
4. Cuarto con cpp[-1][-1]+1, primero nos posicionamos en el segundo elemento de cp, luego se apunta al segundo elemento de c y nos movemos una posición dentro de la dirección de memoria, es decir se apunta a ‘E’ y se imprime ‘EW’.