ILI-134 Estructuras de Datos, 2013-2

Guía de Ejercicios 0: Lenguaje de Programación C

Profesor: Diego Arroyuelo
Universidad Técnica Federico Santa María
Campus Santiago – San Joaquín

1 Guía de Ejercicios 0: Lenguaje de Programación C++

Todos los ejercicios de esta guía son obligatorios.

Conceptos Básicos de C++

1. Dadas las siguientes declaraciones:

```
int i, j, k;
float f, g;
char c;
```

Evaluar las siguientes expresiones en orden, mostrando paso a paso el valor que toma cada una de las variables:

```
i = 1;
j = i++;
k = ++i;
i = (j == 1) ? --k : k++;
f = 2/4;
g = 2.0/4;
c = 'b' - 1;
i = (c >= 'a') && (c <= 'z') ? 1 : 0;
j = (i < j && (f = k-1) != 1);
```

2. ¿Cuál es la salida del siguiente programa?

```
#include <iostream>
using namespace std;
int main ()
{
    char pax[] = "Estructuras de Datos";
    cout << pax << endl;
    cout << pax[11] << endl;
```

```
    cout << &pax[11] << endl;
    return 0;
}
```

3. De acuerdo a las reglas de precedencia del lenguaje C++, indicar el resultado de la evaluación de las siguientes expresiones

   (a) `6 + 2 * 3 - 4 /2`.
   (b) `7 - 6 / 3 + 2 * 3 / 2 - 4 / 2`.
   (c) `7 * 10 - 5 % 3 * 4 + 9`.
   (d) `15 * 14 - 3 * 7`.
   (e) `-4 * 5 * 2`.
   (f) `(24 + 2 * 6) / 4`.
   (g) `3 + 4 *(8 * (4 - (9 + 3)/6))`.
   (h) `4 * 3 * 5 + 8 * 4 * 2 - 5`.
   (i) `4 - 40 / 15`.
   (j) `6 % 2 * 15`.

4. Escribir las siguientes expresiones aritméticas usando lenguaje C++. Use las reglas de asociatividad y precedencia definidas en el lenguaje para emplear la mínima cantidad de paréntesis posible en cada expresión. Tenga en cuenta que una expresión del tipo $(x+y)^2$ puede escribirse como `pow(x+y,2)`.

   (a) $\frac{x}{y} + 1$.
   (b) $\frac{x+y}{x-y}$.
   (c) $\frac{x+\frac{y}{z}}{x-\frac{y}{z}}$.
   (d) $\frac{b}{c+d}$.
   (e) $(a+b)\frac{c}{d}$.
   (f) $\frac{xy}{1-4x}$.

5. Escribir un programa que lea dos enteros en las variables `x` e `y`, y a continuación obtenga los valores de : `x/y` y `x%y`. Ejecute el programa varias veces con diferentes pares de enteros como entrada.

6. ¿Cuáles son los resultados visualizados en la salida estándar por el siguiente programa, si los datos proporcionados son 5 y 8?

```
#include <iostream>
using namespace std;
const int M = 6;
int main ()
{
```

```
    int a, b, c;
    cout << "Introduzca el valor de a y de b:";
    cin >> a;
    cin >> b;
    c = 2 * a - b;
    c -= M;
    b = a + c - M;
    a = b * M;
    cout << "\n a = " << a << endl;
    b = -1;
    cout << b << c << endl;
    return 0;
}
```

7. Escriba un programa para calcular la longitud de la circunferencia y el área del círculo para un radio introducido por el teclado.

8. Un sistema de ecuaciones lineales

$$\begin{aligned} ax + by &= c \\ dx + ey &= f. \end{aligned}$$

se puede resolver como:

$$x = \frac{ce - bf}{ae - bd} \qquad y = \frac{af - cd}{ae - bd}.$$

Diseñar un programa que lea dos conjuntos de coeficientes (a, b y c ; d, e y f) y visualice los valores de x e y.

9. Escribir un programa para convertir una medida dada en pies a sus equivalentes en: a) yardas; b) pulgadas; c) centímetros, y d) metros (1 pie = 12 pulgadas, 1 yarda = 3 pies, 1 pulgada = 2,54 cm, 1 m = 100 cm). Leer el número de pies e imprimir el número de yardas, pies, pulgadas, centímetros y metros.

10. Teniendo como datos de entrada el radio y la altura de un cilindro queremos calcular: el área lateral y el volumen del cilindro.

11. Escribir un programa que lea el radio de un círculo y a continuación visualice: área del círculo, diámetro del círculo y longitud de la circunferencia del círculo.

12. Hacer un programa que acepte como entrada tres carácteres, que deben ser algunos de los dígitos del `0` al `9`, y los convierta en el correspondiente número entero de tres dígitos y lo imprima. Por ejemplo: Entrada: '4', '9', '7' → Salida: 497.

13. Indique la salida producida por el siguiente programa.

```
#include <iostream>
using namespace std;
```

```
int main()
{
    int m = 45, n = 75;
    cout << "m = " << m << " n = " << n << endl;

    cout << "m = " << ++m << " n = " << n-- << endl;

    cout << "m = " << m++ << " n = " << ++n << endl;
    return 0;
}
```

14. Indique la salida producida por el siguiente programa.

```
#include <iostream>
using namespace std;
int main()
{
    int m = 99, n;
    n = ++m;
    cout << "m = " << m << " n = " << n << endl;
    n = m++;
    cout << "m = " << m << " n = " << n << endl;
    cout << "m = " << m++ << endl;
    cout << "m = " << ++m << endl;
    return 0;
}
```

Sentencias Condicionales: `if-else`, `switch`

1. ¿Qué puede siempre afirmarse que es verdad si en una sentencia `if` en C, cuya condición es:

   `(v >= 0 && v <= s)`

   se ejecuta la rama del verdadero?

   (a) `(v < 0 || v > s)`
   (b) `(v < 0 && v > s)`
   (c) `(v >= 0 || v <= s)`
   (d) `(v >= 0 && v <= s)`
   (e) No se ejecuta nunca la rama del verdadero con esa condición.
   (f) Ninguna de las anteriores.

2. ¿Qué valor se asigna a la variable `consumo` en la sentencia if siguiente si `velocidad` es 120?

```
if (velocidad > 80)
   consumo = 10.00;
else if (velocidad > 100)
   consumo = 12.00;
else if (velocidad > 120)
   consumo = 15.00;
```

3. Explique las diferencias entre las sentencias de la columna de la izquierda y de la columna de la derecha. Para cada una de ellas deducir el valor final de `x` si el valor inicial de `x` es `0`.

```
if (x >= 0)                    if (x >= 0)
   x++;                           x++;
else if (x >= 1)               if (x >= 1)
   x += 2;                        x += 2;
```

4. Escribir un programa que lea dos números y visualice el mayor, utilizando el operador ternario `?:`.

5. Determinar si el carácter asociado a un código introducido por teclado corresponde a un carácter alfabético, dígito, signo de puntuación, carácter especial o carácter no imprimible

6. Escribir un programa que lea tres enteros y emita un mensaje que indique si fueron o no ingresados en orden numérico.

7. Hacer un programa que acepte como entrada tres carácteres, que deben ser algunos de los dígitos del `0` al `9`, y opcionalmente un carácter que represente un signo (`+` o `-`) y los convierta en el correspondiente número de tres dígitos y lo imprima. Por ejemplo:

   Entrada:'-', '4', '9', '7' $\rightarrow$ Salida: -497

   Entrada: '+', '4', '9', '7' $\rightarrow$ Salida: 497

8. Escribir un programa que permita introducir el número de un mes (1 a 12) y visualice el número de días que tiene ese mes.

9. Escribir un programa que lea la hora de un día en notación de 24 horas y escriba esa hora notación de 12 horas. Por ejemplo, si la entrada es 13:45, la salida será 1:45 PM. El programa pedirá al usuario que introduzca exactamente cinco carácteres. Así, por ejemplo, las nueve en punto se introduce como 09:00.

10. Se quiere calcular la edad de un individuo. Para ello se va a tener como entrada dos fechas en el formato día (1 a 31), mes (1 a 12) y año (entero de cuatro dígitos), correspondientes a la fecha de nacimiento y la fecha actual, respectivamente. Escribir un programa que calcule y visualice la edad del individuo. Si la persona tiene menos de un año de edad, la edad se debe dar en meses y días; en caso contrario, la edad se calculará en años.

11. Escribir un programa que detemine si un año es bisiesto. Un año es bisiesto si es múltiplo de 4 (por ejemplo, 1984). Sin embargo, los años múltiplos de 100 sólo son bisiestos cuando a la vez son múltiplos de 400 (por ejemplo, 1800 no es bisiesto, mientras que 2000 sí lo es).

12. Escribir un programa que calcule el número de días de un mes, dados los valores numéricos del mes y el año. Tener en cuenta el caso especial de febrero en los años bisiestos.

13. Determinar el menor número de billetes y monedas de curso legal equivalentes a cierta cantidad de dinero en pesos chilenos.

14. Escribir y ejecutar un programa que simule una calculadora simple. Lee dos enteros y un carácter. Si el carácter es `+`, se imprime la suma; si es `-`, se imprime la diferencia; si es `*`, se imprime el producto; si es `/`, se imprime el cociente; y si es `%` se imprime el resto. Nota: utilizar la sentencia switch.

Sentencias de Repetición: `for`, `while`, `do-while`

1. Qué puede siempre afirmarse que es verdad una vez terminada una sentencia de iteración cuya condición es:

   ```
   (i <= 'T' || a)
   ```

   suponiendo que `i` es de tipo `char` y `a` es un entero.

   (a) La condición es verdadera.

   (b) `i <= 'T' && a`

   (c) `i > 'T' && !a`

   (d) `i > 'T' || !a`

   (e) Nada, porque no puede haber una expresión condicional así.

   (f) Ninguna de las anteriores.

2. ¿Cuál es la salida del siguiente segmento de programa?

   ```
   for (cuenta = 1; cuenta < 5; cuenta++)
      cout << 2*cuenta << " ";
   ```

3. ¿Cuál es la salida de los siguientes bucles?

   (a)
   ```
   for (n = 10; n > 0; n = n-2) {
         cout << "Hola  ";
         cout << n << endl;
    }
   ```

   (b)
   ```
   double n = 2.0;
   for (; n > 0; n = n-0.5)
      cout << n << endl;
   ```

4. Considere el siguiente segmento de programa:

   ```
   int i = 1;
   while (i <= n) {
      if ((i%n) == 0) {
   ```

```
      ++i;
   }
}
cout << i << endl;
```

(a) ¿Cuál es la salida si `n` es 0?

(b) ¿Cuál es la salida si `n` es 1?

(c) ¿Cuál es la salida si `n` es 3?

5. ¿Cuál es la salida producida por los siguientes segmentos de programa?

(a)
```
int n, m;
for (n = 1; n <= 10; n++)
   for (m = 10; m >= 1; m--)
      cout << n << " veces " << m << "=" << n*m << endl;
```

(b)
```
for (i = 0; i < 10; i++)
   cout << " 2 * " << i << " = " << 2*i << endl;
```

(c)
```
for (i = 0; i <= 5; i++)
   cout << 2 * i + 1 << " ";
   cout << endl;
```

(d)
```
for (i = 1; i < 4; i++) {
   cout << i << " ";
   for (j = i; j >= 1; j--)
      cout << j << endl;
}
```

(e)
```
for (i = 1; i <= 5; i++) {
   cout << i << " ";
   for (j = i; j >= 1; j-=2)
      cout << j << endl;
}
```

(f)
```
for (i = 3; i > 0; i--)
   for (j = 1; j <= i; j++)
      for (k = i; k >= j; k--)
         cout << i << " " << j << " " << k << endl;
```

(g)
```
for (i = 3; i > 0; i--)
   for (j = 1; j <= i; j++)
      for (k = i; k >= j; k--)
         cout << i << " " << j << " " << k << endl;
```

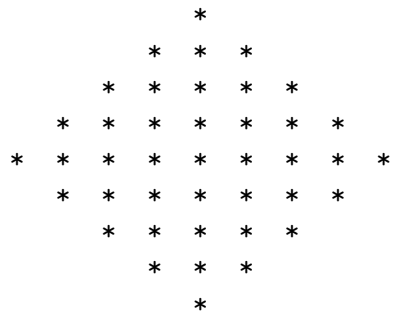
(h)
```
for (i = 1; i <= 3; i++)
   for (j = 1; j <= 3; j++) {
      for (k = i; k <= j; k++)
         cout << i << " " << j << " " << k << endl;
      cout << endl;
   }
```

(i)
```
i = 0;
while (i*i < 10) {
   j = i;
   while (j*j < 100) {
      cout << i+j << endl;
      j *= 2;
   }
   i++;
}
cout << "\n*****\n";
```

6. Escribir un programa que pida ingresar dos enteros, `a` y `b`, e imprima por pantalla un rectángulo de base `a` y altura `b` usando asteriscos (‘`*`’).

7. Modificar el programa anterior, de modo que además se lea una palabra de cinco letras y se imprima en el centro del rectángulo.

8. Escribir un programa que pida ingresar un entero `a` e imprima por pantalla un rombo de lado `a`. Por ejemplo, si `a` es 5, debería mostrar:



9. En una empresa, los salarios de los empleados se van a aumentar de acuerdo a la siguiente tabla:

Contrato	% de aumento
0–9.000 dolares	20%
9.001–15.000 dolares	10%
15.001-20.000 dolares	5%
más de 20.000 dolares	0%

Escribir un programa que solicite el salario actual del empleado y calcule y visualice el nuevo salario. El programa debe permitir ingresar los salarios de varios empleados.

10. Escribir un programa que permita simular una calculadora simple (tal como se hizo en ejercicios anteriores), pero que ahora permita realizar varias operaciones simultáneas.

11. Un método sencillo para aproximar la constante π (3.141592...) es

$$\pi = 4 \cdot \frac{2}{3} \cdot \frac{4}{3} \cdot \frac{4}{5} \cdot \frac{6}{5} \cdot \frac{6}{7} \cdot \frac{8}{7} \cdot \frac{8}{9} \cdots \frac{2n}{2n-1} \cdot \frac{2n}{2n+1} \cdots$$

Escriba un programa que permita calcular π para un valor de n ingresado por el usuario.

12. Escribir un programa que determine todos los años que serán bisiestos en el siglo XXI.

13. El valor de e^x se puede aproximar con la serie

$$e^x = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \cdots + \frac{x^n}{n!} + \cdots$$

Escriba un programa que permita ingresar un valor de x y de n, y muestre la aproximación al valor e^x hasta el n-ésimo término de la serie.

14. El algoritmo de Euclides para encontrar el máximo común divisor (mcd) de dos números trabaja de la siguiente manera. Dados los enteros a y b ($a > b$), si $a/b = q_1$ y el resto es $r_1 \neq 0$, se tiene que $\text{mcd}(a, b) = \text{mcd}(b, r_1)$. Luego, si $b/r_1 = q_2$ y el resto es $r_2 \neq 0$, se tiene que $\text{mcd}(b, r_1) = \text{mcd}(r_1, r_2)$. El proceso continua hasta que se obtiene resto 0. El resto del paso anterior es entonces el mcd de a y b. Escribir un programa que calcule el mcd de dos números dados.

15. Escribir un programa que calcule la suma de la serie

$$1 + \frac{1}{2} + \frac{1}{3} + \cdots + \frac{1}{n},$$

en donde n es el menor número natural tal que $\frac{1}{n} < \epsilon$, para una constante $\epsilon \in \mathbb{R}$ ingresada por teclado, tal que $\epsilon > 0$.

16. Entontrar el número natural n más pequeño tal que la suma de los n primeros números naturales exceda de una cantidad introducida por el teclado.

17. Escribir un programa que calcule y visualice el más grande, el más pequeño y la media de n números. El valor de n se solicitará al principio del programa y los números serán introducidos por el usuario. Sólo puede emplear variables simples en su solución (es decir, no se pueden usar arreglos).

18. Calcular todos los números de tres cifras tales que la suma de los cubos de las cifras es igual al valor del número.

19. Un carácter es un espacio en blanco si es un blanco (`' '`), una tabulación (`'\t'`), un carácter de nueva línea (`'\n'`), o un avance de página (`'\f'`). Escribir un programa que permita ingresar carácteres desde el teclado (puede ser una cantidad fija de carácteres o usando un carácter de finalización para terminar con el ingreso de datos). El programa debe contar la cantidad de espacios en blancos ingresados.

20. Escribir un programa que imprima en orden inverso los dígitos de un número entero positivo dado.

Funciones

1. Para el siguiente programa:

```
#include <iostream>

int cube(int y) {
   return y * y * y;
}

int main() {
   int x;
   for (x = 1; x < = 10; x++)
      cout << cube(x) << endl;
   return 0;
}
```

establecer el ámbito de cada uno de los siguientes identificadores:

(a) La variable `x` en la función `main`.

(b) El parámetro `y` en la función `cube`.

(c) La función `cube`.

(d) La función `main`.

2. Escribir una función que tenga un argumento de tipo `int` y retorne `'P'` si dicho número es positivo, o `'N'` en otro caso.

3. Escriba una función que reciba como argumento un entero n, y retorne la suma de los primeros n naturales.

4. Escriba la función `maximo`, que tome como entrada dos números enteros, y retorne el mayor de esos dos números.

5. Escriba una función que calcule el mcd de dos números usando el algoritmo de Euclides.

6. Escribir una función que calcule el coeficiente binomial a partir de dos números enteros.

7. Escribir una función que calcule el valor de la progresión:

$$1 + x + x^2 + x^3 + \cdots + x^n,$$

para dos argumentos `x` (`float`) y `n` (`int`).

8. Escribir una función que tenga como argumentos dos números naturales n y m, y que devuelva como resultado el producto $m \times n$. La multiplicación debe realizarse empleando la suma y no el producto.

9. Escribir una función que tenga como argumentos dos números naturales n y m, y que devuelva como resultado la división entera m/n. La división debe realizarse empleando la resta y no la división.

10. Definir una función llamada `hipotenusa` que calcule la longitud de la hipotenusa de un triángulo rectángulo dados los otros dos lados del triángulo. Usar esta función en un programa que solicite el ingreso de datos (los lados de los triángulos) y luego calcule la longitud de la hipotenusa de cada triángulo ingresado.

11. Suponga la siguiente definición:

    ```
    #define true 1
    #define false 0
    ```

    Escribir una función `multiplo` que determine para un par de enteros si el segundo entero es múltiplo del primero. La función deberá tomar dos argumentos enteros y retornar `true` si el segundo es múltiplo del primero, y `false` en otro caso. Usar la función `multiplo` en un programa que permita ingresar una serie de pares de enteros e imprima el mensaje adecuado por cada par.

12. (a) Implementar la función `celsius` que retorna el equivalente en grados Celsius de una temperatura en grados Fahrenheit dada.
    (b) Implementar la función `fahrenheit` que retorna el equivalente en grados Fahrenheit de una temperatura en grados Celsius dada.
    (c) Usar las funciones para escribir un programa que imprima una tabla mostrando las equivalencias Fahrenheit de todas las temperaturas desde 0 a 199 grados Celsius y otra tabla mostrando las equivalencias Celsius de todas las temperaturas desde 32 a 231 grados Fahrenheit.

13. Hacer un programa que simule los lanzamientos de una moneda. Para cada lanzamiento de la moneda el programa deberá imprimir "cara" o "sello". Hacer que el programa lance la moneda un número determinado de veces y cuente el número de veces que sale cada lado de la moneda y lo imprima. Para simular los lanzamientos, el programa deberá llamar a una función `flip` que no toma argumentos y retorna 0 para sello y 1 para cara. Para definir la función `flip`, usar el operador de módulo (`%`), y las siguientes funciones predefinidas:

    - `rand()` definida en `<cstdlib>` genera un número entero pseudo random entre 0 y `RAND_MAX` (constante simbólica definida en `<cstdlib>`).
    - `srand(unsigned int)` definida en `<cstdlib>` establece la semilla para la generación de números pseudo random. Distintas semillas generan distintas secuencias de números.
    - Usar la función `time` (definida en `<ctime>`) como semilla de la función `srand` para tener semillas diferentes en cada ejecución:

      ```
      srand(time(NULL));  /* define la semilla */

      i = rand();         /* genera numero random */
      ```

Punteros

1. Para cada uno de los siguientes puntos, escribir una sentencia usando lenguaje C++ que lleve a cabo la tarea indicada. Asumir que las variables `numero1` y `numero2` de tipo `float` se encuentran definidas y que `numero1` ha sido inicializada con el valor 7.3.

(a) Definir la variable `fPtr` como un puntero a un valor de tipo float.

(b) Asignar la dirección de la variable `numero1` a la variable `fPtr`.

(c) Imprimir el valor del dato apuntado por `fPtr`.

(d) Asignar el valor del dato apuntado por `fPtr` a la variable `numero2`.

(e) Imprimir el valor de `numero2`.

(f) Imprimir la dirección de la variable `numero1`.

(g) Imprimir la dirección almacenada en `fPtr`. ¿El valor impreso será el valor de la dirección de `numero1`?

2. Dadas las siguientes declaraciones:

```
int i;
int *p, *q;
```

¿Cuáles de las siguientes asignaciones causa un error de compilación?

(a) `p = &i;`

(b) `p = *&i;`

(c) `p = &*i;`

(d) `i = *&*p;`

(e) `i = *&p;`

(f) `i = &*p;`

(g) `p = &*&i;`

(h) `q = *&*p;`

(i) `q = **&p;`

(j) `q = *&p;`

(k) `q = &*p;`

3. Identificar los errores en los siguientes segmentos de programa. Asuma que `s2` es un string (como arreglo de carácteres) al que ya se le ha asignado un valor.

(a)
```
char *f(char *s) {
   char ch = 'A';
   return &ch;
}
```

(b)
```
char *s1;
strcpy(s1, s2);
```

(c)
```
char *s1;
s1 = new char[strlen(s2)];
strcpy(s1, s2);
```

4. Dadas las declaraciones e inicializaciones:

```
int intArray[] = {1, 2, 3}, *p = intArray;
```

mostrar el contenido de ambas variables después de ejecutar las siguientes sentencias.

(a) `*p++;`

(b) `(*p)++;`

(c) `*p++; (*p)++;`

5. Definir la función `swap`, que permita intercambiar los valores de dos variables enteras pasadas como parámetro.

Arreglos

1. Indicar si son verdaderas o falsas las siguientes afirmaciones:

   (a) El tipo base de los arreglos en C++ es heterogéneo, es decir, se pueden almacenar valores de distintos tipos en un mismo arreglo.

   (b) El índice de un arreglo en C++ puede ser de tipo `double`.

   (c) Si hay un número menor de inicializadores en una lista de inicializadores que el número de elementos en el arreglo, C++ automáticamente inicializa los restantes elementos con el último valor de la lista de inicializadores.

   (d) Es un error si una lista de inicializadores contiene más inicializadores que elementos tiene el arreglo.

   (e) Un elemento de arreglo que es pasado a una función `f` como parámetro de la forma `f(a[i])` y es modificado en la función `f` se modificará también en la función que hace la invocación a `f`.

   (f) En C++, cuando en una función uno de sus parámetros es un arreglo unidimensional, en la definición de la función se debe colocar su tipo base, nombre y tamaño del arreglo.

   (g) En C++, cuando en una función uno de sus parámetros es un arreglo multidimensional, en la definición de la función se debe colocar su tipo base, nombre y tamaño de todas las dimensiones del arreglo, excepto la primera.

2. Realizar las siguientes acciones, siempre que sea posible hacerlo. En caso que no sea posible, indicar por qué.

   (a) Definir un arreglo de diez elementos de tipo `float` llamado `numeros` e inicializar los elementos con los valores `0.0`, `1.1`, `2.2`, ..., `9.9`. Asumir que la constante simbólica `SIZE` ha sido definida como `10`.

   (b) Definir un puntero, `nPtr`, que sea capaz de apuntar a una variable de tipo `float`.

   (c) Imprimir los elementos del arreglo `numeros` usando un índice entero `i` (que debe ser definido) para recorrer el arreglo. Usar una sentencia `for` para el recorrido.

   (d) Dar dos sentencias alternativas para asignar la dirección base del arreglo `numeros` en el puntero `nPtr`.

(e) Imprimir los elementos del arreglo `numeros` usando el puntero `nPtr` para recorrer el arreglo.

(f) Imprimir los elementos del arreglo `numeros` usando el nombre del arreglo como puntero para recorrerlo.

(g) Imprimir los elementos del arreglo `numeros` usando un índice entero `i` (que debe ser definido) para recorrer el arreglo, y el puntero `nPtr` como la base del arreglo.

(h) Referenciar al elemento 4 del arreglo `numeros` usando:

- Un índice entero.
- Usando el nombre del arreglo como puntero.
- Usando un índice entero, con el puntero `nPtr` como base.
- Usando el puntero `nPtr`.

(i) Asumir que `nPtr` apunta al comienzo del arreglo `numeros`.

(j) ¿Qué dirección es referenciada por `nPtr + 8`? ¿Qué valor se encuentra almacenado en esa dirección?

(k) Asumir que `nPtr` apunta a `numeros[5]`. ¿Qué dirección es referenciada por `nPtr -= 4`? ¿Qué valor se encuentra almacenado en esa dirección?

3. Encontrar el error o situaciones erroneas en cada uno de los siguientes segmentos de programa. Asumir:

```
int *zPtr;
int *aPtr = NULL;
void *sPtr = NULL;
int number, i;
int z[5] = {1, 2, 3, 4, 5 };
sPtr = z;
```

(a)
```
++zPtr;
```

(b)
```
/* usa el puntero zPtr para obtener el primer valor de un arreglo;
asumir zPtr inicializado con la direccion base del arreglo z */
number = zPtr;
```

(c)
```
/* asigna el elemento 2 del arreglo (el valor 3) a number; asumir
zPtr inicializado con la direccion base del arreglo z */
number = *zPtr[2];
```

(d)
```
/* imprime el arreglo z completo; asumir zPtr inicializado con la
direccion base del arreglo z */
for (i = 0 ; i <= 5; i++)
   cout << zPtr[i] << " ";
```

(e)
```
/* asigna el valor apuntado por sPtr a number */
number = *sPtr;
```

(f)
```
++z;
```

4. Suponiendo las siguientes declaraciones:

```
int i, j, k;
int Primero[21], Segundo[21];
int Tercero[6][12];
```

determinar la salida producida por cada uno de los siguientes segmentos de programa.

(a)
```
for (i = 1; i <= 6; i++)
   cin >> Primero[i];
for(i = 3; i > 0; i--)
   cout << Primero[2*i] << " ";
   ..................
```
Suponga que la entrada de datos es la siguiente: 3, 7, 4, - 1, 0, 6.

(b)
```
cin >> k;
for(i = 3; i<=k;)
   cin >> Segundo[i++];
j = 4;
cout << Segundo[k] << " " << Segundo[j+1] << endl;
..................
```
Suponga que la entrada de datos es la siguiente: 6, 3, 0, 1, 9.

(c)
```
for(i = 0; i < 3; i++)
   for(j = 0; j < 12; j++)
      Tercero[i][j] = i+j+1;
for(i = 0; i < 3; i++) {
   j = 2;
   while (j < 12) {
      cout << i << " " << j << " " << Tercero[i][j] << endl;
      j +=3;
   }
}
```

5. Escribir una función que intercambie los valores almacenados en la fila i-ésima con los de la fila j-ésima de un arreglo de dos dimensiones, para valores i y j dados.

6. Escribir un programa que rellene una tabla con los 80 primeros números primos, y luego los visualice. Use las funciones programadas en ejercicios anteriores que considere necesarias para resolver el problema.

7. Escribir una función que reciba un arreglo de enteros como parámetro, y devuelva el menor elemento del arreglo.

8. Escribir una función que reciba un arreglo de enteros y un número entero `x` como parámetro, y devuelva -1 en caso que no exista ninguna ocurrencia del valor de `x` en el arreglo. En otro caso, debe devolver la primera posición del arreglo que contenga el valor de `x`.

9. Hacer un programa que permita insertar un carácter en un arreglo de carácteres en una posición dada. Se inserta en el lugar del arreglo que el usuario del programa solicite y debe hacerse lugar al nuevo carácter, desplazando todos los carácteres una posición en el arreglo,

perdiéndose el último carácter del arreglo. El programa deberá definir y usar una función para ingresar los datos, otra para hacer la inserción y otra para mostrar el arreglo antes y después de la inserción.

10. Hacer un programa que permita borrar un carácter en un arreglo de carácteres. Se borra en el lugar del arreglo que el usuario del programa solicite y deben desplazarse los carácteres una posición desde la posición dada hasta la posición final del arreglo. El programa deberá definir y usar (o reusar) una función para ingresar los datos, otra para hacer la supresión y otra para mostrar el arreglo antes y después de la misma.

11. Escribir una función que reciba como parámetro un arreglo de números de tipo `float`. Dicha función debe ordenar el arreglo en forma ascendente.

Structs

1. Indicar si son verdaderas o falsas las siguientes afirmaciones.

   (a) Un struct es una colección de datos relacionados lógicamente, bajo un mismo nombre.

   (b) Los struct sólo pueden contener campos que sean todos de un mismo tipo.

   (c) Los struct sólo pueden contener campos que sean todos de distintos tipos.

   (d) Dos struct no pueden compararse usando los operadores `==` y `!=`.

   (e) Una variable de tipo struct no se puede asignar a otra variable del mismo tipo.

   (f) En una declaración de una variable el nombre del struct es opcional.

   (g) Los campos o miembros de distintos struct deben tener nombres únicos entre sí.

   (h) La palabra clave `typedef` es usada para definir nuevos tipos de datos.

   (i) Los struct en C++ son siempre pasados por referencia a las funciones.

2. Escriba código C++ para:

   (a) Definir un struct llamado `parte`, que sirve para guardar datos de piezas (partes) que contenga un campo entero `numero` para el número de parte y un campo string `nombre` para el nombre de parte, cuya longitud puede ser a lo sumo de 25 carácteres (incluido el carácter nulo).

   (b) Definir `Parte` como un sinónimo para el tipo `struct parte`.

   (c) Use `Parte` para declarar una variable `a` de tipo `struct parte` y un arreglo `b[10]` de tipo base `struct parte`.

   (d) Lea un número y un nombre de parte desde el teclado y los almacene en los campos de la variable `a`.

   (e) Asigne los valores de los campos de la variable `a` al elemento 3 del arreglo `b`.

   (f) Imprima los campos del elemento 3 del arreglo `b`.

3. Dadas las siguientes definiciones y declaraciones en C++:

```
typedef struct fecha {
   int dia, mes, anio;
} Fecha;

struct emp {
   char nombre[30];
   unsigned int nroLegajo;
   unsigned short int edad;
   char estadoCivil;
   Fecha fechaIngreso;
} empleado;

struct espo {
   char nombre[30];
   Fecha fechaNac;
   unsigned short int edad;
} esposa;
```

¿Cuáles de los siguientes grupos de sentencias son válidas (desde el punto de vista del lenguaje C++)?

(a) `empleado.nombre[10] = esposa.nombre[10]; empleado.edad = esposa.edad;`

(b) `emp.nombre[10] = espo.nombre[10]; emp.fechaIngreso = espo.fechaNac;`

(c) `empleado.fechaIngreso = {1, 1, 2010};`

(d) `edad = 26; fechaIngreso.dia = 1; estadoCivil = 'c'; nombre ="Pedro Rodriguez";`

(e) `Fecha.dia = 10; Fecha.mes = 8; Fecha.anio = 1910;`

(f)
```
struct espo esposo = {"Juan", {10,4,1978}, 35};
esposa = {"Maria", {1,1,1980}, 33};
```

4. Dados los siguientes datos, declarar los struct que los representen:

   (a) Datos para calcular la medida de superficie de: un rectángulo, un triángulo y un trapecio.

   (b) Un par de coordenadas cartesianas (x, y) donde x e y son dos números reales.

   (c) Una dirección: calle y número, localidad, código postal, número de teléfono.

   (d) Un alumno: nombre (30 carácteres); número de Rol; Cédula de Identidad; dirección; Cursos (para cada curso: nombre, código, año de aprobación).

   (e) Un empleado: nombre (30 carácteres); Cédula de Identidad; dirección; fecha de nacimiento; estado civil; cantidad de hijos; sexo.

   (f) Un hotel: nombre; habitaciones, donde cada una tiene: una categoría (común, especial, suite, suite especial) y un tipo (individual, dos camas, doble, cuádruple). Además, para un hotel se quiere mantener la cantidad de habitaciones de cada tipo así como la cantidad de habitaciones de cada categoría.

5. Escribir una función que calcule la distancia entre dos puntos dados en el plano, usando el teorema de Pitágoras. Probar la función en un programa que solicite las coordenadas y muestre la distancia entre los puntos.

6. Hacer una función que ingrese los datos de un hotel y otra que los muestre por pantalla. Probar el uso combinado de las dos funciones en un programa.

7. Diseñar un struct para mantener el registro de salud de una persona. Los campos del registro deberán incluir el primer nombre de la persona, sus apellidos, la Cédula de Identidad, el género, la fecha de nacimiento (consistiendo de día, mes y año), el peso (en Kg) y la altura (en metros). El programa deberá contar con una función que pida los datos y los almacene en cada uno de los campos del registro de salud. Además, deberá incluir funciones que calculen y retornen la edad en años y el índice de masa corporal (IMC). El programa deberá solicitar la información de la persona, crear el registro para la persona y mostrar la información almacenada en dicho registro, incluyendo el nombre y apellido, el género, la fecha de nacimiento, la altura y el peso; luego deberá calcular y mostrar la edad en años de la persona y su índice de masa corporal, así como en qué categoría se encuentra de acuerdo a su IMC. El IMC se calcula con la fórmula:

$$\text{IMC} = \frac{\text{peso en kg}}{(\text{altura en metros})^2}.$$

   Se considera bajo de peso cuando el IMC es menor que 18.5; normal cuando es mayor o igual a 18.5 y menor que 25; con sobrepeso cuando es mayor o igual que 25 y menor a 30, y obeso cuando el IMC es 30 o más.

8. Hacer un programa que permita opcionalmente: (a) ingresar los datos de hoteles en un arreglo mientras el usuario lo desee; (b) mostrar por pantalla los datos de cada uno de los hoteles ingresados. Usar cuando sea necesario las funciones definidas en puntos anteriores.

Manejo de Memoria Dinámica

1. Dadas las siguiente declaraciones:

```
int *s, *q, t, ar1[10];
char *m= "HOLA", *n, l, ar2[11];
struct {
   int len;
   char *str;
} *p, ar3[10];
```

   Describa el efecto de las siguientes expresiones, teniendo en cuenta que se ha realizado la asignación `p = ar3`. Grafique las variables involucradas, e indique gráficamente en cada caso el elemento referenciado por la expresión.

   (a) `++p->len`

   (b) `(++p)->len`

   (c) `(p++)->len`

(d) `*p->str`

(e) `*p->str++`

(f) `(*p->str)++`

(g) `*p++->str`

2. Escribir la función `datosHoteles(int n)`, la cual permita ingresar los datos de `n` hoteles (como los definidos anteriormente en esta guía). Los datos de los hoteles deben ser devueltos por la función en un arreglo pedido dinámicamente.

3. Hacer un programa que permita ingresar una cantidad variable de cadenas de carácteres (strings) en un arreglo, y luego las imprima. Cada cadena puede tener, a su vez, un número variable de carácteres, con una longitud máxima de 1024 chars. El programa deberá definir y usar (o reusar) una función para ingresar los datos y otra para imprimir el arreglo, y para cada string deberá usarse el espacio de memoria exacto que permita almacenarla (es decir, no se admite desperdicio de memoria).

Manejo de Archivos

- Escriba un programa que lea un archivo de texto (cuyo nombre es especificado como argumento en la línea de comando), y calcule el promedio de las longitudes de todas las líneas de más de 20 carácteres. Si el nombre del archivo a utilizar no es especificado en la línea de comando, el programa deberá solicitárselo al usuario.

- Escriba la función `agruparLineas(nbreArchivo, l1, l2)`, tal que reciba como parámetros el nombre de un archivo, y dos valores enteros `l1` $<$ `l2`. La función debe agrupar las líneas comprendidas entre los valores `l1` y `l2` en una única línea. Dicho cambio debe ser reflejado en el archivo.

- Escriba la función `saveHoteles`, la que permita almacenar el contenido de un arreglo de hoteles en un archivo. Asuma la definición de hoteles vista anteriormente en esta guía. La función recibe como parámetro el arreglo de hoteles y el nombre del archivo en donde deben almacenarse. Si dicho archivo no existe, deberá crearse.

- Escriba la función `loadHoteles`, la que permita leer desde un archivo una cantidad especificada de hoteles, los que son almacenados en un arreglo. Asuma la definición de hoteles vista anteriormente en esta guía. La función recibe como parámetro el arreglo de hoteles en donde se almacenarán los datos leídos, la cantidad de hoteles a leer desde el archivo, y el nombre del archivo desde donde deben leerse los datos. Si dicho archivo no existe, la función deberá retornar un valor negativo (por ejemplo, -1). En otro caso, deberá retornar la cantidad de hoteles que fue capaz de leer.

- Suponga que un arreglo de hoteles ha sido almacenado en disco. Escriba la función

  ```
  hotel* readHotel(fstream& fp, int i);
  ```

  tal que retorne (como un puntero) el `i`-ésimo hotel almacenado en el archivo `fP`. Si dicho hotel no existe, debe retornar `NULL`.

- Suponga que un arreglo de hoteles ha sido almacenado en disco. Escriba la función

```
int writeHotel(fstream& fp, hotel* H, int i);
```

  tal que modifique el `i`-ésimo hotel almacenado en el archivo `fP` por el hotel `H`. Si el `i`-ésimo hotel no existe, debe retornar -1.

- Se desea escribir una carta de felicitación navideña a los empleados de una empresa. El texto de la carta se encuentra en el archivo `CARTA.TXT`. El nombre y dirección de los empleados se encuentra en el archivo binario `EMPLEADOS.DAT`, como una secuencia de registros con los campos nombre, dirección, etc. Escribir un programa que genere un archivo de texto por cada empleado. La primera línea contiene el nombre, la segunda está en blanco, la tercera la dirección, la cuarta está en blanco, y en la quinta empieza el texto del archivo `CARTA.TXT`.

Tabla de Precedencia y Asociatividad de los Operadores en C++

La siguiente tabla muestra la precedencia y asociatividad de los operadores en C++. Los operadores que aparecen en una misma línea tienen la misma precedencia. La precedencia disminuye de arriba hacia abajo en la tabla.

Operadores	Asociatividad
`() . [] ->`	izquierda a derecha
`! ~ ++ -- + - * & (tipo) sizeof`	derecha a izquierda
`* / %`	izquierda a derecha
`+ -`	izquierda a derecha
`<< >>`	izquierda a derecha
`< <= > >=`	izquierda a derecha
`== !=`	izquierda a derecha
`&` (and bit a bit)	izquierda a derecha
`^`	izquierda a derecha
`|`	izquierda a derecha
`&&`	izquierda a derecha
`||`	izquierda a derecha
`?:`	derecha a izquierda
`= += -= *= /= %= &= ^= |= <<= >>=`	derecha a izquierda
`,`	izquierda a derecha

La columna "Asociatividad" indica la manera en que operadores de una misma precedencia son asociados cuando aparecen en una expresión. Así, por ejemplo, la expresión `a = b = c` es equivalente a `(a = (b = c))`, ya que la asociatividad del operador `=` es de derecha a izquierda. Por otro lado, la expresión `a * b / c` es equivalente a `((a * b) / c)`, ya que la asociatividad de `*` y `/` es de izquierda a derecha.