Ejercicios de practica

Teoria

La computador. realiza la mayor parte del trabajo con números. Incluso las cadenas, `strings` son solo una abstracción, porque cada carácter es un número en la memoria de la computadora. En `C`, por supuesto, existen facilidades para trabajar con números. Familiaricémonos con el nuevo tipo - `int`:

```cpp
int answer = 42;
printf("El numero es %i", answer);
```

Resultado:

```
El numero es 42
```

- `int` es de tipo entero, es decir, es adecuado para almacenar números enteros de ambos signos: `positivo` y `negativo`.

El estándar `C/C++` no especifica qué números específicos se pueden almacenar en una variable `int`. Pero en todos los compiladores populares, estos son números en el rango de -2^{31} a $2^{31} - 1$, es decir, de aproximadamente menos dos mil millones a dos mil millones.

`C/C++` se puede utilizar como una calculadora. Las fórmulas se escriben como en matemáticas. Posibles operaciones:

- `+` para suma,
- `-` restar o cambiar el signo de un número,
- `*` para la multiplicación,
- `/` por división,
- `%` para tomar `el resto de la división de números positivos,`
- `(` y `)` para determinar el orden de las operaciones.

Por ejemplo, necesita transportar `5` camiones con robots en un ferry, pero dudo que un ferry con una capacidad de carga de `10 000` kg sobreviva. La masa del camión es de `1200` kg, la masa del robot es de `100` kg, un camión puede acomodar `8` robots. Calcular y derivar la masa total con la siguiente fórmula:

```c
int total = 5 * (1200 + 100 * 8);
printf("El total es: %i", total);
```

El resultado es `10.000`, el transbordador debe resistir. Sin embargo, es fácil confundirse en dicho código. Cuando regrese a él despues de una semana, lo más probable es que ya no recuerde qué número significa qué. Las variables arreglarán la situación:

```language-cpp
int number_of_trucks = 5;    // number of trucks
int truck_weight = 1200;     // truck weight
int robot_weight = 100;      // robot weight
int robots_per_truck = 8;    // robot per truck

// total weight
int total_weight = number_of_trucks *
                   (truck_weight + robot_weight * robots_per_truck);

printf("El peso total es: %i, "total_weight);
```

Las variables ayudan a documentar el código. Puede combinar libremente números y valores de variables en una expresión. Reescribamos ligeramente este código:

```language-cpp
printf("%i", 5 * (truck_weight + robot_weight * robots_per_truck));
```

Pero mejor no ser perezosos y nombrar cada valor para que no parezca sacado del techo. Se permite especificar varias variables del mismo tipo en una declaración. Las diferentes variables deben estar separadas por una coma. Se pueden escribir en una o más líneas:

```language-cpp
int number_of_trucks = 5, truck_weight = 1200,
    robot_weight = 100, robots_per_truck = 8;

int total_weight = number_of_trucks *
                   (truck_weight + robot_weight * robots_per_truck);

// Mostrar la variable total_weight
```

Cuando se escribe un nuevo valor, se olvidan todos los valores anteriores. Las variables no tienen un historial, y es imposible averiguar qué había antes en la variable. Esto se debe a que se esta sobreescribiendo ese espacio de RAM en el cual existe nuestra variable. Sin embargo, puede utilizar el valor actual para calcular uno nuevo. Por lo tanto, el gerente de la tienda no tendría que resumir la cantidad manualmente y el código se volvería más transparente:

```language-cpp
int production = 20;          // Valor del primer dia
production = production + 25;
production = production + 15;
production = production + 26;

// Mostramos el resultado que es - 86
```

La operación de aumentar un valor es tan común que se ha inventado una abreviatura especial `+=`. Este código se puede simplificar:

```cpp
int production = 20; // Valor el primer dia
production += 25; // Este comando - aumenta el valor de production en 25
production += 15;
production += 26;

// Mostramos el resultado - 86
```

Las operaciones `*=`, `-=`, `/=` e incluso `%=` funcionan de manera similar.

¿Los operadores `+=`, `*=`, `-=`, `/=` y `%=` son de lectura o escritura?

- Son para leer y escribir al mismo tiempo.

Ejercicios

Ejercicio 1

Aquí está el robot de rescate RS-99. En cualquier caso, así lo vio el artista abstracto Benedict Hunt, impresionado por el trabajo de la máquina.

```
-------------
|  /""""""\  |
| /        \ |
| | 0    0 | |
| |    v   | |
| \  ___   / |
|  \_____/   |
-------------
```

Hunt cree que los pinceles y las pinturas no son para artistas progresistas y que solo una máquina puede crear un verdadero retrato de una máquina.
Hunt necesitaba un programa apropiado para la exposición. Escriba un programa para reproducir el dibujo ilustrado arriba.
Hay `13` caracteres en cada una de las ocho líneas de la figura. Tenga en cuenta: la parte superior del marco, la boca y la barbilla se dibujan con guiones bajos. La parte inferior del marco - menos. El cabello está entre comillas dobles, los ojos están hechos con ceros y la nariz está hecha con `v`.
Durante el desarrollo, es útil y conveniente verificar periódicamente lo que está haciendo. Para hacer esto, ejecute su codigo de forma periodica, esto para verificar que se esta obteniendo el dibujo que se desea.

Requisitos

- Imprima cada línea de la imagen con un comando separado que comience con `printf` y termine con `;`.
- Reproduzca el dibujo exactamente como se muestra. No imprima espacios o símbolos adicionales. Para simplificar el trabajo, puede copiarlo del documento.
- Blinde cuidadosamente los caracteres especiales en el dibujo.
- El marco también es parte de la imagen.
- Pase los tests, corra el comando `make test_ex1`

Ejercicio 2

Se quiere imprimir el siguiente mensaje:

```
The quote from "Hamlet":
To be, or not to be, that is the question:
Whether 'tis nobler in the mind to suffer
The slings and arrows of outrageous fortune,
Or to take arms against a sea of troubles
And by opposing end them.
```

Para este objetivo nuestro colega Anton ya ha realizado una gran parte del trabajo. Sin embargo, esta teniendo problemas al momento de correr su codigo. Se acerca y nos comunica que su programa tiene varios errores y no logra encontrarlos. Revise y corrija el codigo de Anton para obtener el mensaje deseado:

```language-cpp
cout << "The quote from "Hamlet":"s << endl;
cout "To be, or not to be, that is the question:"s endl
cout << Whether 'tis nobler in the mind to suffer << Endl
cout << "The slings and arrows of outrageous fortune,"s endl;
cout << "Or to take arms against a sea of troubles"s << endl;
cont << "And by opposing end them."s << end;
```

Requisitos

- Copie el codigo de este documento a el editor de texto.
- Conviertalo a lenguage C.
- Corrija todos los errores que encuentre y trate de correr el codigo
- Una vez haya corregido todos los errores asegurese de obtener exactamente el mensaje que se muestra mas arriba.
- No se olvide de agregar los puntos, dos puntos, verifique cada detalle de su mensaje
- Pase los test corriendo el comando `make test_ex2`

Ejercicio 3

El fabricante de robots SCZ Dynamics necesita un programa para mostrar información sobre la ubicación del robot.
Se sabe que la inicial del robot se almacena en la variable `letter` del tipo `char`, y la tarea se almacena en la variable `task`, también del tipo `char`. Es necesario mostrar información sobre el robot.
Declare estas variables pero no las inicialice en la misma linea. No se olvide de colocar el tipo de dato correcto de la variable.

Requisitos

- Declare las variables en esta tarea. Considere el tipo de la variable y el nombre de la variable. Este debe ser descriptivo.
- Tu misión es generar una frase que contenga los valores de las variables, sean cuales sean esos valores.

Ejemplo

- Si la inicial es `R` y la tarea es `A`, entonces muestre:

```
Se solicita al robot R para A.
```

- Si el nombre es `9` y la tarea es `E`, muestre:

```
Se solicita al robot 9 para E.
```

Formato de salida

Es necesario generar los datos en el siguiente orden:

- La frase `Se solicita al`
- la palabra `robot`;
- la inicial del robot;
- La palabra `para`;
- tarea de robot;
- punto;
- nueva linea
- Se espera que se muestren los dos mensajes de arriba. Corra las pruebas con `make test_ex3`

Los datos deben estar separados del texto por espacios.

Ejercicio 4

Cuando llega un nuevo requisito del cliente: el agente Mooney implementará la salida de información. Su tarea es preparar los datos para ello.
Cree las siguientes variables:

- `robot_name` con el valor de nombre: `RC-21\\M2`
- El nombre del robot cambia luego a `"RC-21\\M2"`

Utilizando las variables previamente creadas muestre el mensaje:

```
El robot RC-21\\M2 tiene un mensaje.
El robot "RC-21\\M2" se está incendiando.
```

Formato de salida

- En la variable `robot_name` la `string`
- Note el punto al final de cada mensaje
- Corra las pruebas con `make test_ex4`

Ejercicio 5

Un agente creo dos variables de tipo `char`, pero mezcló sus valores. Nombró las variables: `x` e `y`. Su tarea es intercambiar los valores de estas variables, sean cuales sean.
El agente trató de resolver el problema por su cuenta, pero algo salió mal. Su intento está en el código. Por favor, corrija la solución del agente. Considere que se deberá crear una nueva variable.

Ejemplo

Supongamos que los valores iniciales de las variables se dan de la siguiente manera:

```
char x = 'A';
char y = 'B';
```

Después de correr el programa, los valores deberían cambiar. `x` debería ser `B` e `y` debería ser `A`.
Veamos cómo funciona el código del agente en este caso.

```
int main() {
  char x = 'A';
  char y = "B";

  x = y;
```

```
  printf("x = %c", x)
  printf("y = %c", y);

  y = x;
  printf(x = "%c", x);
  printf("y = %c", y);

}
```

Primera fila de la solución `x = y`. Después de su ejecución se muestra el contenido de cada variable y se observa que las variables quedan de la siguiente manera:

```
x - B
y - B
```

La siguiente línea de la solución de Mooney `y = x` no cambiará nada, ya que los valores de las variables ya son los mismos.

Considere otros valores iniciales:

```
x — M
y - L
```

Después de ejecutar su programa, los valores deberían cambiar:

```
x - L
y - M
```

Veamos qué sucede cuando se inicia el programa del agente. Después de la primera línea `x = y` los valores son:

```
x - L
y - L
```

Y no cambian después de eso.

Su trabajo es corregir este codigo para que cumpla con lo requerido. Revise el codigo tambien tiene errores de sintaxis.

Formato de salida

Su programa debe usar `cout` y debe imprimir en la terminal las variables `x` y `y`.

Mensaje esperado:

```
x - s
y - o

Antes del programa
La variable x es s
La variable y es o

Despues del programa
La variable x es o
La variable y es s
```

- Corra las pruebas `make test_ex5`

Ejercicio 6

Robert no puede recordar cuándo felicitar a sus amigos por sus cumpleaños. Así que decidió escribir un programa para calcular fechas memorables.
La variable `n` almacena la inicial del amigo, la variable `age` almacena el número de años y la variable `birth_year` almacena el año de nacimiento. Su tarea es decir en qué año la persona con el nombre especificado será el número de años especificado. Declare estas variables con sus tipos respectivos; considere que cuales van a ser de tipo `char` y cuales tienen que ser de tipo `int`.

Ejemplo

Supongamos que los valores de las variables son:

- `name` - Jose
- `age` - 10
- `birth_year` - 1746
- El programa debería generar el texto `Jose cumplira 10 anohs en 1756.`

Si se establecen los siguientes valores:

- `name` - Jesus
- `age` - 49
- `birth_year` - 1770
- El mensaje correcto seria: `Jesus cumplira 49 anhos en 1819.`

Formato de salida

Imprima los siguientes datos en la secuencia `printf`:

- name (debera usar %s para imprimir una string)

- cumplirá (Note el acento)
- Edad + `años`
- La palabra `en`
- El año en el que se cumplirá el número especificado de años.
- Punto al final del mensaje
- No se olvide de agregar los espacios donde corresponda
- Corra las pruebas `test_ex6`
- Revise los ejemplos de arriba.

Ejercicio 7

Desarrolle un programa para mostrar información sobre un número entero: entre qué números está y cuánto es igual a dos veces el numero. El número en sí se debe almacenar en una variable `x` de tipo `int`.

Ejemplo

Si `x = 100`, entonces necesita imprimir la siguiente frase:

- `El numero 100 esta entre 99 y 101. El doble de 100 es 200.`

Si el valor de `x es 55`, entonces debería salir:

- `El numero 55 esta entre 54 y 56. El doble de 55 es 110.`

Formato de salida

Envíe el siguiente texto a la secuencia `printf`: `El numero {x} esta entre {y} y {z}. Doblado {x} es {t}.`, en el que:

- en lugar de `{x}` debería estar el número original de la variable `x`,
- en lugar de `{y}` - el número anterior a `x`,
- en lugar de `{z}` - el siguiente número `x`,
- en lugar de `{t}` - el doble de `x`.

Ejercicio 8

Cuando se transportan robots, se utilizan cajas de madera contrachapada, que se fabrican directamente en la fábrica. La caja consta de un fondo y cuatro paredes, y la tapa está hecha de plexiglás transparente. El mes pasado hubo un tiempo de inactividad debido a que la empresa se quedó sin madera contrachapada. Para evitar que esto vuelva a suceder, se le asignó la tarea de escribir un programa para controlar la madera contrachapada restante.

Se debe calcular la cantidad restante de madera contrachapada y mostrar esta cantidad después de la fabricación de cada uno de los tres robots. Para hacer esto, cree las siguientes variables:

- `wood_balance` - cantidad inicial de madera contrachapada, m²;
- `w1`, `w2`, `w3` - son el ancho de la caja del primer, segundo y tercer robot, m;
- `d1`, `d2`, `d3` - son la profundidad de la caja del primer, segundo y tercer robot, m;
- `h1`, `h2`, `h3` - son la altura de la caja del primer, segundo y tercer robot, m.

Después de que se haya ejecutado el programa, la cantidad de madera contrachapada restante debe almacenarse en la variable `wood_balance`.

Las variables `wood_balance`, `w1`, `w2`, `w3`, `d1`, `d2`, `d3`, `h1`, `h2`, `h3` se deben declarar con su tipo correspondiente.

Formato de salida

Escriba la frase: `Madera contrachapada restante:` seguida de la cantidad de madera contrachapada restante. Haz esto para cada una de las tres cajas.

Ejemplo

Supongamos que todas las medidas de todas las cajas son iguales a un metro. El balance inicial de madera es de `1000 m²`. El consumo por caja en este ejemplo es de `5 m²`, por lo que el programa debería generar tres líneas:

```
Madera contrachapada restante: 995
Madera contrachapada restante: 990
Madera contrachapada restante: 985
```

El valor final de la variable `wood_balance` debería ser 985 despues de la ejecucion del programa.

Ayuda

- Necesitará el operador `-=` para disminuir la variable `wood_balance`.
- Calcula el área de la caja usando la fórmula:

$$S = w \cdot d + 2 \cdot (w \cdot h + d \cdot h)$$

  El esquema de su programa debería verse así:
- Disminuir el valor de la variable `wood_balance` utilizando las variables `w1`, `d1`, `h1` y la formula de arriba. (Puede crear una variable de ayuda)
- Mostrar la información en `cout`.
- Disminuimos el valor de la variable `wood_balance` utilizando las variables `w2`, `d2`, `h2` y

la formula de arriba. (Puede reusar la variable de ayuda)
- Mostramos información en `cout`.
- Disminuimos el valor de la variable `wood_balance` utilizando las variables `w3`, `d3`, `h3` y la formula de arriba. (Puede reusar la variable de ayuda)
- Mostramos información en `cout`.

Los tres comandos de salida tienen el mismo aspecto, pero los comandos de disminución, `-=`, difieren en los valores de las variables.
Para las pruebas use los siguientes valores:

```
wood_balance = 1000

w1 = 1 d1 = 2 h1 = 3
w2 = 1 d2 = 2 h2 = 3
w3 = 1 d3 = 2 h3 = 3
```

- Pruebe en papel cuanto deberia salir.
- Compare sus resultados obtenidos en papel con los obtenidos por su programa
- Corra las pruebas `make test_ex8`