Ejercicios: Homework 2

Ejercicio 1

Escriba un programa que lea dos números `double` e imprima la parte entera de su suma.

Ejemplo

- Entrada: 6 2.75
- Salida: 8

Ejercicio 2

Escriba un programa que lea dos números `double` e imprima su cociente.

Ejemplo

- Entrada1: 5 2
- Salida1: 2.5
- Entrada2: 5 0
- Salida2: inf

Ejercicio 3

Solucione los errores del programa que pide que ingrese dos números enteros: el ancho y el alto de un rectángulo, y luego muestra su área.

```
#include <iostream>

int main() {
    int width height;
    cout << "Enter rectangle width and height"s << endl;
    cin << width << height;
    cout << "Rectangle area is "s << width * height << endl;
}
```

Ejercicio 4

El programa debería leer dos números enteros, `x` e `y`, e imprimir `"Your answer: "` y la seguido de la suma de los números `x + y + z`, donde `z = 1`, en una línea. Pero hay

errores de compilación en el código. Corríjalos uno por uno y vea cómo cambian los mensajes de error.

```
#include <io_stream>

/* using namespace std; */

int ma() {
    int x;
    cin >> x, y;

    z = 1;
    cout << 'Your answer: ';
}

cout < x + y + z;
```

Ejercicio 5

No todos los errores se pueden detectar durante la compilación del programa. Algunos de ellos sobreviven hasta el momento de su `deployment` (Lanzar a produccion). Aquí está el código de un programa que debe leer números del tipo `doble` y mostrar su suma. Hay un error en el código. Corrijalo.

```
#include <iostream>

using namespace std;

int main() {
    int x, y;
    cin >> x >> y;
    cout << x + y;
}
```

Ejercicio 6

Escriba un programa que lea el nombre y el apellido separados por un espacio y genere el nombre completo en el formato `<Apellido>`, `<Nombre>`. Utilice el operador de suma para concatenar (unir) partes de cadenas en una sola cadena antes de generarlas, o imprímalas individualmente en el orden deseado.

Formato de E/S
Entrada
El `nombre` y `apellido` en la entrada se dan en una línea y separados por un espacio.
Use `cin >>` para leer nombres y apellidos.

Salida
Primero se muestra el apellido, luego una coma y un espacio, luego se muestra el nombre.

Ejemplo

- Entrada: Ivan Cabrera
- Salida: Cabrera, Ivan

Ejercicio 7

Escriba un programa que lea los títulos de tres libros, uno por línea, y los imprima en orden inverso, también uno por línea. El título de un libro puede tener más de una palabra, es decir el inicio del titulo de un nuevo libre no esta separado por espacios si no por una nueva linea.

Ejemplo
Entrada

```
Moby Dick
The Tiger Who Came to Tea
Harry Potter and the Deathly Hallows
```

Salida

```
Harry Potter and the Deathly Hallows
The Tiger Who Came to Tea
Moby Dick
```

Teoria (Para el ejercicio 7)
Cuando se lee una línea desde `cin`, se lee hasta el primer carácter `delimitador`: `espacio`, `tabulador` o `nueva línea`. En este caso, los espacios entre palabras no se leen:

```
string name;
string surname;
cout << "Enter your name:"s << endl;
cin >> name >> surname;
cout << "Hello, "s << name << " "s << surname << endl;
```

Si el usuario ingresa la cadena `John Doe`, entonces la palabra `John` se guardara en la variable de `name` y la palabra `Doe` en la variable de `surname`.
Para leer la línea completa, no solo la palabra actual, use la función `getline`. Lee

caracteres del flujo de entrada en una cadena hasta que se encuentra un carácter de `final de línea` o se produce un error.
Este programa lee dos líneas: un nombre y una dirección. Las líneas de entrada pueden constar de varias palabras:

```
string full_name, address;
getline(cin, full_name);
getline(cin, address);
cout << "Hello, "s << full_name << " from "s << address << endl;
```

Metemos los siguientes datos al programa de arriba:

```
Harry Potter
4 Privet Drive, Little Whinging, Great Britain
```

La salida del programa es:

```
Hello, Harry Potter from 4 Privet Drive, Little Whinging, Great Britain
```

En este programa, `cin` actúa como parámetro de la función `getline`. En `C++`, además de `cin`, hay otros flujos de entrada con los que puede trabajar `getline`. Se aprendera sobre ellos en lecciones futuras.

Ejercicio 8

Lea cinco palabras de la entrada estándar e imprima el primer caracter de cada palabra de entrada. Una palabra es una secuencia de caracteres como letras, números, signos de puntuación. Las palabras están separadas por un solo espacio.

Ejemplo

- **Entrada:** apple banana cat dog eleven
- **Salida**: abcde
- **Entrada:** a s d f g
- **Salida**: asdfg
- **Entrada:** 1 two 3 four 5
- **Salida**: 1t3f5

Teoria (Ejercicio 8)

Las cadenas, `strings` permiten que un programa funcione con datos textuales: nombres de personas y nombres de animales, el contenido de documentos de texto, direcciones de correo electrónico y páginas web.

Las cadenas en C++ están representadas por el tipo de `string`. Para usarlo, debe incluir la biblioteca `<string>`. En esta lección, aprenderá sobre los elementos de las cadenas: `char`.

Una cadena es una secuencia arbitraria de unidades elementales, símbolos del alfabeto. En cadena, cada elemento es de tipo `char`. En la mayoría de las plataformas modernas, un `char` tiene `ocho bits` y puede tener hasta $2^8 = 256$ valores diferentes. Esto es suficiente para representar los caracteres del alfabeto inglés, números, signos de puntuación y una serie de caracteres de servicio.

Para representar caracteres cirílicos, idiomas de Medio Oriente, jeroglíficos y emoji, un carácter ya no es suficiente. Según la codificación utilizada, es posible que se requieran dos o más caracteres.

En C++, el tipo `char` almacena un número igual al código del carácter. Por ejemplo, la letra `A` mayúscula en inglés en la mayoría de las plataformas tiene un código de carácter de `65`. Los caracteres literales se usan a menudo para representar caracteres, en los que el carácter está encerrado entre comillas simples. Entonces el código se vuelve más claro:

```
char letter_a = 'A';    //  Se entiende que en el codigo se guardara la
letra A
char letter_a_too = 65; // Tambien se guarda la letra A, pero no lo podemos
saber a simple vista

char letter_b = letter_a + 1; // La letra B tiene el codigo siguiente a la
letra A (65 + 1)
```

Para acceder a caracteres individuales de una cadena, se define una operación de indexación. Para hacer referencia a un carácter de cadena, debe especificar su número (índice) entre corchetes `[]`. La indexación de elementos de cadenas y matrices en C++ comienza desde `cero`. Después del último carácter de la cadena hay un carácter de servicio con el código `0`. Es necesario para la compatibilidad con las cadenas C.

```
string greeting = "Hello"s;
char ch = greeting[1];
cout << ch; // Obtenemos el simboo e. Ya que el primer simbolo seria con el
indice 0 (la letra H)
```

El operador de indexación también se puede usar para cambiar los caracteres de una cadena:

```
// En greeting estaba la palabra Hello
greeting[0] = 'Y'; // Ahora en greeting se contiene "Yello"
```

Ten cuidado. El índice de caracteres debe estar en el rango `[0; longitud de la string)`. **Si especifica un índice fuera de este rango, el comportamiento del programa será indefinido.**
Los operadores `+` y `+=` pueden agregar no solo cadenas, sino también caracteres individuales a una cadena:

```
// En greeting habia la palabra Yello
greeting += 'w';   // Dentro de la string greeting ahora se tiene "Yellow".
Esta liena de codigo equivale a greeting = greeting + 'w'

cout << greeting << endl;
```

El tipo `char` no almacena una cadena, sino un código de carácter. Por lo tanto, si agrega dos valores del tipo `char`, no será la concatenación de caracteres en una cadena, sino la adición de códigos de caracteres. El resultado será un número:

```
cout << 'A' + 'B' << endl; // Se obtiene el numero 131, el cual es igual a
la suma de 65 y 66 (los codigos para A y B, respectivamente)
```

Ejercicio 9

Escriba un programa que lea dos enteros no negativos separados por un espacio. El programa debe calcular su suma S y generar el producto de S por el número de dígitos en la notación decimal S.

Ejemplos

- **Entrada**: 3 8
- **Salida**: 22
- **Explicacion:** La suma de los dos enteros es `3 + 8 = 11`. Los numeros de digitos de la suma `11` es igual a `2` digitos. Entonces, la multiplicacion `2 * 11 = 22`.
- **Entrada**: 8 17

- **Salida**: 50

- **Explicacion:** La suma de los dos enteros es `8 + 17 = 25`. Los numeros de digitos de la suma `25` es igual a `2` digitos. Entonces, la multiplicacion `2 * 25 = 25`.

Teoria (Ejercicio 9 y 10)
Seguramente ha notado en algunos ejercicios anteriores de C++ el uso de `""s`-literals cuando se trabaja con cadenas. Ahora, aprenderá sobre situaciones en las que tales literales protegen su código de errores.

Concatenación de literales de cadena
Ya sabes que el operador `+` se usa para concatenar cadenas. Funciona bien con variables, incluso cuando se inicializan con literales de cadena regulares, sin la s después de las comillas:

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

int main() {
    string i_know = "I know "; // Simple string sin la al final `s`
    string cpp = "C++";        // Simple string sin la al final `s`
    cout << i_know + cpp << endl;
}
```

Sin embargo, usar `+` para agregar literales de cadena normales no funcionará:

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

// Pruebe este codigo en el compilador
// Que observa?
int main() {
    cout << "I know " + "C++" << endl;
}
```

El error obtenido se debe a que el operador `+` no se puede usar para agregar tipos `const char [8]` y `const char [4]`. Los literales de cadena `"I know"` y `"C++"` son literales de cadena `C` o `cadenas C`. El compilador los convierte en matrices de caracteres, que se rellenan con el código de carácter 0. Es por eso que el tamaño de la matriz es uno más que el número de caracteres dentro de las comillas. Sea como fuese, las matrices de caracteres no son cadenas y no tienen operaciones `+`.

Para corregir el error, use `""s`-literals, que crean directamente un objeto de tipo cadena. Cuando el tipo de al menos uno de los sumandos es cadena, la operación de suma funcionará como se esperaba:

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

int main() {
    cout << "I know "s + "C++"s << endl;
    // Lo siguiente tambien funciona
    // cout << "I know"s + "C++" << endl;
    // cout << "I know" + "C++"s << endl;
}
```

Adición de cadenas y números
JavaScript, Java, C# y varios otros lenguajes le permiten agregar cadenas y números. Pero en C++, no puede agregar cadenas y números:

```
// Intentelo en el compilador y revise el error
string s = "Hello"s;
string error = s + 1; // Error!
```

Las cadenas y los números son tipos diferentes, y para ellos, la operación `+` realiza acciones completamente diferentes: concatenación de cadenas y adición de números. En `JavaScript`, si uno de los argumentos de una operación de suma es una cadena, el otro argumento se convierte implícitamente en una cadena. Entonces las cadenas se concatenan.
En `C++`, la conversión de un número a una cadena y viceversa debe hacerse explícitamente. Las funciones para esto son:

- `stoi` - String to Integer
- `stod` - String to Double
- `to_string` - To String

```
// Pruebe este codigo en el compilador (use cout para verificar las
variables)
int x = stoi(x_str);      // string -> int
double y = stod(x_str);   // string -> double
string s = to_string(x);  // int or double -> string

int z = 42;
string t = "Hello"s + to_string(z); // En t se guardara el valor "Hello42"
```

Las funciones `stoi` y `stod` fallarán si la cadena que se les pasa no se puede convertir a un número entero o real. Dichos errores deben manejarse adecuadamente, de lo contrario, el programa fallará. Cómo procesarlos exactamente, se vera en lecciones futuras.

Al agregar cadenas y números, se produce un error de compilación. En particular, esto sucede cuando se suman `""s`-literales y enteros. Los `""s`-literals protegen contra los errores ocasionales que son inherentes a los literales de cadena normales.
`C++` heredó de `C` la capacidad de agregar cadenas literales simples a los números. En este caso, el código compila sin errores, pero cuando se ejecuta, se obtiene un resultado inusual, que incluso puede provocar que el programa se rompa:

```
string s = "Hello" + 1; // En la variable s tendremos el valor "ello"
string q = 1 + "Hello"; // En la variable q tambien habra el valor "ello"

// Aquí el comportamiento del programa será indefinido,
// ya que el argumento numérico excede la longitud de la cadena:
// string e = "Hello" + 10000;
```

Mientras tanto, siempre use `""s`-literals cuando trabaje con cadenas. Protegerán su código de sorpresas.

Tamaño de string
La función de `size` le permite averiguar el tamaño de la cadena. La sintaxis para llamarlo: primero viene el objeto de cadena, luego el nombre de la función se escribe a través del punto:

```
// La funcion size es invocado en una cadena creada por un literal de cadena
"Hello"s
int length = "Hello"s.size(); // El valor de length es igual a 5

string greeting = "Hi"s;
// La funcion size se llama en la string greeting
int greeting_size = greeting.size(); // El valor de greeting_size es 2
greeting = "Bye"s;
greeting_size = greeting.size(); // El valor de greeting_size es 3
```

Estas funciones se denominan funciones miembro o métodos. Veremos más información sobre los métodos más adelante en el curso.
El tamaño de la cadena devuelta por el método de tamaño se calcula en elementos de tipo `char`. Esta no es necesariamente la misma cantidad de caracteres gráficos que verá el usuario:

```
cout << "Hello"s.size() << endl // 5
     << "Привет"s.size() << endl // 12 — cirílico en UTF-8
     << "👋"s.size() << endl; // 8 — emoji
```

Note que si una cadena consta únicamente de letras latinas, números, espacios y signos de puntuación básicos, su tamaño será igual a su longitud. Por lo tanto, a menudo hablaremos de la longitud de una cadena, es decir, de su tamaño.

Ejercicio 10

Escriba un programa que lea un número (número de documentos) y muestre `<número de documentos> documentos encontrados`. Primero, forme una cadena con la respuesta, solo luego muéstrela en la pantalla en su totalidad.

Ejemplo

- Formato de entrada: 2
- Formato de salida: 2 documentos encontrados

Ejercicio 11

Julia perdió a su cachorro Fibo, que recibió su nombre en honor al matemático Fibonacci. El signo especial de Fibo es que tiene grabado en su collar el código `11235813`. Escriba un programa que use el código del collar para determinar si pertenece a Fibo.

Formato de entrada y salida
El flujo de entrada estándar, `cin` , recibe un numero entero, el código del collar del perro. Si se trata de un collar de Fibonacci, el programa debe mostrar la línea: `"Se encontró Fibonacci".` Si este no es un collar de Fibonacci, el programa debe mostrar - `"Esto no es de Fibonacci"`.

Ejemplos

- **Entrada**: 3
- **Salida**: Esto no es de Fibonacci
- **Entrada**: 1123
- **Salida**: Esto no es de Fibonacci
- **Entrada**: 11235813
- **Salida**: Se encontró Fibonacci

- **Entrada**: 11235814
- **Salida**: Esto no es de Fibonacci

Ejercicio 12

Compras un smartphone en una tienda con un sistema de descuento flexible. Si el costo original del teléfono es más de `A` bolivianos, se establece un descuento del `X` por ciento. Si el costo original del teléfono inteligente es más de `B` bolivianos, el descuento será `Y` por ciento.

- El programa recibe cinco números reales como entrada: `N, A, B, X, Y`.

En donde, `N` - el costo inicial del producto. El número `A` es menor que el número `B`.

Imprima el costo de comprar un teléfono, incluido el descuento.

Ejemplos

- **Entrada**: 100 110 120 5 10
- **Salida**: 100
- **Entrada**: 115 110 120 5 10
- **Salida**: 109.25
- **Entrada**: 150 110 120 5 12.5
- **Salida**: 131.25

Ejercicio 13

En anteriores ejercicios, se escribió un código de programa que: lee un número — el número de documentos; muestra cuántos documentos encontrados: `<número de documentos> documentos encontrados.`
El programa funciona, pero la palabra "documento" siempre está en plural, por ejemplo, se encontraron `1 documentos`. Edite el código. Si la entrada es 0, deje que el programa genere `No se encontraron documentos`, si es 1: `Se encontró un documento`. En todos los demás casos, el programa debe generar `<número de documentos> documentos encontrados.`

Ejemplos

- **Entrada**: 0
- **Salida** No se encontraron documentos
- **Entrada**: 1
- **Salida** Se encontró un documento
- **Entrada**: 2
- **Salida** 2 documentos encontrados.

Ejercicio 14

Martha perdió a su gato Marqués. Un transeúnte le dijo que vio un gato similar afuera de una de las tres casas cerca de la oficina de correos. Cuando se le pregunta en cuál de las casas vio al Marqués, un transeúnte puede recordar el color de la casa, así como el número de pisos en la casa. Una situación posible es cuando un transeúnte ha olvidado parte de la información o toda ella.
Escribe un programa que, basado en el testimonio de un transeúnte, ayude a Martha a decidir a cuál de las tres casas entrar.

Formato de datos de entrada y salida
La entrada al programa son los siguientes datos, separados por un espacio:

- el color de la primera casa y el número de pisos en ella,
- el color de la segunda casa y el número de pisos en ella,
- el color de la tercera casa y el número de pisos en ella,
- el color de la casa y el número de pisos informado por un transeúnte. Si un transeúnte no recuerda el color de la casa o el número de pisos, la string sera `?` (signo de interrogación), y en lugar del número de pisos - el número `-1`.
  El programa debe mostrar, en orden ascendente, los números de las casas que Martha necesita visitar, una en cada línea. Los números de casa se muestran como los números `1`, `2` o `3`. Los números de casa que no necesita visitar no deben mostrarse.

Ejemplo
Entrada

```
red 3 green 3 green 2 green -1
```

Salida

```
2
3
```

En este ejemplo, el transeúnte solo recuerda el color de la casa, `verde`, pero no el número de pisos. Por lo tanto, Marta necesita ir a las casas número `2` y `3`.

Entrada

```
red 2 green 1 red 3 red 3
```

Salida

```
3
```

Aquí, un transeúnte informó tanto el color de la casa como su número de pisos: `red 3`. Por lo tanto, Marta solo necesita visitar la casa número 3.

Entrada

```
red 1 green 2 blue 3 yellow 3
```

Salida

```

```

El testigo dijo que vio un gato en una casa amarilla de tres pisos, pero no hay tal casa cerca de la oficina de correos: la única casa de tres pisos es azul. Probablemente, el transeúnte mezcló todo, esto también sucede.

Entrada

```
grey 5 grey 5 grey 5 grey 5
```

Salida

```
1
2
3
```

El gato está en uno de estos edificios grises de cinco pisos. Tiene que ir a las tres casas.

Ejercicio 15

Se te da una letra, un tipo de dato `char`. Se tiene que encontrar si el caracter dado esta en mayúsculas, esta en minúsculas o un carácter no alfabético.

Entrada

- Un `caracter`

Salida

- Si el caracter esta en mayúsculas, su programa debe imprimir `el caracter esta en mayúsculas` en la consola.
- Si el caracter esta en minúsculas, su programa debe imprimir `el caracter esta en minúsculas` en la consola.
- Si el caracter no es alfabético, su programa debe imprimir `No es un caracter` en la consola.

Ejemplo

- **Entrada**: c
- **Salida**: el caracter esta en minúsculas
- **Entrada**: A
- **Salida**: el caracter esta en mayusculas
- **Entrada**: 1
- **Salida**: No es un caracter

Ejercicio 16

Suponga que conoce la `edad` y `los años de experiencia` de un candidato y desea encontrar el puesto de mayor jerarquía al que puede postularse. Tienes la siguiente información:

- Los puestos disponibles son los de `coordinador de proyecto`, `director de proyecto` y `director de proyecto senior`.
- Un candidato no puede tener menos de dieciocho años.

- No se requiere experiencia previa para un `coordinador de proyecto`. Sin embargo, un `director de proyecto` debe tener al menos tres años de experiencia y un `director senior` debe tener al menos cinco años de experiencia.

Entrada

- La entrada es edad, age, y años de experiencia, years of experience.
  Salida
- El resultado es el título del puesto senior para el que es elegible el candidato.

Realice

- Pseudo codigo
- Analizar entradas y salidas
- Codigo en c++