Soluciones OIFem II Nivel 1

Entrenamiento 1

Hola a todos!

Teoría

- Salida de datos (cout)

Solución

Le decimos al usuario el mensaje a través de `cout`, como vimos en clase.

Código

```cpp
#include <iostream>

using namespace std;

int main() {
    cout << "Hola a todos!\n";
    return 0;
}
```

Sum of two integer numbers

Teoría

- Entrada y salida de datos
- Variables
- Operadores aritméticos

Solución

Creamos dos variables de tipo `int`, a y b, y guardamos en ellas los números que nos dice el usuario. Le decimos su suma a través de `cout`.

Código

```cpp
#include <iostream>

using namespace std;

int main() {
    int a, b;
    cin >> a >> b;
    cout << a+b << '\n';
    return 0;
}
```

Maximum of two integer numbers

Teoría

- Entrada y salida de datos
- Variables
- `max`

Solución

Creamos dos variables de tipo `int`, a y b, y guardamos en ellas los números que nos dice el usuario. Le decimos el máximo (encontrado usando `max`) a través de `cout`.

Código

```cpp
#include <iostream>

using namespace std;

int main() {
    int a, b;
    cin >> a >> b;
    cout << max(a, b) << '\n';
    return 0;
}
```

Tres palabras

Teoría

- Entrada y salida de datos
- Variables

Solución

Creamos tres `string`s, a, b y c, y guardamos en ellas las palabras que nos dice el usuario. Luego las imprimimos en el orden que nos dice el problema.

Código

```cpp
#include <iostream>

using namespace std;

int main() {
    string a, b, c;
    cin >> a >> b >> c;
    cout << c << ' ' << b << ' ' << a << '\n';
    return 0;
}
```

Sum of three integer numbers

Teoría

- Entrada y salida de datos
- Variables
- Operadores aritméticos

Solución

Creamos tres variables de tipo `int`, a, b y c, y guardamos en ellas los números que nos dice el usuario. Le decimos su suma a través de `cout`.

Código

```cpp
#include <iostream>

using namespace std;

int main() {
    int a, b, c;
    cin >> a >> b >> c;
    cout << a+b+c << '\n';
    return 0;
}
```

Integer division and remainder of two natural numbers

Teoría

- Entrada y salida de datos
- Variables
- Operadores aritméticos

Solución

Creamos dos variables de tipo `int`, a y b, y guardamos en ellas los números que nos dice el usuario. Le decimos a/b y $a\%b$ a través de `cout`.

Código

```cpp
#include <iostream>

using namespace std;

int main() {
    int a, b;
    cin >> a >> b;
    cout << a/b << ' ' << a%b << '\n';
    return 0;
}
```

Integer division and remainder of an integer number by a natural number

Teoría

- Entrada y salida de datos
- Variables
- Operadores aritméticos

Solución

Partiremos este problema en dos: calcular el resto y el cociente.

¿Cómo podemos calcular el resto? Nuestra primera intuición puede ser pensar que el resto es igual a $a\%b$. Y esto no es mala idea y se obtiene la respuesta correcta para $a \geq 0, b > 0$, pero no cuando $a < 0$ o $b < 0$ ya que en C++ cuando uno de los dos enteros es negativo, el operador % nos dice un resto negativo, siendo este igual al resto positivo que buscamos menos b.

Por lo tanto, podemos concluir que si $a \geq 0$ y $b > 0$, el resto será igual a $a\%b$ y, en caso contrario, a $(a\%b) + b$. ¿Y cómo podemos escribir una fórmula que englobe los dos casos? $(a\%b + b)\%b$ es lo que buscamos.

Ahora encontrar el cociente es fácil. Basta con restarle el resto a a y dividir el resultado entre b.

Código

```cpp
#include <iostream>

using namespace std;

int main() {
    long long int a, b;
    cin >> a >> b;
    long long int resto = (a%b + b)%b;
    long long int cociente = (a - resto)/b;
    cout << cociente << ' ' << resto << '\n';
    return 0;
}
```

Rounding

Teoría

- Entrada y salida de datos
- Variables
- Operadores aritméticos
- Operadores de comparación

Solución

Para resolver este problema, hay que encontrar tres números. Iremos uno a uno viendo cómo hacerlo.

El *floor* de un número x es el mayor número entero y que satisfaga $y \leq x$. Podemos calcular y *borrando* los dígitos detrás del punto decimal de x. Para esto, basta con convertir x a una variable de tipo `int`, ya que esta conversión borra estos dígitos.

El *ceil* de un número x es el menor número entero y que satisfaga $y \geq x$. Por lo tanto, `ceil(x) = floor(x) + 1` si x tiene algún dígito después del punto decimal y ceil(x) = floor(x) = x en caso contrario. x tendrá algún dígito después del punto decimal si `x - floor(x) > 0`. Recordemos que los booleanos pueden usarse como `int`, donde verdadero = 1 y falso = 0. Calcularemos entonces `x + (x - floor(x) > 0)`, que será igual a x si x es un entero e igual a $x + 1$ si hay dígitos tras el punto decimal. Calcular el *ceil* entonces se vuelve sencillo: solo debemos quitar los dígitos tras el punto decimal de este cálculo y ya vimos antes cómo hacer esto.

Por último, nos gustaría redondear x. Si `x - floor(x) < 0.5`, nos interesa x y, en caso contrario, nos interesa $x + 1$. Pues podemos calcular `x - floor(x)` y multiplicarlo por 2. El resultado será menor que 1 si `x - floor(x) < 0.5` e igual o mayor que 1 en caso contrario. Por lo tanto, x redondeado será igual a `floor(x) + floor(2 * (x - floor(x)))`.

Código

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

int main() {
    double x;
    cin >> x;
    int floor_x = (int) x;
    int ceil_x = (int) (x + (x - floor_x > 0));
    int round_x = (int) x + (int) 2*(x - floor_x);
    cout << floor_x << ' ' << ceil_x << ' ' << round_x << '\n';
    return 0;
}
```

Maximum of three integer numbers

Teoría

- Entrada y salida de datos
- Variables
- `max`

Solución

Encontramos el máximo de los tres enteros de la misma forma que hicimos en clase.

Código

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

int main() {
    int a, b, c;
    cin >> a >> b >> c;
    cout << max(a, max(b, c)) << '\n';
    return 0;
}
```

Uppercase and lowercase letters

Teoría

- Entrada y salida de datos
- Variables
- Operadores aritméticos
- Operadores de comparación

Solución

Si un carácter c es una letra minúscula, `c >= 'a'` es verdadero, y falso si c es una letra mayúscula. Podemos multiplicar esto por 2 y por la diferencia entre `'A'` y `'a'` en la tabla ASCII, obteniendo 2 veces esta diferencia si c es minúscula, y 0 si c es mayúscula. Bastará entonces con sumar eso al valor ASCII de c y de restar la diferencia para que así si c es minúscula el resultado sea c más la diferencia y si es mayúscula este sea c menos la diferencia.

Código

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

int main() {
    char c;
    cin >> c;
    cout << (char) (c + 2 * (c >= 'a') * ('A' - 'a') - ('A' - 'a')) << '\n';
    return 0;
}
```