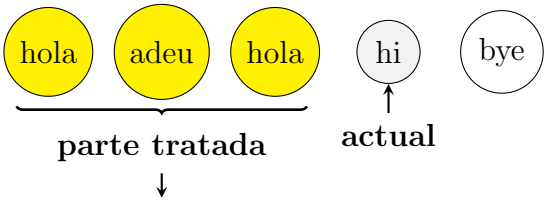
PRO1: Lista P5

Emma Rollón
© Departamento CS, UPC

20 de marzo de 2020

Problema P89872

¿Qué información necesitamos mantener de la parte tratada de la secuencia? Parece claro que necesitamos saber la segunda palabra máxima en orden alfabético (eso es lo que tenemos que escribir). Pero eso sólo lo podremos saber si también almacenamos la palabra máxima.

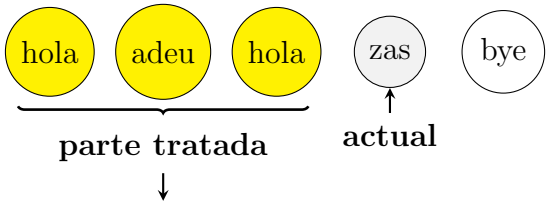


max: palabra máxima alfabéticamente

smax: segunda palabra máxima alfabéticamente

¿Cómo incluímos el elemento actual en la parte tratada? Es decir, cómo tenemos que actualizar *max* y *smax* para que sigan manteniendo su significado e incluya también la palabra actual. Vamos a pensar en los diferentes casos que se pueden dar:

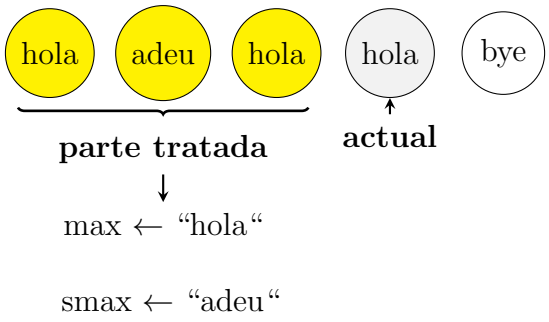
1. El *actual* es mayor que *max*:



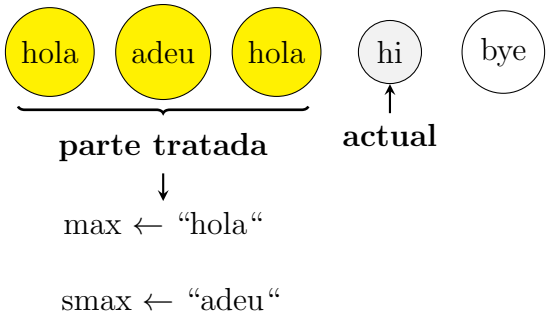
max ← "hola"

smax ← "adeu"

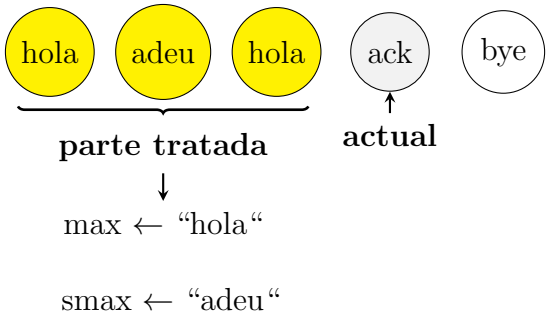
2. El *actual* es igual que *max*:



3. El *actual* es menor que *max* pero mayor que *smax*:



4. El *actual* es menor que *max* y menor que *smax*:

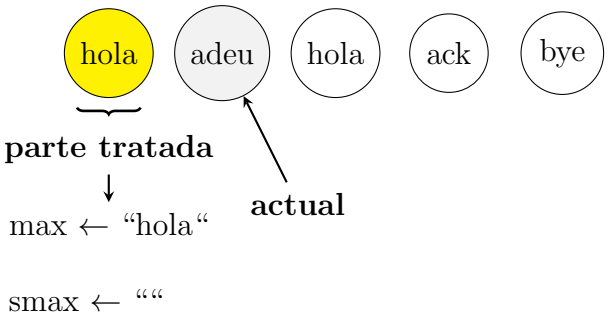


Al responder esta pregunta tendremos parte de las instrucción del cuerpo del bucle. Sólo falta avanzar al siguiente elemento. Fácil: cin >> actual.

¿Cómo sabemos que no hay elementos a tratar? En este caso, en la entrada directamente nos dan la secuencia que queremos tratar. Así es que lo sabremos cuando al leer un nuevo elemento con la instrucción cin ésta nos ”devuelva”false. Por tanto, la instrucción cin >> actual es la condición del bucle.

¿Cómo inicializamos la información a mantener de la parte tratada de la secuencia?
Inicializar *actual* es fácil, la propia condición del bucle lo hace. Tenemos que pensar cómo inializar *max* y *smax* para que mantengan su significado una vez les hemos dado valor. Para hacerlo, nos tenemos que fijar en las condiciones de la entrada (precondición): siempre habrán al menos dos elementos diferentes en la secuencia. Por tanto, fíjate que la secuencia no será vacía (ésa no es una entrada válida). Que la secuencia no sea vacía nos permite leer un elemento sin comprobar si la lectura ha sido correcta (claro, seguro que será correcta, al menos hay un elemento en la secuencia).

Una posible opción es la siguiente:



Fíjate que como la parte tratada está formada sólo por la palabra “hola“, su valor máximo es “hola“. El valor de *smax* tiene que ser un valor tal que signifique *segunda palabra máxima alfabéticamente* sea cual sea la primera palabra de la secuencia. Una opción es el string vacío. Además, como nos aseguran que la secuencia tendrá al menos dos palabras diferentes, sabemos que, si el cuerpo del bucle lo hemos pensado bien, siempre se actualizará con la seguna palabra máxima alfabéticamente cuando la parte tratada tenga al menos dos palabras diferentes.

La estructura del código es:

```
int main() {
  string s, max;
  cin >> max;
  string smax = "";
  while (cin >> s) {
      ...
  }
  cout << smax << endl;
}
```

Problema P89078

Este ejercicio es interesante porque lo podemos plantear desde dos puntos de vista:

- La entrada es una secuencia de naturales, que contiene al menos un número par, y tenemos que decir cuál es su posición. Desde esta perspectiva, tenemos que plantear una búsqueda.
- Como nos aseguran que siempre existe un número par en la secuencia, y tenemos que decir cuál es su posición, lo podemos refrasear a: la entrada es una secuencia de naturales impares, cuyo final está marcado por un número par, y queremos saber cuál es la posición de ese centinela. Desde esta perspectiva, tenemos que plantear un recorrido.

Esto es posible porque estamos seguros de que SIEMPRE existe lo que estamos buscando. Fíjate también que sea cual sea la opción que escojamos, nunca vamos a leer los números de la secuencia que haya después del primer número par.

La estructura de ambas opciones es:

```
int main() {
    int x;
    int pos = ...;
    bool found = false;
    while (not found and cin >> x) {
        ...
    }
    ...
}
```

```
int main() {
    int x;
    cin >> x;
    int pos = ...;
    while (...) {
        ...
        cin >> x;
    }
    ...
}
```

Intenta hacer el razonamiento de ambos puntos de vista y completa los códigos que te damos.

Problema P50095

Vamos a plantear el tratamiento de la secuencia que nos dan:

- Para cada elemento de la secuencia, tenemos que hacer una cierta tarea (escribir su siguiente número primo). Fíjate que no necesitamos mantener ningún tipo de información de la parte ya tratada de la secuencia, sólo el elemento actual.
- Detectaremos el final de la secuencia de naturales primos cuando leamos un elemento que no sea primo: es una lectura con centinela.

Si encapsulamos la tarea de saber si un número es primo o no en una función que devuelva un booleano, la estructura del main es simple. Es muy útil encapsular!

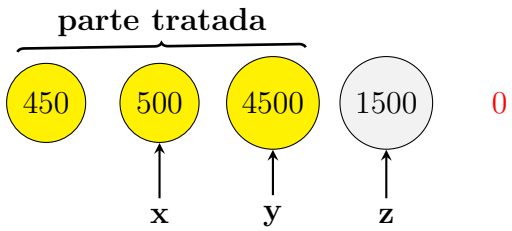
```
// Pre: x >= 0 (natural)
// Post: retorna true si x es primo, false en caso contrario
bool primo(int x) {
    ...
}

int main() {
    int x;      // almacenará el elemento actual
    cin >> x;
    while (primo(x)) {
        // tratar x --> escribir el siguiente primo a x
        ...
        // leer siguiente elemento de la secuencia
        cin >> x;
    }
}
```

Para escribir el siguiente primo a x estaremos en el fondo tratando otra secuencia: la de los números mayores que x. Igual que pasaba en el ejercicio anterior, como ese número primo siempre va a existir, podemos plantearlo como:

- Una búsqueda sobre la secuencia de números mayores que x.
- Un recorrido sobre la secuencia de números mayores que x cuyo final está marcado por un número primo (marca de final o centinela).

Problema P90620

Para solucionar este ejercicio necesitamos mantener, además del elemento actual (que llamaremos z), los dos elementos anteriores a él:



Es muy importante recordar que sólo puedo almacenar valores de la parte ya tratada de la secuencia (es decir, del pasado). El actual (que es el presente) nos sirve para comprobar si

la secuencia se ha acabado o no. De los demás elementos (que son el futuro) no puedo saber nada.

Como estoy buscando si existe o no un pico en la secuencia, la estructura será:

```
int main() {
    ...
    bool existe = false;
    while (not existe and ...) {
        ...
    }
    if (existe) cout << "SI" << endl;
    else cout << "NO" << endl;
}
```

Completa el código siguiendo el razonamiento de tratamiento de secuencias.