Delay a task until a specified time. This function can be used by periodic tasks to ensure a constant execution frequency.

This function differs from vTaskDelay() in one important aspect: vTaskDelay() will cause a task to block for the specified number of ticks from the time vTaskDelay() is called whereas xTaskDelayUntil() will cause a task to block for the specified number of ticks from the time specified in the pxPreviousWakeTime parameter.

Parameters:

- *pxPreviousWakeTime*
  Pointer to a variable that holds the time at which the task was last unblocked. The variable must be initialised with the current time prior to its first use (see the example below). Following this the variable is automatically updated within xTaskDelayUntil().
- *xTimeIncrement*
  The cycle time period. The task will be unblocked at time (*pxPreviousWakeTime + xTimeIncrement). Calling xTaskDelayUntil with the same xTimeIncrement parameter value will cause the task to execute with a fixed interval period.

Returns:

A value which can be used to check whether the task was actually delayed: pdTRUE if the task way delayed and pdFALSE otherwise.

```
// Perform an action every 10 ticks.
void vTaskFunction( void * pvParameters )
{
TickType_t xLastWakeTime;
const TickType_t xFrequency = 10;
BaseType_t xWasDelayed;

    // Initialise the xLastWakeTime variable with the current time.
    xLastWakeTime = xTaskGetTickCount ();
    for( ;; )
    {
        // Wait for the next cycle.
        xWasDelayed = xTaskDelayUntil( &xLastWakeTime, xFrequency
);

        // Perform action here. xWasDelayed value can be used to
determine
        // whether a deadline was missed if the code here took too
long.
    }
}
```