مخبر هياكل البيانات والخوارزميات

الاسم: باسل محمد الحمود

الكود المعدّل:

```
using System;
using System.Collections.Generic;

class Program
{
  // تمثيل المهمة
  class TaskItem
  {
    public string Title { get; set; }
    public int Priority { get; set; } // 1: عالي، 2: متوسط، 3: منخفض
    public DateTime CreatedAt { get; set; }

    public TaskItem(string title, int priority, DateTime createdAt)
    {
      Title = title;
      Priority = priority;
      CreatedAt = createdAt;
    }
  }

  // عقدة لقائمة المهام المنجزة
  class DoneTaskNode
  {
    public TaskItem Task { get; set; }
    public DoneTaskNode Next { get; set; }

    public DoneTaskNode(TaskItem task)
    {
      Task = task;
      Next = null;
    }
  }

  static TaskItem[] tasks = new TaskItem[100];
  static int taskCount = 0;
  static DoneTaskNode doneHead = null;
  static Queue<TaskItem> urgentTasks = new Queue<TaskItem>();

  static void Main()
  {
```

```
    while (true)
    {
        Console.WriteLine("\nقائمة إدارة المهام:");
        Console.WriteLine("1. إضافة مهمة جديدة");
        Console.WriteLine("2. عرض المهام الحالية");
        Console.WriteLine("3. حذف مهمة");
        Console.WriteLine("4. ترتيب حسب الأولوية");
        Console.WriteLine("5. ترتيب حسب التاريخ");
        Console.WriteLine("6. إنهاء مهمة");
        Console.WriteLine("7. عرض المهام المنجزة");
        Console.WriteLine("8. إضافة مهمة عاجلة");
        Console.WriteLine("9. عرض المهام العاجلة");
        Console.WriteLine("0. خروج");

        Console.Write("اختر: ");
        if (int.TryParse(Console.ReadLine(), out int option))
        {
            switch (option)
            {
                case 1: AddTask(); break;
                case 2: ShowTasks(); break;
                case 3: RemoveTask(); break;
                case 4: SortByPriority(); break;
                case 5: SortByDate(); break;
                case 6: FinishTask(); break;
                case 7: ShowDoneTasks(); break;
                case 8: AddUrgent(); break;
                case 9: ShowUrgent(); break;
                case 0: return;
                default: Console.WriteLine("خيار غير صالح."); break;
            }
        }
    }
}

static void AddTask()
{
    Console.Write("اسم المهمة: ");
    string title = Console.ReadLine();
    Console.Write("الأولوية (1-عالي، 2-متوسط، 3-منخفض): ");
    int priority = int.Parse(Console.ReadLine());

    tasks[taskCount++] = new TaskItem(title, priority, DateTime.Now);
    Console.WriteLine("تمت إضافة المهمة.");
}
```

```
  static void ShowTasks()
  {
    Console.WriteLine("\nالمهام النشطة:");
    for (int i = 0; i < taskCount; i++)
      Console.WriteLine($"[{i}] {tasks[i].Title} | أولوية: {tasks[i].Priority} |
{tasks[i].CreatedAt}");
  }

  static void RemoveTask()
  {
    ShowTasks();
    Console.Write("أدخل رقم المهمة للحذف: ");
    int index = int.Parse(Console.ReadLine());

    if (index >= 0 && index < taskCount)
    {
      for (int i = index; i < taskCount - 1; i++)
        tasks[i] = tasks[i + 1];
      tasks[--taskCount] = null;
      Console.WriteLine("تم الحذف.");
    }
    else Console.WriteLine("رقم غير صالح.");
  }

  static void SortByPriority()
  {
    Array.Sort(tasks, 0, taskCount, Comparer<TaskItem>.Create((a, b) =>
a.Priority.CompareTo(b.Priority)));
    Console.WriteLine("تم الترتيب حسب الأولوية.");
  }

  static void SortByDate()
  {
    Array.Sort(tasks, 0, taskCount, Comparer<TaskItem>.Create((a, b) =>
a.CreatedAt.CompareTo(b.CreatedAt)));
    Console.WriteLine("تم الترتيب حسب التاريخ.");
  }

  static void FinishTask()
  {
    ShowTasks();
    Console.Write("أدخل رقم المهمة للإنهاء: ");
    int index = int.Parse(Console.ReadLine());

    if (index >= 0 && index < taskCount)
    {
```

```
            var completed = tasks[index];
            for (int i = index; i < taskCount - 1; i++)
                tasks[i] = tasks[i + 1];
            tasks[--taskCount] = null;

            var node = new DoneTaskNode(completed) { Next = doneHead };
            doneHead = node;

            Console.WriteLine("تم إنهاء المهمة.");
        }
    }

    static void ShowDoneTasks()
    {
        Console.WriteLine("\nالمهام المنجزة:");
        var current = doneHead;
        while (current != null)
        {
            Console.WriteLine($"{current.Task.Title} | أولوية: {current.Task.Priority} |
{current.Task.CreatedAt}");
            current = current.Next;
        }
    }

    static void AddUrgent()
    {
        Console.Write("اسم المهمة العاجلة: ");
        string title = Console.ReadLine();
        Console.Write("الأولوية (1-عالي، 2-متوسط، 3-منخفض): ");
        int priority = int.Parse(Console.ReadLine());

        urgentTasks.Enqueue(new TaskItem(title, priority, DateTime.Now));
        Console.WriteLine("تمت إضافة المهمة العاجلة.");
    }

    static void ShowUrgent()
    {
        Console.WriteLine("\nالمهام العاجلة:");
        foreach (var t in urgentTasks)
            Console.WriteLine($"{t.Title} | أولوية: {t.Priority} | {t.CreatedAt}");
    }
}
```

الشرح المبسّط:

- TaskItem: تمثل مهمة عادية (العنوان – الأولوية – تاريخ الإضافة).
- DoneTaskNode: عقدة لقائمة مرتبطة للمهام المنجزة.
- tasks[]: مصفوفة لحفظ المهام النشطة (حتى 100).
- doneHead: بداية القائمة للمهام المنجزة.
- urgentTasks: قائمة انتظار للمهام العاجلة.

البرنامج يسمح بإضافة، عرض، حذف، ترتيب، وإنهاء المهام بالإضافة إلى دعم المهام العاجلة.