Introduction to Programming EE2310　Homework 8

103061142 楊淳佑

Problem

Assign tasks (with start time and duration) to different resources and let them not to overlap with other tasks:

- Assign resources for tasks generated randomly. If all the existed resources are not available for a task, then create a new resource.
- Check if there is no overlapping in every resources at any time.
- Given a time and show how much resource needed (how many tasks is going on) at the time given.

Solution, Additional Feature, Program Flow & Structure

Classes

- `Task` : Include the time(int), the duration(int) and the resource(int) of a task.
- `Resource` : Include a integer array with 110 elements to store the timeline status.

Functions and flow

- `main()`
  - Generate Task List
    1. Create an empty `Task` vector called `list`.
    2. Generate a `buffer` task with random data inside and push it back to the `list` vector.
    3. Repeat 1.~2. for 100 times.
  - Generate Resource List
    1. Create a `Resource` vector with 1 element inside.
  - Assign Tasks
    1. Call `Assign` function to assign a task.
    2. Print out the assign result.
    3. Repeat 1.~2. For 100 times.
  - Check Resources
    1. Call `ResourceCheck` function.
    2. If return true (means all correct), print out the correct message. If return false, print out the error message.
  - Count Resources Needed

    1. Get a time.
    2. Call `LookUp` function to get the number of resources needed at the time the user inputs. Print out the information at the same time.

- `Assign(vector<Task>& list, vector<Resource>& resource, int i)`
  - Check if any resource is available
    1. Check if a day in a resource that in the task duration is available.
    2. Repeat 1., until every day in the task duration has been checked.
    3. If a resource is available, the program will break from the loop, and the counter `r` will be maintained.
    4. If a resource is not available, repeat 1.~2. to check the next resource.
  - If none of the existed resource is available, create a new one
    1. Add a new resource to the vector by resizing the vector 1 element bigger.
    2. Assign the size of the vector (means the last one) to the counter `r`.
  - Assign the work
    1. Add 1 to every day in the task duration of resource[r-1].
    2. Assign r-1 to the `resource` of the task.
- `ResourceCheck(vector<Resource>& resource)`
    1. Check a day of a resource if it is bigger than 1 (means more than 1 task has been assigned to the same day).
       If there is one day that has been assigned more than one work, return false and end the function.
    2. Repeat 1., until every day of a resource has been checked.
    3. Repeat 1.~2., until every resource has been checked.
    4. Return true and end the function
- `LookUp(vector<Resource>& resource, int d)`
    1. Check the selected day of a resource, add it to `count`.
    2. Repeat 1., until every resource has been checked.
    3. Return `count`.

Output Result

```
 41   8   0
 72   1   0
 80   5   0
 65   9   1
 96   5   0
 49   6   0
 51   8   1
 63   2   0
 66   3   0
 80   7   1
 71   5   2
 64   4   2
 90   3   0
 49   7   2
 23   6   0
 94   7   1
 25   9   1
 51   3   3
 13  10   1
 67   5   3
 19   2   0
 12   4   0
 99   5   2
 92   2   2
 74   9   3
 95   5   3
 39   8   1
 39  10   3
 37   2   0
 57   9   3
 95   6   4
 71   3   4
 86   7   3
 51   3   4
 74   9   4
 75   6   5
```

```
 33  10   2
 29   1   0
 58   2   0
 52   9   5
 35   4   4
 46   5   5
 71   1   5
 17   7   2
 94   9   5
 77  10   2
 83   4   4
 83   9   5
 59   3   1
  2   2   0
 86   6   6
 94   9   6
 80   1   6
 39   7   4
 60   7   4
 72   6   6
 58   3   2
  8  10   3
 12   4   2
 47   3   1
 49   1   3
 71   7   7
 34   8   5
 21   5   3
 92   3   7
 80   1   7
 74   5   0
 28   1   2
100   2   3
 29   8   3
  4   8   1
 79   4   8
```

```
 98  10   7
 91   9   8
 82   9   7
 59   7   6
 34   1   0
 51   9   7
 92   2   4
 77   6   9
 38  10   6
 51   8   8
 35   4   7
 22   2   4
 67   1   4
 90   7   9
 82   1   6
 51   2   6
 78  10  10
 74   5   1
100   5   8
 53  10   9
 57   9  10
 32   9   8
 48   8  10
 92   4  10
  4   4   0
 31   2   0
 69   8   8
 88   5   1
Check result: all
correct.
Enter a time
between 0~110:88
There are 5 tasks
at time 88 .
```