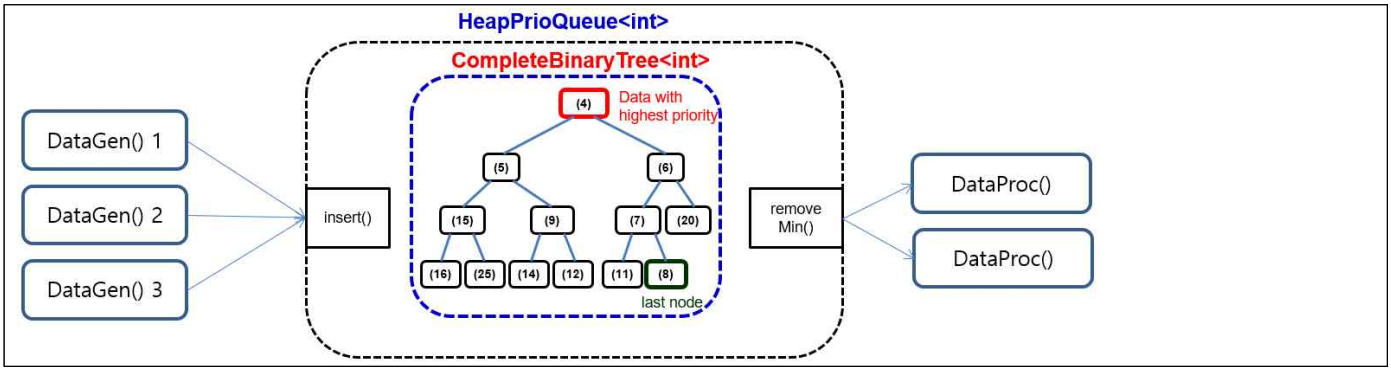
2022-2 객체지향형 프로그래밍과 자료구조 Exam3B

2022. 11. 19.
담당교수 김영탁

3B. class T_Array, class CompleteBinaryTree, class HeapPrioQ (25점)

3B.1 기능 구성



3B.2 class T_Array

- class T_Array는 생성 단계에서 주어지는 capacity 개수의 원소 (템플릿 자료형 E)를 저장할 수 있는 동적 배열을 생성하며 다양한 응용 분야에 사용할 수 있는 템플릿 배열임.
- class T_Array와 관련 멤버함수인 생성자와 소멸자는 직접 구현하며, T_Array.h 헤더파일에 포함시킬 것.

```
/* T_Array.h */
template<typename E>
class T_Array
{
public:
        T_Array(int capacity, string nm); // constructor
        ~T_Array(); // destructor
        int size() { return num_elements; }
        bool empty() { return num_elements == 0; }
        string getName() { return name; }
        E& operator[](int index) { return t_array[index]; }
protected:
        E *t_array;
        int num_elements;
        int capacity;
        string name;
};
```

3B.3 class CompleteBinaryTree

- class CompleteBinaryTree 및 관련 멤버함수들을 구현하여 CompleteBinaryTree.h 파일에 포함시킬 것

```
/* CompleteBinaryTree.h */
. . . . // include necessary header files and definition
template<typename E>
class CompleteBinaryTree : public T_Array<E>
{
public:
        CompleteBinaryTree(int capa, string nm);
        int add_at_end(E& elem);
        E& getEndElement() { return t_array[end]; }
        E& getRootElement() { return t_array[CBT_ROOT]; }
        int getEndIndex() { return end; }
        void removeCBTEnd();
        void fprintCBT(ofstream& fout);
        void fprintCBT_byLevel(ofstream& fout);
protected:
        void _fprintCBT_byLevel(ofstream& fout, int p, int level);
        int parentIndex(int index) { return index / 2; }
        int leftChildIndex(int index) { return index * 2; }
        int rightChildIndex(int index) { return (index * 2 + 1); }
        bool hasLeftChild(int index) { return ((index * 2) <= end); }
        bool hasRightChild(int index) { return ((index * 2 + 1) <= end); }
        int end;
};
```

3B.4 class HeapPrioQueue

```
/* HeapPrioQ.h */
. . . . // include necessary header files and definition
template<typename E>
class HeapPrioQueue : public CompleteBinaryTree<E>
{
public:
        HeapPrioQueue(int capa, string nm);
        ~HeapPrioQueue();
        bool isEmpty() { return size() == 0; }
        bool isFull() { return size() == capacity; }
        int insert(E& elem);
        E* removeHeapMin();
        E* getHeapMin();
        void fprint(ofstream &fout);
        int size() {return end; }
private:
        mutex mtx_heap_prioQ;
};
```

- class HeapPrioQ 및 관련 함수들은 HeapPrioQ.h 파일에 작성할 것

3B.5 Multi_thread.h

```
. . .   // include necessary header files and definitions
typedef struct
{
      mutex   mtxThrdMon;
      mutex   mtx_console;
      int numGenData;
      int numProcData;
      int genData_Array[NUM_DATA];
      int procData_Array[NUM_DATA];
} ThreadMon;

typedef struct
{
      HeapPrioQueue<int>* pPriQ;
      int max_round;
      int targetNumData;
      int* pDataArray;
      ThreadMon* pThrdMon;
} ThreadParam;

void Thread_DataGen(ThreadParam *pParam);
void Thread_DataProc(ThreadParam *pParam);
```

3B.6 SimParam.h

```
#define MAX_ROUNDS 100
#define NUM_DATA_GEN 3
#define NUM_DATA_PROC 2
#define NUM_DATA_PER_GEN 20
#define TOTAL_NUM_DATA   (NUM_DATA_PER_GEN * NUM_DATA_GEN)
```

3B.7 Thread_DataGenenerator.cpp

```
/* Thread_DataGenenerator.cpp */
#include <Windows.h>
#include "HeapPrioQ.h"
#include "Multi_Thread.h"

void Thread_DataGen(ThreadParam* pParam)
{
      . . .  // 필요한 지역변수 선언 및 전달된 인수로부터 해당 값 설정
      pParam->pThrdMon->mtx_console.lock();
      cout << "Thread_DataGen[" << myAddr << "] is activated now ..." << endl;
      pParam->pThrdMon->mtx_console.unlock();

      for (int round = 0; round < max_round; round++)
      {
            if (genDataCount >= targetDataGen)
            {
                  if (*pThrdMon->pFlagThreadTerminate == TERMINATE)
                        break;
                  else {
                        sleep_for(std::chrono::milliseconds(500));
                        continue;
                  }
            }
            genData = TOTAL_NUM_DATA - (round * NUM_DATA_GEN + myAddr + 1);
            while (pPriQ->insert(genData) == NULL)
```

```
            {
                sleep_for(std::chrono::milliseconds(10));
            }

            pParam->pThrdMon->mtx_thrd_mon.lock();
            pThrdMon->genDataArray[myAddr][pThrdMon->numGenData[myAddr]] = genData;
            pThrdMon->totalNumGenData++;
            pThrdMon->numGenData[myAddr]++;
            genDataCount++;
            pParam->pThrdMon->mtx_thrd_mon.unlock();
            sleep_for(std::chrono::milliseconds(10 + rand() % 10));
        }
    }
```

3B.8 Thread_DataProcessor.cpp

```
/* Thread_DataProcessor.cpp */
#include <Windows.h>
#include <mutex>
#include "Multi_Thread.h"
#include "HeapPrioQ.h"
#define THREAD_EXIT_CODE 0

void Thread_DataProc(ThreadParam* pParam)
{
    . . .  // 필요한 지역변수 선언 및 전달된 인수로부터 해당 값 설정

    pParam->pThrdMon->mtx_console.lock();
    cout << "Thread_DataProc[" << myAddr << "] is activated now ..." << endl;
    pParam->pThrdMon->mtx_console.unlock();
    for (int round = 0; round < maxRound; round++)
    {
        if (*pThrdMon->pFlagThreadTerminate == TERMINATE)
            break;
        if (!pPriQ->isEmpty())
        {
            pData_proc = pPriQ->removeHeapMin();
            if (pData_proc == NULL)
            {
                pParam->pThrdMon->mtx_console.lock();
                //cout << "Thread_DataProc:: removeHeapMin () ==> HeapPrioQ is Empty Now" << endl;
                pParam->pThrdMon->mtx_console.unlock();
            }
            else
            {
                pParam->pThrdMon->mtx_console.lock();
                //cout << "Thread_DataProc:: removeHeapMin (" << setw(3) << *pData_proc << ")" << endl;
                pParam->pThrdMon->mtx_console.unlock();
                pThrdMon->mtx_thrd_mon.lock();
                pThrdMon->procDataArray[myAddr][pThrdMon->numProcData[myAddr]] = *pData_proc;
                pThrdMon->totalNumProcData++;
                pThrdMon->numProcData[myAddr]++;
                pThrdMon->mtx_thrd_mon.unlock();
            }
        } // end if
        sleep_for(std::chrono::milliseconds(10 + rand() % 10));
    } // end for
}
```

3B.9 main() 함수

```
/* main() for Heap Priority Queue with Multi-thread */
. . . . // 필요한 헤더파일 추가

#define INITIAL_CBT_CAPA 100

void   main()
{
    . . . .  // 지역변수 설정, 출력 파일 준비
    thread thrd_dataProc[NUM_DATA_PROC];
    thread thrd_dataGen[NUM_DATA_GEN];
    HeapPrioQueue<int> HeapPriQ_int(INITIAL_CBT_CAPA, string("Heap_Priority_Queue_Int"));

    for (int p = 0; p < NUM_DATA_PROC; p++)
    {
        for (int i = 0; i < TOTAL_NUM_DATA; i++)
            thrdMon.procDataArray[p][i] = -1;
        thrdMon.numProcData[p] = 0;
        thrdParam_Proc[p].myAddr = p;
        thrdParam_Proc[p].pPriQ = &HeapPriQ_int;
        thrdParam_Proc[p].targetGenData = NUM_DATA_PER_GEN;
        thrdParam_Proc[p].maxRound = MAX_ROUNDS;
```

```
        thrdParam_Proc[p].pThrdMon = &thrdMon;
        thrd_dataProc[p] = thread(Thread_DataProc, &thrdParam_Proc[p]);
    }

    for (int g = 0; g < NUM_DATA_GEN; g++)
    {
        for (int i = 0; i < TOTAL_NUM_DATA; i++)
            thrdMon.genDataArray[g][i] = -1;
        thrdMon.numGenData[g] = 0;
        //thrdParam_Gen[g].dataList = dataList;
        thrdParam_Gen[g].myAddr = g;
        thrdParam_Gen[g].pPriQ = &HeapPriQ_int;
        thrdParam_Gen[g].targetGenData = NUM_DATA_PER_GEN;
        thrdParam_Gen[g].maxRound = MAX_ROUNDS;
        thrdParam_Gen[g].pThrdMon = &thrdMon;
        thrd_dataGen[g] = thread(Thread_DataGen, &thrdParam_Gen[g]);
    }

    thrdMon.mtx_console.lock();
    cout << "Testing " << HeapPriQ_int.getName() << "with " << NUM_DATA_GEN << " data generators and ";
    cout << NUM_DATA_PROC << " data processors" << endl;
    thrdMon.mtx_console.unlock();
    for (int round = 0; round < MAX_ROUNDS; round++)
    {
        thrdMon.mtx_console.lock();
        cout << "Round (" << setw(3) << round << ") : totalDataGenerated = " << setw(3) << thrdMon.totalNumGenData;
        cout << ", totalDataProcessed = " << setw(3) << thrdMon.totalNumProcData << endl;
        for (int g = 0; g < NUM_DATA_GEN; g++)
        {
            //cout << "thrdDataGen[" << g << "] generated " << thrdMon.numGenData[g] << " data" << endl;
        }
        for (int p = 0; p < NUM_DATA_PROC; p++)
        {
            //cout << "thrdDataProc[" << p << "] processed " << thrdMon.numProcData[p] << " data" << endl;
        }
        thrdMon.mtx_console.unlock();

        if (thrdMon.totalNumProcData >= TOTAL_NUM_DATA)
        {
            thrd_flag = TERMINATE;
            break;
        }

        Sleep(100);
    }
    for (int g = 0; g < NUM_DATA_GEN; g++)
    {
        thrd_dataGen[g].join();
    }
    for (int p = 0; p < NUM_DATA_PROC; p++)
    {
        thrd_dataProc[p].join();
    }

    int count = 0;
    for (int g = 0; g < NUM_DATA_GEN; g++)
    {
        cout << "Thread_Gen[" << g << "] generated " << thrdMon.numGenData[g] << " data :" << endl;
        count = 0;
        for (int i = 0; i < thrdMon.numGenData[g]; i++)
        {
            cout << setw(5) << thrdMon.genDataArray[g][i];
            count++;
            if ((count % 20) == 0)
                cout << endl;
        }
        cout << endl;
    }
    for (int p = 0; p < NUM_DATA_PROC; p++)
    {
        cout << "Thread_Proc[" << p << "] processed " << thrdMon.numProcData[p] << " data :" << endl;
        count = 0;
        for (int i = 0; i < thrdMon.numProcData[p]; i++)
        {
            cout << setw(5) << thrdMon.procDataArray[p][i];
            count++;
            if ((count % 20) == 0)
                cout << endl;
        }
        cout << endl;
    }
    return 0;
} // end main();
```

3B.10 class HeapPrioQ 기능 시험 결과

```
Testing Heap_Priority_Queue_Intwith 3 data generators and 2 data processors
Round (  0) : totalDataGenerated =   0, totalDataProcessed =   0
Thread_DataProc[1] is activated now ...
Thread_DataProc[0] is activated now ...
Thread_DataGen[0] is activated now ...
Thread_DataGen[1] is activated now ...
Thread_DataGen[2] is activated now ...
Round (  1) : totalDataGenerated =  18, totalDataProcessed =   8
Round (  2) : totalDataGenerated =  33, totalDataProcessed =  16
Round (  3) : totalDataGenerated =  43, totalDataProcessed =  25
Round (  4) : totalDataGenerated =  60, totalDataProcessed =  36
Round (  5) : totalDataGenerated =  60, totalDataProcessed =  46
Round (  6) : totalDataGenerated =  60, totalDataProcessed =  58
Round (  7) : totalDataGenerated =  60, totalDataProcessed =  60
Thread_Gen[0] generated 20 data :
   59   56   53   50   47   44   41   38   35   32   29   26   23   20   17   14   11    8    5    2

Thread_Gen[1] generated 20 data :
   58   55   52   49   46   43   40   37   34   31   28   25   22   19   16   13   10    7    4    1

Thread_Gen[2] generated 20 data :
   57   54   51   48   45   42   39   36   33   30   27   24   21   18   15   12    9    6    3    0

Thread_Proc[0] processed 30 data :
   55   51   52   45   42   40   37   33   27   25   21   18   17    9    6    3    4    2    1   11
   14   23   29   31   35   38   44   49   57   59
Thread_Proc[1] processed 30 data :
   54   53   48   46   43   39   36   34   28   24   22   19   15   12    7    5    8    0   10   13
   16   20   26   30   32   41   47   50   56   58
```

3B.11 결과물 제출

- 바탕화면의 Exam3 폴더에 Exam3B 프로젝트를 생성
- 압축 파일 내에 포함사항 : 작성한 프로젝트, 실행결과 Capture(채점 시 정확한 실행 유무를 확인하기 위함)
- 실행 화면 캡쳐파일은 각 시험 섹션별로 프로젝트 폴더 내에 저장 후 시험 섹션별 폴더별로 압축
- 제출시 .vs 폴더는 삭제 후 문제별 폴더를 압축하여 제출