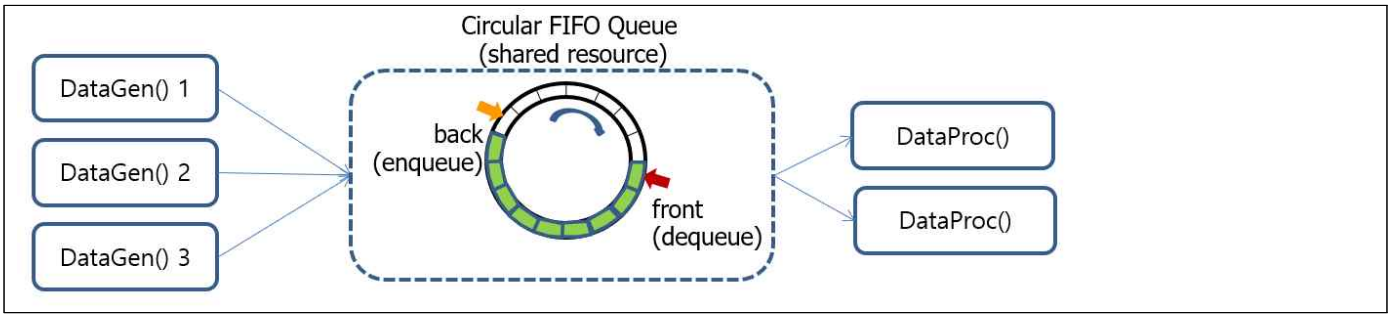
2022-2 객체지향형 프로그래밍과 자료구조 Exam3A

2022. 11. 19.
담당교수 김영탁

3A. class T_Array, TA_CirQ, 및 Multi-thread 응용 프로그램 (25점)

3A.1 기능 구성



3A.2 class T_Array

- 확장형 배열을 위한 template array

```
template<typename E>
class T_Array
{
public:
	T_Array(int capacity, string nm); // constructor
	~T_Array(); // destructor
	int size() { return num_elements; }
	bool isEmpty() { 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;
};
```

3A.3 class TA_CirQ

- 확장형 배열을 기반으로 circular queue를 구현
- circular queue의 공유를 위하여 mutex mtx_cirQ를 사용

```
template<typename E>
class TA_CirQ : public T_Array<E>
{
private:
	int front; // index of queue_front
	int back;    // index of queue_back
	mutex    mtx_cirQ;
public:
	TA_CirQ(int capacity, string nm); // constructor
	~TA_CirQ() {} //destructor
	E* dequeue(); // return the element at front of FIFO queue
	E* enqueue(E& element); // insert an element at the back of the FIFO queue
	bool isEmpty() {    return (this->num_elements == 0); }
	bool isFull() { return (this->num_elements >= this->capacity); }
	int size() { return this->num_elements; }
	void fprint(ostream& fout, int elements_per_line);
};
```

3A.4 Multi_Thread.h - ThreadStatusMonitor, ThreadParam 구조체

```
typedef struct
{
	mutex    mtx_thrd_mon;
	mutex    mtx_console;
	int numGenData[NUM_DATA_GEN];
	int numProcData[NUM_DATA_PROC];
	int totalNumGenData;
	int totalNumProcData;
```

```
    int genDataArray[NUM_DATA_GEN][TOTAL_NUM_DATA]; // used for monitoring only
    int procDataArray[NUM_DATA_PROC][TOTAL_NUM_DATA]; // used for monitoring only
    THREAD_FLAG *pFlagThreadTerminate;
} ThreadMon;

typedef struct
{
    TA_CirQ<int>* pTA_CirQ;
    int myAddr;
    int maxRound;
    int targetGenData;
    ThreadMon* pThrdMon;
}   ThreadParam;
```

3A.5 SimParam.h - Simulation Parameters

```
#define MAX_ROUNDS 100
#define QUEUE_SIZE 10
#define NUM_DATA_GEN 3
#define NUM_DATA_PROC 2
#define NUM_DATA_PER_GEN 50
#define TOTAL_NUM_DATA   (NUM_DATA_PER_GEN * NUM_DATA_GEN)
```

3A.6 Thread_ DataGen.cpp

```
/* Thread_DataGen.cpp */
. . . // 필요한 헤더파일 첨가, 필요한 전처리기 선언
void Thread_DataGen(ThreadParam* pParam)
{
    . . . .  // 필요한 지역 변수 선언
    pParam->pThrdMon->mtx_console.lock();
    cout << "Thread_DataGen[" << myAddr << "] is activated now ..." << endl;
    pParam->pThrdMon->mtx_console.unlock();
    for (int round = 0; round < maxRound; round++)
    {
        if (genDataCount >= targetDataGen)
        {
            if (*pThrdMon->pFlagThreadTerminate == TERMINATE)
                break;
            else {
                sleep_for(std::chrono::milliseconds(500));
                continue;
            }
        }
        genData = round + myAddr*pParam->targetGenData;
        while (pTA_CirQ->enqueue(genData) == NULL)
        {
            sleep_for(std::chrono::milliseconds(100));
        }
        . . . . //스레드 모니터링을 위한 pThrdMon의 procDataArray[myAddr][]에 처리된 데이터 기록
        sleep_for(std::chrono::milliseconds(10 + rand() % 10));
     }
}
```

3A.7 Thread_DataProc.cpp

```
/* Thread_DataProc.cpp */
. . . // 필요한 헤더파일 첨가, 필요한 전처리기 선언
void Thread_DataProc(ThreadParam* pParam)
{
    . . . .  // 필요한 지역 변수 선언

    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 (!pTA_CirQ->isEmpty())
        {
            pDeQ_data = pTA_CirQ->dequeue();
            . . . .  // 스레드 모니터링을 위한 pThrdMon의 procDataArray[myAddr][]에 처리된 데이터 기록
        }
        sleep_for(std::chrono::milliseconds(10 + rand() % 10));
    } // end for
}
```

3A.8 main()

```
/* main.cpp */
. . . // 필요한 헤더파일 첨가, 필요한 전처리기 선언

void main()
{
	ThreadMon thrdMon;
	ThreadParam thrdParam_Gen[NUM_DATA_GEN]; // thrdPram for each data generator
	ThreadParam thrdParam_Proc[NUM_DATA_PROC]; // thrdPram for each data processor
	THREAD_FLAG thrd_flag = RUN;
	thrdMon.pFlagThreadTerminate = &thrd_flag;
	thrdMon.totalNumGenData = thrdMon.totalNumProcData = 0;
	thread thrd_dataProc[NUM_DATA_PROC];
	thread thrd_dataGen[NUM_DATA_GEN];

	TA_CirQ<int> TA_CirQ_int(QUEUE_SIZE, string("TA_CirQ of Integer"));
	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].pTA_CirQ = &TA_CirQ_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].pTA_CirQ = &TA_CirQ_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]);
	}


	cout   << "Testing " << TA_CirQ_int.getName() << "with " << NUM_DATA_GEN << " data generators and ";
	cout   << NUM_DATA_PROC << " data processors" << endl;
	for (int i = 0; i < MAX_ROUNDS; i++)
	{
		thrdMon.mtx_console.lock();
		cout << "Round (" << setw(3) << i << ") : 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;
            . . . . // printout the data generated by each thread data generator
        }
        for (int p = 0; p < NUM_DATA_PROC; p++)
        {
            cout << "Thread_Proc[" << p << "] processed " << thrdMon.numProcData[p] << " data :" << endl;
            . . . . // printout the data processed by each thread data generator
        }
} // end of main()
```

3A.9 기능 시험 결과

```
Testing TA_CirQ of Integerwith 3 data generators and 2 data processors
Thread_DataProc[0] is activated now ...
Round (  0) : totalDataGenerated =   0, totalDataProcessed =   0
Thread_DataProc[1] 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 =  10
Round (  2) : totalDataGenerated =  27, totalDataProcessed =  20
Round (  3) : totalDataGenerated =  38, totalDataProcessed =  30
Round (  4) : totalDataGenerated =  48, totalDataProcessed =  40
Round (  5) : totalDataGenerated =  51, totalDataProcessed =  50
Round (  6) : totalDataGenerated =  66, totalDataProcessed =  58
Round (  7) : totalDataGenerated =  74, totalDataProcessed =  66
Round (  8) : totalDataGenerated =  84, totalDataProcessed =  74
Round (  9) : totalDataGenerated =  86, totalDataProcessed =  84
Round ( 10) : totalDataGenerated =  98, totalDataProcessed =  94
Round ( 11) : totalDataGenerated = 111, totalDataProcessed = 102
Round ( 12) : totalDataGenerated = 120, totalDataProcessed = 112
Round ( 13) : totalDataGenerated = 124, totalDataProcessed = 121
Round ( 14) : totalDataGenerated = 140, totalDataProcessed = 131
Round ( 15) : totalDataGenerated = 145, totalDataProcessed = 141
Round ( 16) : totalDataGenerated = 150, totalDataProcessed = 150
Thread_Gen[0] generated 50 data :
    0   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17  18  19
   20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39
   40  41  42  43  44  45  46  47  48  49
Thread_Gen[1] generated 50 data :
   50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69
   70  71  72  73  74  75  76  77  78  79  80  81  82  83  84  85  86  87  88  89
   90  91  92  93  94  95  96  97  98  99
Thread_Gen[2] generated 50 data :
  100 101 102 103 104 105 106 107 108 109 110 111 112 113 114 115 116 117 118 119
  120 121 122 123 124 125 126 127 128 129 130 131 132 133 134 135 136 137 138 139
  140 141 142 143 144 145 146 147 148 149
Thread_Proc[0] processed 75 data :
    0 100   1 102  53 103 107  54   5   6  56 107   9  10  12  58  14  59 113  16
   61  62  19  63  64  65  21 116  66  67  24 118  69  70 121  72 125  28  73 124
   30  76  31 127  32 128 129  80 130 131 132  37  83  84 134  40  41 136 137  88
   89 140  91 142  92  45 143  94  47 145 146 148  96  98  99
Thread_Proc[1] processed 75 data :
   50  51 101  52   2   3 104  55 105 106  57   7   8  11  13 108 109  60  15 111
   17  18 112 113 114  20 115  22  23 117  68  25 119 120  71 122  27 123  29  74
   75 125 126  77  78  33  79  34  35  81  36  82 133  38  39  85 135  86  87 138
  139  42  90 141  43  44  93 144  46  95  48 147 149  97  49
```

3A.10 결과물 제출

- 바탕화면의 Exam3 폴더에 Exam3A 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