**Homework 13**

|  |
| --- |
| 그림입니다. 원본 그림의 이름: YU_UI_RGB-10.png 원본 그림의 크기: 가로 2256pixel, 세로 3047pixel 프로그램 이름 : Adobe ImageReady |

|  |  |
| --- | --- |
| 과목명 | 프로그래밍 언어 |
| 교수님 | 김영탁 교수님 |
| 이 름 | 김주환 |
| 학 번 | 21812158 |
| 일 자 | 2021.05.31.월 |

1. **Multi-thread 프로그램에서 공유자원에 Critical section (임계구역) 설정이 필요한 이유에 대하여 구체적으로 예를 들어 설명하라.**

|  |
| --- |
| 테이블이(가) 표시된 사진  자동 생성된 설명  은행을 예로 들겠습니다.  입금과 출금이 각각 스레드로 동작하고 계좌 잔고를 공유 자원으로 사용하는 방식으로 은행이 운영되고 있다고 가정합니다. 이 때 임계구역을 설정하지 않는다면 동시에 잔고의 값을 불러옵니다.  만약 1) 출금(80)이 먼저 끝나고 2) 입금(70)이 나중에 이루어진다면(이전 잔고 : 100)  1) 에서는 100 – 80 = 20이 되겠지만  2) 에서는 100 + 70 = 170이 될 것입니다.  원래대로라면 100 – 80 + 70 = 90이 되어야 합니다.  이러한 문제를 사전에 예방하고자 임계구역의 설정이 필요합니다.  이 때 mutex를 이용하여 아래와 같이 코드를 작성하면 문제를 해결할 수 있습니다.  테이블이(가) 표시된 사진  자동 생성된 설명 |

1. **C++ 프로그래밍 환경에서 Thread 생성과 실행을 위한 파라메터/인수를 전달하는 방법에 대하여 구체적으로 예를 들어 설명하고, 일반 함수 호출에서의 인수 전달 방법과의 차이점에 대하여 설명하라.**

|  |
| --- |
| * 스레드 생성 * 헤더 파일 #include <thread>를 통해 헤더 파일을 지정한다. * 스레드 함수의 구현 프로그램에 포함되는 함수 중, 병렬로 실행되어야 하는 함수를 스레드로 지정합니다. 구조체 포인터를 통해 스레드 생성 및 실행에 관련된 정보를 main() 함수로부터 전달받으며, 해당 구조체는 필요에 따라 정의할 수 있습니다. 이 때 스레드는 보통 지정된 회수만큼 실행하거나, 무한 루프로 실행합니다. * 생성 thread 사용할\_스레드\_이름 ( 스레드 함수, 구조체 주소 ); * 스레드에 파라메터 인수 전달 필요에 따라 파라메터 항목들을 포함하는 구조체를 정의합니다. 기본적으로 mutex에 관한 정보, 공유자원에 대한 정보, 파일 입출력에 관련된 정보를 포함합니다.   이 때 구조체는 아래와 같이 정의하고 초기화 합니다.   * 일반 함수 호출에서의 인수 전달 방법과의 차이점   일반적인 함수는 인수를 전달하면 각 함수의 실행마다 인수를 독립적으로 사용합니다. 하지만 각 스레드로 전달된 인수(구조체)는 서로 공유(공유자원)됩니다. 따라서 멀티 스레드는 작업 수행을 병렬로 처리할 수 있습니다. |

1. **Multi-thread의 동작 상태를 monitoring 하여, 주기적으로 상태를 출력하는 함수를 구상하고, 필요한 파라메터 전달, 출력 포맷에 따른 주기적인 출력방법에 대하여 예를 들어 설명하라.**

|  |
| --- |
| * 실행창 초기화 main문의 반복문 내에서 주기적으로 상태를 출력합니다. 이 때 반복문은 다음 현황을 실행창에 출력하기 위해 매 round마다 system(“cls”)를 사용하여 실행창을 초기화 합니다. 이 함수의 헤더파일은 Windows.h입니다. * 구조체, 구조체 변수의 변경 구조체는 생성된 각 스레드 내부에서 동작의 순서를 확인하기 위해 생성하거나 처리한 이벤트를 배열에 순서대로 저장합니다. 구조체의 변수는 생성된 각 스레드 내부에서 각 라운드가 끝날 때 mutex를 사용하여 다른 스레드 중간에 개입하지 못하도록 만들고 변경합니다. * 필요 파라메터   모니터링에는 thrdMon구조체를 사용합니다. 각 스레드에 전달된 thrdParam\_Handlr와 thrdParam\_Gen은 같은 thrdMon 구조체를 공유하고 있습니다. 이 구조체는 해당 라운드에 생성하고 처리한 이벤트의 수, 전체 과정에서 생성하고 처리한 이벤트의 수, 생성하고 처리한 순서를 저장하는 배열을 멤버로 가집니다. 이 때 각 스레드의 실행에서 이 구조체는 공유자원 이므로 이벤트 생성과 처리에 관련된 정보를 받아옵니다.     * 텍스트이(가) 표시된 사진    자동 생성된 설명출력 포멧 생성한 이벤트, 큐 내부 각 요소의 위치, 처리된 이벤트를 출력합니다. 그리고 각 요소와 함께 번호와 우선순위를 출력합니다. 처리된 이벤트의 출력에는 생성되고 처리되기까지 지난 시간을 함께 출력해줍니다. |

1. **동적으로 할당된 배열을 기반으로 구현된 Circular FIFO Queue의 기본 기능인 enCirQ()와 deCirQ() 함수의 기본 구조 및 동작 원리에 대하여 그림으로 나타내어 설명하라. (사용되는 구조체의 모습을 그림으로 표현할 것.)**

|  |
| --- |
| * 큐의 기본 구조 환형 큐는 동적으로 할당된 배열을 기반으로 구현되었습니다. 형태는 원형으로 꼬리를 무는 형식으로 구성되어 있습니다. 하지만 실제로는 배열의 끝부분에 도달하면 배열의 처음을 돌아오는 구조를 사용하여 원형을 유지하고 있습니다. * 함수의 기본 구조 * enQueue()   1. 공간이 남아있는가?  2. 이벤트를 end에 집어넣기  3. 관련 정보 최신화  4. 배열의 끝이라면 다음 위치를 계산하여 이동   * deQueue()   1. 공간이 비어있는가?  2. front의 이벤트 뽑아내기  2. 관련 정보 최신화  3. 배열의 끝이라면 다음 위치를 계산하여 이동  4. 뽑아낸 이벤트 반환   * 기본 동작 - enQueue()  큐에 자리가 없을 경우 탈출  생성한 이벤트를 큐의 end에 입력  배열의 마지막에 도착하면 배열의 처음부터 다시 입력 - deQueue()  큐가 비어 있으면 탈출  front 위치 조정  front의 성분을 반환 - isFull()  성분의 개수가 전체 용량과 같으면 1, 아니면 0 반환 -isEmpty()  성분의 개수가 0이면 1, 아니면 0 반환 |

1. **우선 순위를 고려한 Event 처리를 위하여 사용되는 Priority Queue에서 우선 순위가 높은 event가 우선적으로 처리될 수 있는 구조와 동작원리 (enPriQ()에서의 up-heap bubbling, dePriQ()에서의 down-heap bubbling 포함)에 대하여 상세히 설명하라.**

|  |
| --- |
| * 내부 구조 우선순위 큐는 동적 확장성 배열을 사용한 완전 이진 트리 구조입니다. 공간이 가득 차면 배열의 크기를 2배만큼 확장합니다.        * 동작 원리 * up-heap bubbling 자식 위치는 부모 위치의 2배입니다. 이때 자식의 우선순위가 부모 보다 높으면 스왑하고 위의 동작을 반복합니다. 반대의 경우에는 반복을 탈출합니다. * down-heap bubbling last node를 root로 변경하고 배열의 크기를 한 칸 줄입니다. 이때 현재 위치는 root입니다. 현 위치에서 자식이 어느 쪽에 있는 지, 둘 다 있다면 어느 쪽의 우선순위가 높은 지 판단합니다. 선택된 자식의 우선순위가 부모의 것보다 높다면 스왑합니다. 자식이 없는 경우까지 반복합니다. |