**Artur Grudkowski 283738**

**SOI**

**Lab 4 – Monitory**

1. **Koncepcja**

Projekt realizacji urzędu będzie się opierał o monitory, które będą klasami zawierającymi w sobie kolejki priorytetowe. Klasa monitor będzie same w sobie obsługiwać synchronizację za pomocą mutexów i zmiennych warunkowych, będzie się to odbywało przy dodawaniu i usuwaniu elementów.

class Monitor{

conndition\_variable full, empty;

mutex mtx;

int \*buffor;

int buff\_size, front, head;

public:

Monitor();

~Monitor();

pop();

push();

int get\_size();

};

Jak widać do obsługi zmiennych warunkowych będą korzystał z std::condition\_variable ( z biblioteki condition\_variable) a mutexy będą pochodzić z biblioteki mutex. Warto dodać, że wraz z obsługi zmiennych warunkowych będę używał obiektów unique\_lock, które obsługują mutexy otworzone w danym monitorze. Za obsługę urzędników będą odpowiadały funkcje w pliku głównym. Funkcje te będą postaci void\* aby mogły być odpalone przy użyciu wątków same nazwy wątków będą zmiennymi globalnymi zdefiniowanymi w pliku głównym. Dzięki użyciu wątków nie będzie potrzeby tworzenia pamięci współdzielonej.

1. **Symulacja**

Do obsługi symulacji będzie utworzony dodatkowy mutex. . Sama funkcja korzystając a z terminala będzie w wypisywać zwartość wszystkich kolejek w mocno uproszczonym 2D korzystając z tabeli znaków ASCII.

1. **Pliki**

Planuję stworzyć jedna bibliotekę obsługującą mutexy.

* main.cpp – plik zawierający główną strukturę programu, czyli m.in. funkcje symulujące poszczególnych urzędników do uruchomienia na poszczególnych wątkach
* monitor.h – plik nagłówkowy zawierający deklaracje funkcji wykonywanych na monitorach
* monitor.cpp – plik zawierający definicję funkcji z pliku monitor.h
* Makefile – plik typu make do kompilacji projektu

4.Pytania

1. Czy można używać mutexów i zmiennych warunkowych poza monitorami?