Билет №2. Бекимова Асылзат МКМ-41.

1. Ағын функцияға аргументтер беру

Windows амалдық жүйесінің ағындары – бұл операциялық жүйеге параллель есептеулерді жүргізуге мүмкіндік беретін төмен деңгейлі механизм. Ағындардың арқасында ОЖ көп бағдарламалы жұмысты – бір уақытта бірнеше бағдарламаны орындауды, сонымен қатар параллель есептеуді – бір бағдарламаның ~~ішінде~~ бірнеше код үзінділерін бір уақытта орындауды үйлестіреді.

„Қарапайым" компьютерлерді алсақ, параллель компьютерлерде де есептің шешімі көптеген жай операциялардың орындалуы нәтижесінде табылады.

Барлық операциялардың санаулы аргументтері болады. Әдетте олардың саны екеуден аспайды. Операция аргументтерінің нақты мәндері ретінде не кіріс мәліметтері, не болмаса өзі орындалған операциялардың нәтижелері алынады. Қай нәтижелер қандай аргументтер болатындығын төмендегі сәйкестік, бағдарламаны әзірлеуші арқылы белгіленеді. Осыдан түсінетініміз: ∀ операция – аргументтер тұтынушыға өзіне сол аргументтерді жіберуші – бүкіл операциялар орындалмайынша іске қосыла алмайды. Осылайша, көптеген операциялар үшін бағдарламаны әзірлеуші қарапайым реттілікті нақты не нақты емес түрде белгілейді.

2. std :: lock_guard қолдану мысалдары

Ашық блокировка және одан босату қателіктері жиі мүмкін, мысалы, сіз ашқан блокировкадан шығуды ұмытып кеткен кезде. Және блокировканың дұрыс емес тәртібі белсенділігі бірі deadlock-ты тудырады. std бұл мәселені шешу үшін бірнеше кластар мен әдістерді ұсынады

lock_guard: нысан жасалған кезде ол мьютексті алуға тырысады, ал нысан жойылған кезде мьютексті автоматты түрде босатады.

```
template <typename T>
class container
{
   std::recursive_mutex_lock;
   std::vector <T> _elements;
 public:
   void add (T element)
  { ~~void add~~
    std:: lock_guard <std::recursive_mutex> locker(_lock);
   _elements.push_back (element);
  }
  void addrange (int num, ...)
  { va_list arguments;
    va_start (arguements, num);
    for (int i=0 ; i< num ; i++)
       { std :: lock_guard <std :: recursive_mutex> locker (_lock);
       add (va_arg (arguments, T));
       }
     va_end (arguments);
   }
  void dump ()
   { std :: lock_guard <std :: recursive_mutex> locker(_lock);
   for (auto e : _elements)
     std :: cout << e << std :: endl; }};
```