Imię i nazwisko: ..... **T2** czas: 60 min

Przyjąć, że udostępniona jest przestrzeń nazw std, std::placeholders i boost

Zadanie 1 (6pkt)

Poniżej przedstawiono klasy , które przetwarzają zadania (tutaj obiekty typu `int`) w różnych wątkach. Dla przedstawionego poniżej przykładu nie wszystkie wątki się kończą. Popraw kod.

```
void zad1() {
   MsgQueue queue;
   Consumer c1(queue), c2(queue);
   vector<int> tasks = {1,2,3,4,5,6};
   Producer p(queue, tasks);
   boost::thread thrd1( ref(c1) );
   boost::thread thrd2( ref(c2) );
   boost::thread thrd3( ref(p) );
   thrd1.join(); thrd2.join();
   thrd3.join();
}
class MsgQueue {
public:
   static const int END_ID = 0;
   boost::optional<int> get() {
      mutex::scoped_lock scoped(m_);
      boost::optional<int> out;
      if( ! q_.empty() ) {
         out = q_.front();
         q_.pop();
      }
      return out;
   }
   void put(int i) {
      mutex::scoped_lock scoped(m_);
      q_.push(i);
   }
private:
   boost::mutex m_;
   queue<int> q_;
};
```

```
class Producer {
public:
  Producer(MsgQueue& q, vector<int> in) : q_(q), tasks_(in) {}
  void operator()() {
    for_each(tasks_.begin(),tasks_.end(),[&](int i){q_.put(i);});
    q_.put(MsgQueue::END_ID);
  }
private:
  MsgQueue& q_;
  vector<int> tasks_;
};
class Consumer {
public:
   Consumer(MsgQueue& q) : q_(q), finish_(false) {}
   void operator()() {
      while(!finish_) {
         boost::optional<int> task = q_.get();
         if( ! task)
            this_thread::sleep(posix_time::millisec(1000));
         else if( *task == MsgQueue::END_ID)
            finish_ = true;
         else {
            //tutaj obsługa zadania
         }
      }
   }
private:
   MsgQueue& q_;
   bool finish_;
};
```

Zadanie 2 (6pkt)

Klasa `Vector` przechowuje obiekty w specjalnym obszarze pamięci, w przedstawionym przykładzie obszar ten jest reprezentowany przez wskaźnik `buffer_`. Popraw metodę `~Vector`, aby nie wołać destruktorów dla obiektów typu `T`, gdy `T` ma trywialny destruktor. Wtedy można zrezygnować z pętli, wystarczy usunięcie bufora.

Jeżeli typ ma trywialny destruktor to znaczy, że destruktor nie ma żadnego efektu i wołanie destruktora można pominąć. Jeżeli `T` ma trywialny destruktor to `boost::has_trivial_destructor<T>::type` jest typu `boost::true_type`, w przeciwnym wypadku jest typu `boost::false_type`. Jeżeli `T` ma trywialny destruktor to `boost::has_trivial_destructor<T>::value` jest typu `bool` i ma wartość `true`, w przeciwnym wypadku ma wartość `false`. Podobną rolę ma szablon `std::is_trivially_destructible`.

```
template<typename T> class Vector {
public:
  typedef unsigned char BUF_ELEMENT; static const int BUFFER_SIZE = 100;
  Vector() : buffer_(new BUF_ELEMENT[BUFFER_SIZE]), size_(0) { }
  void add(const T& t) {
    new(getRawPointer(size_)) T(t); ++size_;
  }
  ~Vector() {
    for(int i=0;i<size_;++i)
      getPointer(i)->~T(); //woła destruktor
    delete [] buffer_;
  }
private:
  BUF_ELEMENT* getRawPointer(int idx) { return buffer_ + idx * sizeof(T); }
  T* getPointer(int idx) { return reinterpret_cast<T*>(getRawPointer(idx)); }
private:
  BUF_ELEMENT* buffer_;
  int size_;
};
```

Zadanie 3 (3pkt)

Napis NAME zawiera Twoje nazwisko zapisane za pomocą wielkich liter ASCII (zamiast 'Ą' jest 'A'), np. `WROZKA` dla nazwiska Wróżka.

```
const string NAME = "␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣;
```

Podaj napis, który zostanie wydrukowany przez funkcję `zad3` [ ]

```
class Vis : public boost::static_visitor<void> {
public:
   Vis() {}
   void operator()(const int& i){ str_ += lexical_cast<string>(i+1); }
   void operator()(const std::string& s) { str_ += s; }
   string str_;
};
void zad3() {
   typedef boost::variant<int,string> V;
   std::vector<V> v;
   v.push_back(NAME.size()); v.push_back(NAME);
   v.push_back(boost::lexical_cast<string>(v.front())); v.push_back(v.front());
   Vis vis;
   for_each(v.begin(), v.end(), [&](const V& v){apply_visitor(vis, v);});
   cout << vis.str_ << endl;
```

Zadanie 4 (4pkt)

Uzupełnij kod funkcji `getNames`. Funkcja dostarcza nazwiska osób, które mają dochód nie mniejszy niż `min_salary`. Dla pokazanego poniżej testu funkcja ta zwraca wektor zawierający napisy `"B", "D"`. Użyj algorytmu z biblioteki standardowej.

```
typedef std::pair<string, int> Person;//osoba; nazwisko i dochód
void zad4() {
 vector<Person> v = {Person("A",1700), Person("B",2100), Person("C",1800), Person("D",2200)};
 vector<string> names_over_2000 = getNames(v, 2000); //zwraca 2 elementy: "B" i "D"
}
```

```
vector<string> getNames(const vector<Person>& persons, int min_salary) {
```

Zadanie 5 (3pkt)

`NAME` zdefiniowano w zad. 3. Podaj napis drukowany przez funkcję `zad5` [ ]

```
int sum(int& s, int a)
{
  s += a + 1;
  return s;
}
```

```
void zad5() {
   vector<int> v;
   v.push_back(1);
   v.push_back(NAME.size());
   int s = 0;
   for_each( v.begin(), v.end(), bind(sum, ref(s), bind(sum, ref(s), _1) ) );
   cout << s << endl;
}
```

Pytanie 1 (1pkt)

Do czego wykorzystujemy serializację obiektów (przekształcenie do postaci szeregowej)?

Pytanie 2 (1pkt)

Zaproponuj zagadnienie, które Twoim zdaniem warto byłoby omówić na przedmiocie ZPR

Pytanie 3 (1pkt)

Ile godzin w semestrze poświęciłeś na przedmiot ZPR (wykład, projekt, kolokwia, nauka własna i inne) [ ]

Notatki / uwagi do prowadzącego