Task 1

1. Construct a parameterizable array of **pointers** (*TArray*). Implement a constructor and the destructor (the destructor has to delete all the pointers as well as all the objects pointed to).

2. Construct a nested class *iterator* which implements an iterator. The class must include a reference to *TArray* and an index (to mark the current position within the array). Implement the constructor, the operator ++ *postfix* (to iterate to the next element within the array), the operator* (to obtain a value of the element of the array pointed to by the iterator), the operators ==, != (to compare two iterators) and << (for displaying the element of the array pointed to by the iterator).

3. Add to the class *TArray* the functions *begin* and *end* that generate two iterators, the first of which points to the beginning of the array and the second one points to the end of the array (actually, one position behind the end).

4. Implement the functions *Insert* and *Max* (to find out the “maximum” element in the array).

5. The classes should have the following interfaces:

```
template <class T> class TArray
{     T** m_array;
      unsigned m_nSize;

public:
      class iterator;
      friend class iterator;
      class iterator
      {     const TArray& t;
            unsigned index;
      public:
            iterator(const TArray<T>& a, unsigned s = 0);
            const iterator operator++ (int);
            const T* const operator* () const;

            bool operator== (const iterator& r) const;
            bool operator!= (const iterator& r) const;

            friend std::ostream& operator << (std::ostream& os,
                        const iterator& it) { return os << **it; }
      };

      TArray();
      virtual ~TArray();
      void Insert (T* t);
      const T* const Max() const;
      const iterator begin() const;
      const iterator end() const;
};
```

6. Implement a global function `template <class IT> void Visualize(const std::string& text, IT start, const IT& end)`. This function receives a text for displaying and two iterators and should visualize all the elements pointed to by the iterators.

7. Test you code with the following *main* function:

```
int main()
{
    using std::endl;    using std::cout;

    TArray<int> ar_int;
    for (unsigned i = 0; i < 10; i++)
        ar_int.Insert(new int(i));

    Visualize("Template of integers: ", ar_int.begin(), ar_int.end());
    cout << "Maximum: " << *ar_int.Max() << endl << endl;

    TArray<double> ar_dob;
    for (unsigned i = 0; i < 10; i++)
        ar_dob.Insert(new double (i*2.3));

    Visualize("Template of doubles: ", ar_dob.begin(), ar_dob.end());
    cout << "Maximum: " << *ar_dob.Max() << endl << endl;

    TArray<CFigure> ar_fig;
    ar_fig.Insert(new CCircle (4));
    ar_fig.Insert(new CSquare (6));
    ar_fig.Insert(new CRectangle (2,7));

    Visualize("Template of figures: ", ar_fig.begin(), ar_fig.end());
    cout << "Maximum: " << *ar_fig.Max() << endl << endl;

    return 0;
}
```