```cpp
//
// Created by liu on 2017/1/2.
//

#ifndef PROJECT_CONVERTER_H
#define PROJECT_CONVERTER_H

#include "AbstractConverter.h"
#include "vector"
#include "../base/debug.h"
#include "../base/progressBar.h"
using namespace std;

namespace extractor{
  template <class T1, class T2>
  class Converter : public AbstractConverter<vector <T1>, vector <T2>>{
  public:
    vector <T1> * origin_data_ref   ;
    vector <T2> data ;
    base::Debug debug ;
    Converter(): debug ("Converter") {};
    virtual  void  init(vector <T1> & origin_data) {
      origin_data_ref    = & origin_data;
    }

    virtual  void  run() {
      unsigned   num  = origin_data_ref   ->size ();
      debug .debug ("Convert '%s' to '%s' start. total %u %s items",
T1::getClassName  (). c_str (),  T2::getClassName  (). c_str (),  num ,
T1::getClassName  (). c_str ());
      base::ProgressBar bar (num );
      beforeConvert  ();
      for (unsigned  i = 0; i < num ; i++) {
        convert ((* origin_data_ref   )[ i]);
        if (bar .updateLength  (i + 1))
          debug .info ("(%d)%s is converted %d data generate.", i + 1,
bar .getProgress  (i + 1). c_str (),  data .size ());
      }
      afterConvert  ();
      debug .debug ("Convert '%s' to '%s' finish. generate %u %s items",
T1::getClassName  (). c_str (),  T2::getClassName  (). c_str (),  data .size (),
T2::getClassName  (). c_str ());
    }

    virtual  void  beforeConvert  () {} // 开始转换之前
    virtual  void  convert (T1 &) = 0;
    virtual  void  afterConvert  () {} // 结束之后

    virtual  void  insert (T2 item ) {
      data .push_back  (item );
    };
    virtual  vector <T2> & getResult  () {
      return  data ;
    };
    virtual  string  getClassName  () {
      return  "Converter<"   + T1::getClassName  () + ", "  + T2::getClassName  ()
+ ">" ;
    }

    virtual  void  clear () {
      data .clear ();
    }
  };
}

#endif  //PROJECT_CONVERTER_H

```