Зеленковский 4 группа, 1 курс 15 февраля 2021

**Задание 1**. Считать слова из текстового файла и занести в list<string>. Использовать итератор ввода.

**std::istream& operator>>(std::istream& in,std::list<std::string> data\_list){**

**std::istream\_iterator<std::string> input(in);**

**std::string str;**

**while(in>>str){**

**data\_list.push\_back(\*input);**

**input++;//пробел**

**}**

**}**

**Задание 2**. Вывести в стандартный выходной поток содержимое list<string>. Использовать итератор вывода.

**std::ostream& operator<<(std::ostream& out, const std::list<std::string>& data\_list){**

**std::ostream\_iterator<std::string> output(out);**

**std::copy(data\_list.begin(), data\_list.end(),**

**std::ostream\_iterator<std::string>(out, " "));**

**}**

**Задание 3**. Продемонстрировать копирование данных из одного контейнера в другой. Использовать стандартный алгоритм

**void CopyData(std::vector<std::string> vector,const std::list<std::string> data\_){**

**std::copy(data\_.begin(),data\_.end(),std::back\_inserter(vector));**

**}**

**Задание 4**. Продемонстрировать для пользовательского типа данных сортировки данных в контейнере: по умолчанию, с использованием различных механизмов для задания правила сравнения

**Ответ**

**Задание 5**. Для list<string> - поиск элементов списка, которые начинаются на заданную букву. Два варианта: лямбда-выражение и функциональный класс

**1**

**std::list<std::string> FindByLetter(const char& character,const std::list<std::string> data){**

**std::list<std::string> elements;**

**std::copy\_if(data.begin(),data.end(),std::inserter(elements,elements.begin()),[character](std::string str){return str[0]==character;});**

**return elements;**

**}**

**2**

**class Compare{**

**private:**

**char character\_;**

**public:**

**Compare(char character):character\_(character){};**

**bool operator() (const std::string& element){**

**return(element[0]==character\_);**

**}**

**};**

**std::list<std::string> FindByLetterObject(const char& character,std::list<std::string> data){**

**std::list<std::string> elements;**

**Compare compare(character);**

**std::copy\_if(data.begin(),data.end(),std::inserter(elements,elements.begin()),compare);**

**return elements;**

**};**

**Задание 6**. Удаление элементов списка, начинающихся на заданную букву.

**void DeleteByLetter(const char& character,std::list<std::string>& data){**

**auto iterator=std::find\_if(data.begin(),data.end(),**

**[character](std::string str){return str[0]==character;});**

**if(iterator==data.end()){**

**return;**

**}**

**while(iterator!=data.end()){**

**data.erase(iterator);**

**iterator=std::find\_if(iterator++,data.end(),**

**[character](std::string str){return str[0]==character;});**

**}**

**}**

**Задание 7**. Подсчет количества слов в списке, содержащих заданную букву.

**int AmountOfWords(const char& character,const std::list<std::string>& data){**

**int i=0;**

**auto iterator=std::find\_if(data.begin(),data.end(),**

**[character](std::string str){return str[0]==character;});**

**while(iterator!=data.end()){**

**i+=1;**

**auto iterator=std::find\_if(data.begin(),data.end(),**

**[character](std::string str){return str[0]==character;});**

**}**

**return i;**

**}**

**Задание 8**. Вывести в стандартный выходной поток содержимое list<string> в обратном порядке.

**void Reverse(const std::list<std::string>& data,std::string filename){**

**std::ofstream out(filename);**

**std::reverse\_copy(data.begin(),data.end(),std::ostream\_iterator<std::string>(out," "));**

**}**

**Задание 9**. Для заданного текстового файла вычислить множество различных слов и вывести их в алфавитном порядке.

**std::vector<std::string> DifferentWords(std::string filename){**

**std::string word;**

**std::ifstream in(filename);**

**std::set<std::string> dif\_words;**

**while(in>>word){**

**dif\_words.insert(word);**

**}**

**}**

**Задание 10**. Для заданного текстового файла вычислить количество повторений каждого слова, вывести результат.

**std::map<std::string,int> WordFrequency(std::string filename){**

**std::map<std::string,int> frequency;**

**std::ifstream in(filename);**

**std::vector<std::string> words;**

**std::string word;**

**while(!in.eof()){**

**in>>word;**

**words.push\_back(word);**

**}**

**for(std::string str:words){**

**int amount=std::count\_if(words.begin(),words.end(),[str](const std::string& word){return str==word;});**

**frequency.insert(std::make\_pair(str,amount));**

**}**

**return frequency;**

**}**

**Задание 11**. Разработать пользовательский тип данных Последовательная коллекция содержит данные разработанного типа. Выполнить сортировку по одному, двум и трем ключам.

**class MyClass{**

**private:**

**int id;**

**int number1\_;**

**int number2\_;**

**int number3\_;**

**public:**

**MyClass()=default;**

**MyClass(int number1,int number2,int number3):number1\_(number1),number2\_(number2),number3\_(number3){}**

**~MyClass()=default;**

**int GetNumber1() const {return number1\_;}**

**int GetNumber2() const {return number2\_;}**

**int GetNumber3() const {return number3\_;}**

**int GetId() const {return id;}**

**bool operator<(MyClass obj) {**

**return (id < obj.GetId());**

**}**

**};**

**struct SortByOneParameter{**

**bool operator()(const MyClass& obj1,const MyClass& obj2){**

**return obj1.GetNumber1()<obj2.GetNumber1();**

**}**

**};**

**struct SortByTwoParameter{**

**bool operator()(const MyClass& obj1,const MyClass& obj2){**

**if(obj1.GetNumber1()==obj2.GetNumber1()){**

**return obj1.GetNumber2()<obj2.GetNumber2();**

**}**

**return obj1.GetNumber1()<obj2.GetNumber1();**

**}**

**};**

**struct SortByThreeParameter{**

**bool operator()(const MyClass& obj1,const MyClass& obj2){**

**if(obj1.GetNumber1()==obj2.GetNumber1()){**

**if(obj1.GetNumber2()==obj2.GetNumber2()){**

**return obj1.GetNumber3()<obj2.GetNumber3();**

**}**

**return obj1.GetNumber2()<obj2.GetNumber2();**

**}**

**return obj1.GetNumber1()<obj2.GetNumber1();**

**}**

**};**

**std::vector<MyClass>data\_;**

**SortByOneParameter compare1;**

**//?std::sort(data\_.begin,data\_.end(),compare1);**