**НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ**

**МОСКОВСКИЙ ЭНЕРГЕТИЧЕСКИЙ ИНСТИТУТ**

**Лабораторная работа №3**

**Функции и пользовательские типы данных.**

**Выполнил:**

А-02-13

Грязнов С.М.

**Принял:**

Никитин В.В.

Москва, 2015**1. В программе из ЛР № 1 выделить код для преобразования температуры между шкалами в функцию.**

#include "sdt.h"

double convert(double, char, char);

int main()

{ double x; //значение температуры

char scale;

vector<double> temp; //хранение значений температуры

while (cin)

{

cout << "Enter temperature with its scale: ";

cin >>x;

cin >>scale;

temp.push\_back(convert(x,scale,'C'));

temp.push\_back(convert(x,scale,'K'));

temp.push\_back(convert(x,scale,'F'));

}

//вывод на экран

cout <<"C\t K\t F\t\n";

for (int i=0; i<temp.size(); ++i)

{cout <<temp[i] <<"\t";

if ((i+1)%3==0) cout <<endl;}

}

double convert(double temp, char from, char to)

{ //перевод в шкалу С

double tempC;

switch (from)

{case 'C': tempC=temp;

break;

case 'K': tempC=temp-273.15;

break;

case 'F': tempC=5/9.0\*(temp-32);

break;

default: cout <<"Unknow scale.\n";

return 0;}

if (tempC<-273.15)

{ cout <<"Temperature less than absolute zero!\n";

return 0;}

//перевод в нужную шкалу

switch (to)

{case 'C': return tempC;

break;

case 'K': return tempC+273.15;

break;

case 'F': return 1.8\*tempC+32;

break;

default: cout <<"Unknow scale.\n"; }

}

**2. Завести перечисление кодов ошибок, которые могут возникать в функции convert().**

enum error.

{ ErrScale, //неизвестная шкала

ErrTemp, //температура меньше абсолютного нуля};

**3. Реализовать обработку ошибок вовне convert()по кодам возврата.**

#include "sdt.h"

enum error

{ ErrScale, //неизвестная шкала

ErrTemp, //температура меньше абсолютного нуля};

error convert(double, char, char,double&);

int main()

{

double x,result; //значение температуры

char scale;

vector<double> temp; //хранение значений температуры

while (cin)

{

cout << "Enter temperature with its scale: ";

cin >>x;

cin >>scale;

switch (convert(x,scale,'C',result))

{

case (ErrScale): cout <<"Unknow scale.\n";

break;

case (ErrTemp): cout <<"Temperature less than absolute zero!\n";

break;

default:

convert(x,scale,'C',result);

temp.push\_back(result);

convert(x,scale,'K',result);

temp.push\_back(result);

convert(x,scale,'F',result);

temp.push\_back(result);

}

}

cout <<"C\t K\t F\t\n";

for (int i=0; i<temp.size(); i++)

{

printf("%6.2f\t",temp[i]);

//cout <<temp[i] <<"\t";

if ((i+1)%3==0) cout <<endl;

}

}

error convert(double temp, char from, char to, double& result)

{

//перевод в шкалу С

double tempC;

switch (from)

{

case 'C': tempC=temp;

break;

case 'K': tempC=temp-273.15;

break;

case 'F': tempC=5/9.0\*(temp-32);

break;

default: return ErrScale; //ошибка: неизвестная шкала

}

if (tempC<-273.15)

{

return ErrTemp; //ошибка: недопустимая температура

}

//перевод в нужную шкалу

switch (to)

{

case 'C': result=tempC;

break;

case 'K': result=tempC+273.15;

break;

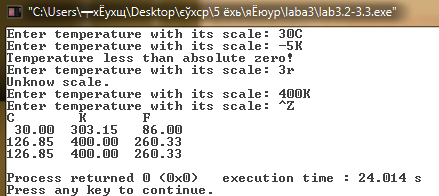
case 'F': result=1.8\*tempC+32;

break;

default: return ErrScale; //ошибка: неизвестная шкала

}

}



**4. Реализовать обработку ошибок вовне convert() через код последней ошибки.**

#include "sdt.h"

int last\_error=0;

int get\_last\_error()

{return last\_error;}

double convert(double, char, char);

int main()

{

double x, tempC; //значение температуры

char scale;

vector<double> temp; //хранение значений температуры

cout << "Enter temperature with its scale: ";

while (cin>>x>>scale)

{

//cout << "Enter temperature with its scale: ";

//cin >>x;

//cin >>scale;

tempC=convert(x,scale,'C');

switch (get\_last\_error())

{

case (1): cout <<"Unknow scale.\n";

break;

case (2): cout <<"Temperature less than absolute zero!\n";

break;

default:

temp.push\_back(tempC);

temp.push\_back(convert(x,scale,'K'));

temp.push\_back(convert(x,scale,'F'));

}

last\_error=0;

cout << "Enter temperature with its scale: ";

}

//вывод на экран

cout <<"C\t K\t F\t\n";

for (int i=0; i<temp.size(); i++)

{

printf("%6.2f\t",temp[i]);

//cout <<temp[i] <<"\t";

if ((i+1)%3==0) cout <<endl;

}

}

double convert(double temp, char from, char to)

{

//перевод в шкалу С

double tempC;

switch (from)

{

case 'C': tempC=temp;

break;

case 'K': tempC=temp-273.15;

break;

case 'F': tempC=5/9.0\*(temp-32);

break;

default: last\_error=1; //ошибка: неизвестная шкала

return 0;

}

if (tempC<-273.15)

{

last\_error=2; //ошибка: недопустимая температура

return 0;

}

//перевод в нужную шкалу

switch (to)

{

case 'C': return tempC;

break;

case 'K': return tempC+273.15;

break;

case 'F': return 1.8\*tempC+32;

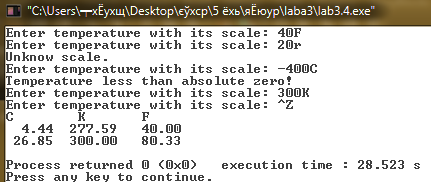
break;

default: last\_error=1; //ошибка: неизвестная шкала

return 0;

}

}



**5. Реализовать обработку ошибок вовне функции через механизм исключительных ситуаций.**

#include "sdt.h"

double convert(double, char, char);

int main()

{ double x, tempC; //значение температуры

char scale;

vector<double> temp; //хранение значений температуры

cout << "Enter temperature with its scale: ";

while (cin>>x>>scale)

{

try

{

temp.push\_back(convert(x,scale,'C'));

temp.push\_back(convert(x,scale,'K'));

temp.push\_back(convert(x,scale,'F'));

}

catch (const invalid\_argument& e)

{

cerr <<e.what();

}

catch (const logic\_error& er)

{

cerr <<er.what();

}

catch (...)

{

cerr <<"Unknow error.\n";

}

cout << "Enter temperature with its scale: ";

}

//вывод на экран

cout <<" C\t K\t F\t\n";

for (int i=0; i<temp.size(); i++)

{

printf("%6.2f\t",temp[i]);

//cout <<temp[i] <<"\t";

if ((i+1)%3==0) cout <<endl;

}

}

double convert(double temp, char from, char to)

{

//перевод в шкалу С

double tempC;

switch (from)

{

case 'C': tempC=temp;

break;

case 'K': tempC=temp-273.15;

break;

case 'F': tempC=5/9.0\*(temp-32);

break;

default: throw invalid\_argument("Unknow scale.\n");

return 0;

}

if (tempC<-273.15)

{

throw logic\_error("Temperature less than absolute zero!\n");

return 0;

}

//перевод в нужную шкалу

switch (to)

{

case 'C': return tempC;

break;

case 'K': return tempC+273.15;

break;

case 'F': return 1.8\*tempC+32;

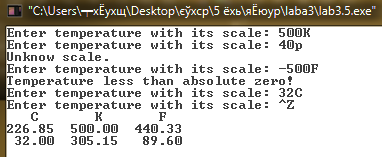
break;

default: throw invalid\_argument("Unknow scale.\n");

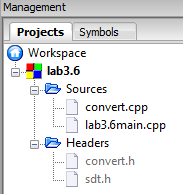
return 0;

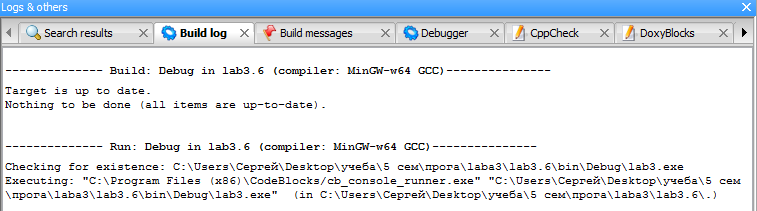
}

}



**6. Создать (изменить) проект так, чтобы в нем были: основной файл исходного кода с функцией main(), заголовочный файл с объявлением функции convert(), файл реализации с определением функции convert().**





**7. Добавить в заголовочный файл определение перечисления Scale, заменить тип параметров to и from функции convert() на Scale.**

**Заголовочный файл convert.h**

enum Scale

{

Celsius,

Kelvin,

Fahrenheit

};

double convert(double t, Scale from, Scale to);

Scale change\_format\_scale(char scale);

**Измененная функция convert(double temp, Scale from, Scale to):**

double convert(double temp, Scale from, Scale to)

{

//перевод в шкалу С

double tempC;

switch (from)

{

case Celsius: tempC=temp;

break;

case Kelvin: tempC=temp-273.15;

break;

case Fahrenheit: tempC=5/9.0\*(temp-32);

break;

default: throw invalid\_argument("Unknow scale.\n");

return 0;

}

if (tempC<-273.15)

{

throw logic\_error("Temperature less than absolute zero!\n");

return 0;

}

//перевод в нужную шкалу

switch (to)

{

case Celsius: return tempC;

break;

case Kelvin: return tempC+273.15;

break;

case Fahrenheit: return 1.8\*tempC+32;

break;

default: throw invalid\_argument("Unknow scale.\n");

return 0;

}

}

**Функция для перевода char в Scale:**

Scale change\_format\_scale(char scale)

{ switch (scale)

{

case 'C': return Celsius;

break;

case 'F': return Fahrenheit;

break;

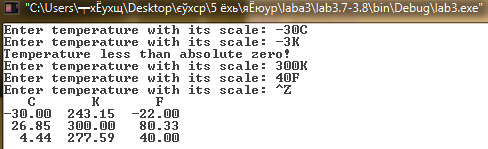
case 'K': return Kelvin;

break;

default: throw invalid\_argument("Unknow scale.\n");

}

}



**8. Возбуждение исключения порождается invalid\_argument(см. пункт 5). Этим решается проблема неопределенного поведения функции при некорректном переводе из одной шкалы в другую.**

**9. Добавить в заголовочный файл определение структуры Temperature, представляющей температуру со шкалой, с конструктором Temperature(double value, char scale). Реализацию конструктора поместить в отдельный файл исходного кода в проекте.**

**Описание структуры (в заголовочном файле):**

struct Temperature

{

Temperature(double value, char scale);

Temperature();

double value;

char scale;

};

**Реализация конструктора (в structure.cpp):**

Temperature::Temperature()

{

this->value=-38.83;

this->scale='C';

}

Temperature::Temperature(double value, char scale)

{

this->value=value;

this->scale=scale;

}

**10. Модернизировать основную программу: использовать Temperature вместо отдельных переменных для значения и шкалы, а также для хранения в векторе.**

**Основная программа:**

int main()

{

Temperature vvod;

vector<Temperature> temp; //хранение значений температуры

cout << "Enter temperature with its scale: ";

while (cin>>vvod.value>>vvod.scale)

{

try

{

temp.push\_back(convert(vvod,Celsius));

temp.push\_back(convert(vvod,Kelvin));

temp.push\_back(convert(vvod,Fahrenheit));

}

catch (const invalid\_argument& e)

{

cerr <<e.what();

}

catch (const logic\_error& er)

{

cerr <<er.what();

}

catch (...)

{

cerr <<"Unknow error.\n";

}

cout << "Enter temperature with its scale: ";

}

//вывод на экран

cout <<" C\t K\t F\t\n";

for (int i=0; i<temp.size(); i++)

{

printf("%6.2f\t",temp[i]);

if ((i+1)%3==0) cout <<endl;

}

}

**Функция convert:**

Temperature convert(Temperature temp, Scale to)

{

//перевод в шкалу С

double tempC;

Temperature answer;

switch (temp.scale)

{

case 'C': tempC=temp.value;

break;

case 'K': tempC=temp.value-273.15;

break;

case 'F': tempC=5/9.0\*(temp.value-32);

break;

default: throw invalid\_argument("Unknow scale.\n");

}

if (tempC<-273.15)

throw logic\_error("Temperature less than absolute zero!\n");

//перевод в нужную шкалу

switch (to)

{

case Celsius: answer.value=tempC;

answer.scale='C';

break;

case Kelvin: answer.value=tempC+273.15;

answer.scale='C';

break;

case Fahrenheit: answer.value=1.8\*tempC+32;

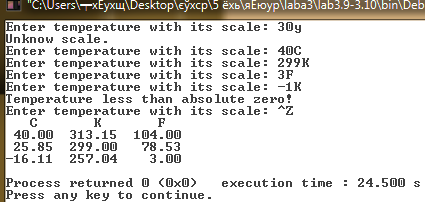
answer.scale='C';

break;

default: throw invalid\_argument("Unknow scale.\n");

}

return answer;



**11. Перегрузить операторы ввода и вывода в поток для перечисления Scale и для структуры Temperature (итого четыре новые функции). Задействовать их в основной программе.**

**В заголовочный файл добавлено:**

//операторы ввода-вывода для структуры

istream& operator>>(istream& input, Temperature& temp);

ostream& operator<<(ostream& output, const Temperature& temp);

//операторы ввода-вывода для перечисления

istream& operator>>(istream& input, Scale& scale);

ostream& operator<<(ostream& output, const Scale& scale);

**Реализация в structure.cpp:**

//операторы ввода-вывода для структуры

istream& operator>>(istream& input, Temperature& temp)

{

input>>temp.value >>temp.scale;

return input;

}

ostream& operator<<(ostream& output, const Temperature& temp)

{

output<<temp.value <<temp.scale;

return output;

}

//операторы ввода-вывода для перечисления

istream& operator>>(istream& input, Scale& scale)

{

char s;

input>>s;

switch(s)

{

case 'C': scale=Celsius;

break;

case 'K': scale=Kelvin;

break;

case 'F': scale=Fahrenheit;

break;

default: input.setstate(ios\_base::failbit); //ошибка

}

return input;

}

ostream& operator<<(ostream& output, const Scale& scale)

{

switch(scale)

{

case Celsius: output<<'C';

break;

case Kelvin: output<<'K';

break;

case Fahrenheit: output<<'F';

break;

}

return output;

}

**12. Перегрузить оператор сравнения для двух температур (<), сложения двух температур (+), деления температуры на число (/) и соответствующие им операторы сокращенного присваивания (+=, /=). Задействовать их для того, чтобы по окончании ввода данных печатать наибольшую, наименьшую и среднюю температуру. В заголовочный файл добавлено:**

//перегрузка операторов

bool operator>(const Temperature left, const Temperature right);

Temperature operator+(const Temperature& left, const Temperature& right);

Temperature operator+=(Temperature& left, const Temperature& right);

Temperature operator/(Temperature& left, const int division);

Temperature operator/=(Temperature& left, const int division);

**Реализация в structure.cpp:**

//перегрузка операторов

**Оператор сравнения “>”:**

bool operator>(const Temperature& left, const Temperature& right)

{

return left.value>right.value;

}

**Оператор сложения “+”:**

Temperature operator+(const Temperature& left, const Temperature& right)

{

return {left.value+right.value,

left.scale};

}

**Оператор сокращенного присваивания “+=”:**

Temperature operator+=(Temperature& left, const Temperature& right)

{

left.value+=right.value;

return left;

}

**Оператор деления “/”:**

Temperature operator/(Temperature& left, const int division)

{

return {left.value/division,

left.scale};

}

**Оператор сокращенного присваивания “+/”:**

Temperature operator/=(Temperature& left, const int division)

{

left.value/=division;

return left;

}

**13. Выделить вычисление наибольшей, наименьшей и средней температур в функцию. Типы и способы передачи параметров выбрать самостоятельно.**

Temperature MaxTemp(vector<Temperature> temp)

{

Temperature MaxT={-373, Celsius};

for (int i=0; i < temp.size(); i++)

{

if (temp[i].value > MaxT.value)

MaxT=temp[i];

}

return MaxT;

}

Temperature MinTemp(vector<Temperature> temp)

{

Temperature MinT={10000, Celsius};

for (int i=0; i < temp.size(); i++)

{

if (temp[i].value < MinT.value)

MinT=temp[i];

}

return MinT;

}

Temperature MeanTemp(vector<Temperature> temp)

{

Temperature MeanT={0, Celsius};

for (int i=0; i < temp.size(); i++)

{

MeanT+=temp[i];

}

MeanT/=temp.size();

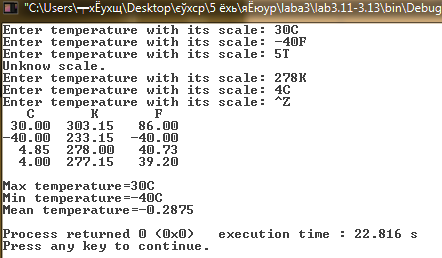
return MeanT;}

**Вывод в основной программе:**

cout <<"/nMax temperature=" <<MaxTemp(tempC)<<endl;

cout <<"Min temperature=" <<MinTemp(tempC)<<endl;

cout <<"Mean temperature=" <<MeanTemp(tempC)<<endl;



**14. Добавить сохранение всех введенных температур в текстовый файл перед окончанием работы программы.**

ofstream out("temperature.txt", ios\_base::trunc); //перед записью файл очищается

for (int i=0; i<temp.size(); i++)

{ out <<temp[i]<<"\t\t";

if ((i+1)%3==0)

{ out <<endl; }

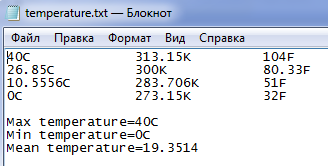
}

out <<"\nMax temperature=" <<MaxTemp(tempC)<<endl;

out <<"Min temperature=" <<MinTemp(tempC)<<endl;

out <<"Mean temperature=" <<MeanTemp(tempC)<<endl;

out.close();



**Код итоговой программы:**

**Основная программа**

#include "sdt.h"

#include "convert.h"

int main()

{

Temperature vvod;

vector<Temperature> temp, tempC; //temp хранение значений температуры

//tempC хранение в одной шкале

cout << "Enter temperature with its scale: ";

while (cin>>vvod)

{

try

{

temp.push\_back(convert(vvod,Celsius));

temp.push\_back(convert(vvod,Kelvin));

temp.push\_back(convert(vvod,Fahrenheit));

tempC.push\_back(convert(vvod,Celsius));

}

catch (const invalid\_argument& e)

{

cerr <<e.what();

}

catch (const logic\_error& er)

{

cerr <<er.what();

}

catch (...)

{

cerr <<"Unknow error.\n";

}

cout << "Enter temperature with its scale: ";

}

//вывод на экран

cout <<" C\t K\t F\t\n";

for (int i=0; i<temp.size(); i++)

{

printf("%6.2f\t",temp[i]);

if ((i+1)%3==0) cout <<endl;

}

cout <<"\nMax temperature=" <<MaxTemp(tempC)<<endl;

cout <<"Min temperature=" <<MinTemp(tempC)<<endl;

cout <<"Mean temperature=" <<MeanTemp(tempC)<<endl;

}

**Заголовочный файл:**

#include "sdt.h"

enum Scale

{

Celsius,

Kelvin,

Fahrenheit

};

struct Temperature

{

Temperature(double value, char scale);

Temperature();

double value;

char scale;

};

Temperature convert(Temperature tepm, Scale to);

//операторы ввода-вывода для структуры

istream& operator>>(istream& input, Temperature& temp);

ostream& operator<<(ostream& output, const Temperature& temp);

//операторы ввода-вывода для перечисления

istream& operator>>(istream& input, Scale& scale);

ostream& operator<<(ostream& output, const Scale& scale);

//перегрузка операторов

bool operator>(const Temperature left, const Temperature right);

Temperature operator+(const Temperature& left, const Temperature& right);

Temperature operator+=(Temperature& left, const Temperature& right);

Temperature operator/(Temperature& left, const int division);

Temperature operator/=(Temperature& left, const int division);

//пункт 13

Temperature MaxTemp(vector<Temperature> temp);

Temperature MeanTemp(vector<Temperature> temp);

Temperature MinTemp(vector<Temperature> temp);

**Файл convert.cpp:**

#include "sdt.h"

#include "convert.h"

Temperature convert(Temperature temp, Scale to)

{

//перевод в шкалу С

double tempC;

Temperature answer;

switch (temp.scale)

{

case 'C': tempC=temp.value;

break;

case 'K': tempC=temp.value-273.15;

break;

case 'F': tempC=5/9.0\*(temp.value-32);

break;

default: throw invalid\_argument("Unknow scale.\n");

}

if (tempC<-273.15)

throw logic\_error("Temperature less than absolute zero!\n");

//перевод в нужную шкалу

switch (to)

{

case Celsius: answer.value=tempC;

answer.scale='C';

break;

case Kelvin: answer.value=tempC+273.15;

answer.scale='K';

break;

case Fahrenheit: answer.value=1.8\*tempC+32;

answer.scale='F';

break;

default: throw invalid\_argument("Unknow scale.\n");

}

return answer;

}

**Файл structure.cpp:**

#include "convert.h"

#include "sdt.h"

Temperature::Temperature()

{

this->value=451;

this->scale='F';

}

Temperature::Temperature(double value, char scale)

{

this->value=value;

this->scale=scale;

}

//операторы ввода-вывода для структуры

istream& operator>>(istream& input, Temperature& temp)

{

input>>temp.value >>temp.scale;

return input;

}

ostream& operator<<(ostream& output, const Temperature& temp)

{

output<<temp.value <<temp.scale;

return output;

}

//операторы ввода-вывода для перечисления

istream& operator>>(istream& input, Scale& scale)

{

char s;

input>>s;

switch(s)

{

case 'C': scale=Celsius;

break;

case 'K': scale=Kelvin;

break;

case 'F': scale=Fahrenheit;

break;

default: input.setstate(ios\_base::failbit); //ошибка

}

return input;

}

ostream& operator<<(ostream& output, const Scale& scale)

{

switch(scale)

{

case Celsius: output<<'C';

break;

case Kelvin: output<<'K';

break;

case Fahrenheit: output<<'F';

break;

}

return output;

}

//перегрузка операторов

bool operator>(const Temperature& left, const Temperature& right)

{

return left.value>right.value;

}

Temperature operator+(const Temperature& left, const Temperature& right)

{

return {left.value+right.value,

left.scale};

}

Temperature operator+=(Temperature& left, const Temperature& right)

{

left.value+=right.value;

return left;

}

Temperature operator/(Temperature& left, const int division)

{

return {left.value/division,

left.scale};

}

Temperature operator/=(Temperature& left, const int division)

{

left.value/=division;

return left;

}

//пункт 13

Temperature MaxTemp(vector<Temperature> temp)

{

Temperature MaxT={-373, Celsius};

for (int i=0; i < temp.size(); i++)

{

if (temp[i].value > MaxT.value)

MaxT=temp[i];

}

return MaxT;

}

Temperature MinTemp(vector<Temperature> temp)

{

Temperature MinT={10000, Celsius};

for (int i=0; i < temp.size(); i++)

{

if (temp[i].value < MinT.value)

MinT=temp[i];

}

return MinT;

}

Temperature MeanTemp(vector<Temperature> temp)

{

Temperature MeanT={0, Celsius};

for (int i=0; i < temp.size(); i++)

{

MeanT+=temp[i];

}

MeanT/=temp.size();

return MeanT;

}