// Imports and namespace

#include <iostream>

#include <string>

#include <typeinfo>

#include <vector>

#include <deque>

using namespace std;

// part: where templates could be helpful

// when you write these, consider just copying the below functions directly

// if you are going too slow

int timesTen(int number);

float timesTen(float number);

double timesTen(double number);

// template<typename NUM>

// NUM timesTen(NUM number);

// also a part: where templates could be helpful

// use this after using them once

// IMPORTANT! DEMONSTRATE how syntax errors are only there when you use them

template<typename NUM>

NUM max(NUM \* array, int size);

void nmain();

void mmain();

int main() {

cout << "5 times 10 = " << timesTen(5) << "." << endl;

cout << "5.5 times 10 = " << timesTen(5.5f) << "." << endl;

cout << "5.9 times 10 = " << timesTen(5.9) << "." << endl;

cout << typeid(timesTen(5)).name() << " " << typeid(timesTen(5.5f)).name() << " " << typeid(timesTen(5.9)).name() << endl;

int foo[] {1, 2, 3, 4, 5, 6, 7, 8, 9, 10};

double foo2[] {1.5, 2.0, 3.5, 4.0, 5.5, 6.0, 7.5, 8.0, 9.5, 0.0};

cout << "The maximum of foo array is: " << max<int>(foo, 10) << endl;

cout << "The maximum of foo2 array is: " << max<double>(foo2, 10) << endl;

nmain(); // demonstrates classes with templates

return 0;

}

/\* TEMPLATE FUNCTIONS \*/

int timesTen(int number) {

return number \* 10;

}

float timesTen(float number) {

return number \* 10;

}

double timesTen(double number) {

return number \* 10;

}

// template<typename NUM>

// NUM timesTen(NUM number) {

// return number \* 10;

// }

template<typename NUM>

NUM max(NUM \* array, int size) {

NUM max = array[0];

for(int i = 0; i < size; i++) {

if(max < array[i]) {

max = array[i];

}

}

return max;

}

/\* CLASSES PERSON, FIGHTER \*/

template<class T>

class Person {

public:

Person(string name, T id) {

this->name = name;

this->id = id;

}

string getName();

T getID();

string getInfo();

protected:

string name;

T id;

};

template<class T>

string Person<T>::getInfo() {

return "Name: " + name + "\nID: " + id;

}

template<class T>

string Person<T>::getName() {

return name;

}

template<class T>

T Person<T>::getID() {

return id;

}

template<class T>

class Fighter : public Person<T> {

public:

Fighter(string name, T id, int wins, int loss) : Person<T>(name, id) {

this->numWins = wins;

this->numLoss = loss;

}

int getWins();

int getLoss();

private:

int numWins;

int numLoss;

string name;

T id;

};

template<class T>

int Fighter<T>::getWins() {

return this->numWins;

}

template<class T>

int Fighter<T>::getLoss() {

return this->numLoss;

}

// template<class T>

// class Celebrity : public Person<T> {

// public:

// Celebrity(string name, T id, int twitterFollowers);

// private:

// int twitterFollowers;

// };

void nmain() {

Person<int> Gerardo("Gerardo", 5472);

cout << "'" << Gerardo.getID() << "' is of type " << typeid(Gerardo.getID()).name() << endl;

Person<string> Josh("Josh", "4X43lSDB3");

cout << "'" << Josh.getID() << "' is of type " << typeid(Josh.getID()).name() << endl;

Fighter<double> Cena("Cena", 2091.5, 13, 3);

cout << "'" << Cena.getID() << "' is of type " << typeid(Cena.getID()).name() << endl;

mmain();

}

/\* TEMPLATE TEMPLATE PARAMETERS \*/

template<template<class, class> class Cont = vector, class T = float, class A>

T getLastElement(const Cont<T, A> &stack) {

return stack.back();

}

//// VS non-template template functions

// template<class Cont, typename T>

// T getLastElement(const Cont &stack) {

// return stack.back();

// }

// call with: getLastElement<vector<int>, int>(vector);

// int getLastElement(vector<int> stack) {

// return stack.back();

// }

// int getLastElement(deque<int> stack) {

// return stack.back();

// }

void mmain() {

vector<float> vect;

vect.push\_back(1.1f);

vect.push\_back(10.2f);

vect.push\_back(100.3f);

vect.push\_back(1000.4f);

cout << "Vector: ";

for (float value : vect) {

cout << value << " ";

}

cout << endl;

cout << "Last Element: " << getLastElement(vect) << endl;

deque<int> deq;

deq.push\_back(1000);

deq.push\_back(100);

deq.push\_back(10);

deq.push\_back(1);

cout << "Deque: ";

for (int value : deq) {

cout << value << " ";

}

cout << endl;

// cout << "Last Element: " << getLastElement<deque, int>(deq) << endl;

}