Program 1:

#include <iostream>

class IncomeTax {

protected:

float income;

public:

IncomeTax(float inc) : income(inc) {}

virtual float calculateTax() const = 0;

};

class Slab1 : public IncomeTax {

public:

Slab1(float inc) : IncomeTax(inc) {}

float calculateTax() const override {

if (income <= 150000)

return 0;

else if (income <= 300000)

return (income - 150000) \* 0.1;

else

return 15000;

}

};

class Slab2 : public IncomeTax {

public:

Slab2(float inc) : IncomeTax(inc) {}

float calculateTax() const override {

if (income <= 300000)

return 0;

else if (income <= 500000)

return (income - 300000) \* 0.2 + 15000;

else

return 50000;

}

};

class Slab3 : public IncomeTax {

public:

Slab3(float inc) : IncomeTax(inc) {}

float calculateTax() const override {

if (income <= 500000)

return 0;

else

return (income - 500000) \* 0.3 + 50000; // Previous slab's maximum tax

}

};

int main() {

float income;

std::cout << "Enter your income: ";

std::cin >> income;

IncomeTax \*tax;

if (income <= 150000)

tax = new Slab1(income);

else if (income <= 300000)

tax = new Slab1(income);

else if (income <= 500000)

tax = new Slab2(income);

else

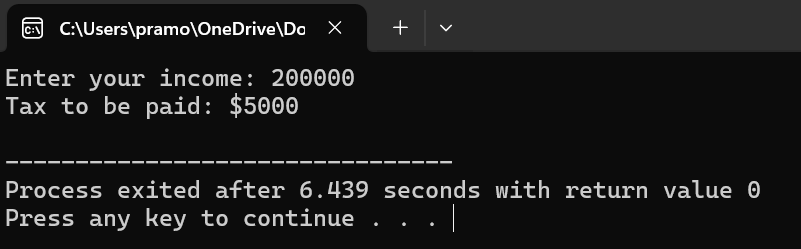
tax = new Slab3(income);

std::cout << "Tax to be paid: $" << tax->calculateTax() << std::endl;

delete tax;

return 0;

}



Program 2:

#include <iostream>

#include <stdexcept>

class Student {

private:

int marks[4];

public:

void inputMarks() {

std::cout << "Enter marks for four subjects: ";

for (int i = 0; i < 4; ++i) {

std::cin >> marks[i];

}

}

int calculateTotal() const {

int total = 0;

for (int i = 0; i < 4; ++i) {

total += marks[i];

}

return total;

}

float calculateAggregate() const {

return calculateTotal() / 4.0f;

}

void displayGrade() const {

int total = calculateTotal();

float aggregate = calculateAggregate();

std::cout << "Total Marks: " << total << std::endl;

std::cout << "Aggregate Marks: " << aggregate << std::endl;

if (total < 50) {

throw std::runtime\_error("Fail");

} else {

std::cout << "Grade: Pass" << std::endl;

}

}

};

int main() {

try {

Student student;

student.inputMarks();

student.displayGrade();

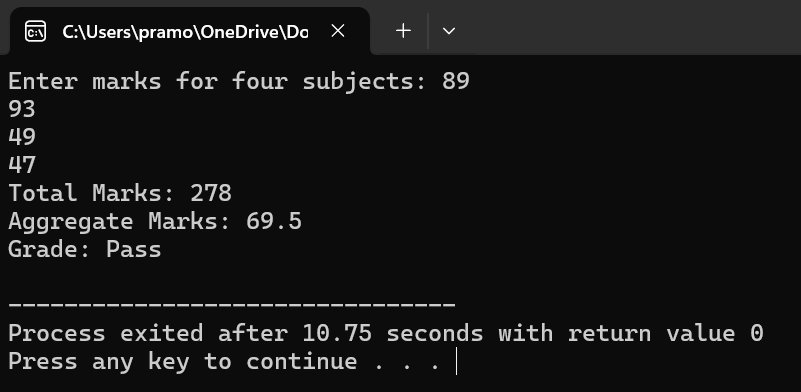
} catch (const std::exception& e) {

std::cerr << "Exception: " << e.what() << std::endl;

}

return 0;

}



Program 3:

#include <iostream>

#include <stdexcept>

int divide(int numerator, int denominator) {

if (denominator == 0) {

throw std::runtime\_error("Division by zero error");

}

return numerator / denominator;

}

int main() {

int numerator, denominator;

std::cout << "Enter numerator: ";

std::cin >> numerator;

std::cout << "Enter denominator: ";

std::cin >> denominator;

try {

int result = divide(numerator, denominator);

std::cout << "Result of division: " << result << std::endl;

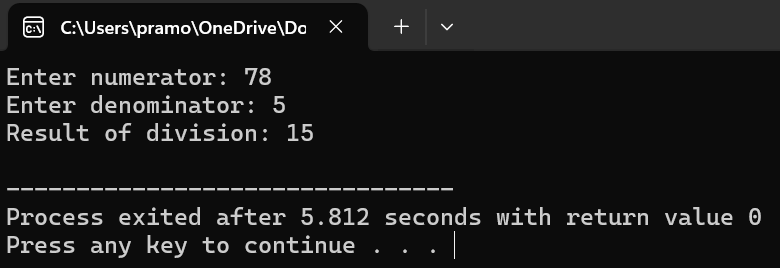
} catch (const std::runtime\_error& e) {

std::cerr << "Exception: " << e.what() << std::endl;

}

return 0;

}



Program 4:

#include <iostream>

#include <stdexcept>

int main() {

int arr[5] = {1, 2, 3, 4, 5};

int index;

std::cout << "Enter an index to access: ";

std::cin >> index;

try {

if (index < 0 || index >= 5) {

throw std::out\_of\_range("Array index out of bounds");

}

std::cout << "Value at index " << index << ": " << arr[index] << std::endl;

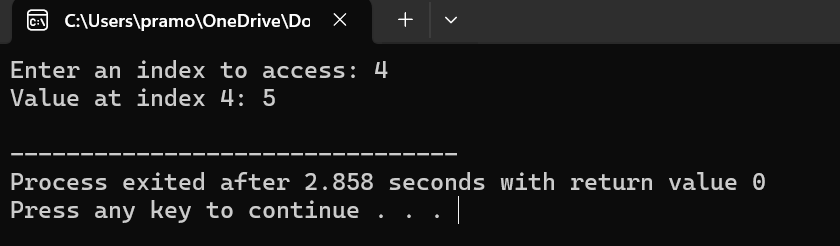
} catch (const std::out\_of\_range& e) {

std::cerr << "Exception: " << e.what() << std::endl;

}

return 0;

}



Program 5:

#include <iostream>

#include <stdexcept>

void checkNumber(int number) {

if (number == 0) {

throw std::invalid\_argument("Number cannot be zero");

} else if (number < 0) {

throw std::out\_of\_range("Number cannot be negative");

} else if (number % 2 != 0) {

throw std::logic\_error("Number must be even");

}

}

int main() {

int num;

std::cout << "Enter a positive even number (non-zero): ";

std::cin >> num;

try {

checkNumber(num);

std::cout << "Number is valid." << std::endl;

} catch (const std::invalid\_argument& e) {

std::cerr << "Invalid argument exception: " << e.what() << std::endl;

} catch (const std::out\_of\_range& e) {

std::cerr << "Out of range exception: " << e.what() << std::endl;

} catch (const std::logic\_error& e) {

std::cerr << "Logic error exception: " << e.what() << std::endl;

} catch (...) {

std::cerr << "Unknown exception occurred." << std::endl;

}

return 0;

}

