**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра САПР**

отчет

**по лабораторной работе №2**

**по дисциплине «****Объектно-ориентированное программирование»**

Тема: «Использование типов-значений»

|  |  |  |
| --- | --- | --- |
| Студенты гр. 1335 |  | Максимов Ю. Е. |

|  |  |  |
| --- | --- | --- |
| Преподаватель: |  | Новакова Н. Е. |

Санкт-Петербург

2024

# **1. Цель работы**

Изучение типов-значений и структур данных на языке C++ с помощью программного продукта компании CLion.

# **2. Анализ задачи**

Необходимо:

1. Написать программу, включающую перечислимый тип enum, включающий различные типы банковских счетов, с выводом результата на экран.
2. Написать программу, включающую структуру, применяемую для представления банковского счета и выводящую присвоенные значения переменным на экран.

# **3. Ход выполнения работы**

## **3.1 Упражнение 1**

В ходе выполнения данного упражнения написана программа, выводящая на экран значения переменных, включенных в перечислимый тип enum.

### 3.1.1 Пошаговое описание алгоритма

Создать перечислимый тип enum и объявить две переменные, затем присвоить значения cheking и deposite двум переменным.

На экран пользователя выводятся значения переменных.

### 3.1.2 Используемые классы и методы

В программе, написанной в данном упражнении, используются следующие методы:

- std::cout – служит для отображения на экране строк и значений переменных, переданных в метод в качестве параметров, с переходом на новую строку;

- std::cin – ожидает следующего нажатия клавиши пользователем;

- main() – служит для запуска программы.

- enum class– перечислимый тип.

### 3.1.3 Контрольный пример

На рис. 3.1.3.1 представлены результаты выполнения программы 1.

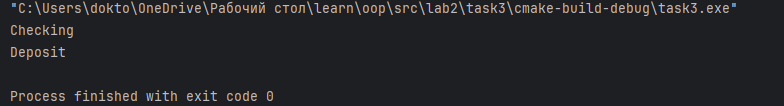
****

Рис. 3.1.3.1 Контрольный пример для упражнения 1

Как видно из рисунка, на экран выведены значения двух переменных.

### 3.2 Упражнение 2

В ходе выполнения данного упражнения, написанная в предыдущем пункте программа, теперь включает в себя структуру и выполняет присвоение переменным значений и вывод их на экран.

## **3.2.1 Пошаговое описание алгоритма**

С помощью структуры struct дополняется программа из упражнения 1.

В ней объявляются три переменные для счета, баланса и типа банковского счета, а в методе main им присваиваются значения.

На экран пользователя выводятся значения данных переменных.

### 3.2.2 Используемые классы и методы

В программе, написанной в данном упражнении, используются следующие методы:

- std::cout – служит для отображения на экране строк и значений переменных, переданных в метод в качестве параметров, с переходом на новую строку;

- std::cin – ожидает следующего нажатия клавиши пользователем [1];

- main() – служит для запуска программы.

- struct – структура;

- enum class – перечислимый тип.

### 3.2.3 Контрольный пример

На рис.3.2.3.1 представлены результаты выполнения программы 2.

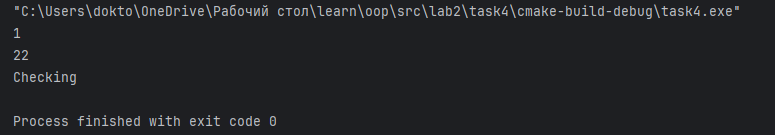
****

Рис.3.2.3.1 Контрольный пример для упражнения 2

Как видно из рисунка, на экран выведены значения core.accNo, core.accBal, core.accType.

# **4. Листинг программы**

**Первая программа:**

**bankState.cpp**

//

// Created by dokto on 10.06.2024.

//

#include "iostream"

#include "bankState.h"

namespace {

/\*\*

\* Outputs the gold and platinum accounts held by the bank state.

\*

\* @return void

\*

\* @throws None

\*/

auto out(const bank::AccountType &state) -> void {

if(state == bank::AccountType::Checking) {

std::cout << "Checking" << std::endl;

} else if(state == bank::AccountType::Deposit) {

std::cout << "Deposit" << std::endl;

}

}

}

/\*\*

\* Constructs a BankState object with the given gold and platinum accounts.

\*

\* @param goldAccount the gold account

\* @param platinumAccount the platinum account

\*/

bank::BankState::BankState(const AccountType &goldAccount, const AccountType &platinumAccount)

: goldAccount(goldAccount), platinumAccount(platinumAccount)

{

}

/\*\*

\* The main function initializes a `BankState` object with two `AccountType`s:

\* `Checking` and `Deposit`. It then calls the `out()` function of the `BankState`

\* object to print the state of the bank. The function returns 0 indicating a

\* successful execution.

\*

\* @return 0 indicating successful execution

\*

\* @throws None

\*/

void bank::BankState::out() const {

::out(goldAccount);

::out(platinumAccount);

}

**bankState.h**

//

// Created by dokto on 10.06.2024.

//

#pragma once

namespace bank {

enum class AccountType{

Checking,

Deposit

};

class BankState {

public:

BankState(const AccountType &goldAccount, const AccountType &platinumAccount);

void out() const;

private:

AccountType goldAccount;

AccountType platinumAccount;

};

}

**main.cpp**

#include "Core/bankState.h"

/\*\*

\* The main function of the program.

\*

\* @return An integer indicating the exit status of the program.

\*/

int main() {

const bank::BankState state(bank::AccountType::Checking, bank::AccountType::Deposit);

state.out();

return 0;

}  
  
**CmakeLists.txt**

cmake\_minimum\_required(VERSION 3.15.0)

include\_guard(GLOBAL)

project(task3

VERSION 0.0.1

DESCRIPTION "task1 for OOP"

LANGUAGES C CXX

)

if(NOT CMAKE\_CXX\_STANDARD)

message(STATUS "[${PROJECT\_NAME}] setting c++ standard to c++23")

set(CMAKE\_CXX\_STANDARD 23)

set(CMAKE\_CXX\_STANDARD\_REQUIRED ON)

set(CMAKE\_CXX\_EXTENSIONS OFF)

endif()

add\_executable( ${PROJECT\_NAME}

${CMAKE\_CURRENT\_SOURCE\_DIR}/src/c++/main.cpp

${CMAKE\_CURRENT\_SOURCE\_DIR}/src/c++/core/bankState.cpp

${CMAKE\_CURRENT\_SOURCE\_DIR}/src/c++/core/bankState.h

)

target\_include\_directories(${PROJECT\_NAME}

PRIVATE

${CMAKE\_CURRENT\_SOURCE\_DIR}/src/c++

)

**Вторая программа:**

**CmakeLists.txt**

cmake\_minimum\_required(VERSION 3.15.0)

include\_guard(GLOBAL)

project(task4

VERSION 0.0.1

DESCRIPTION "task1 for OOP"

LANGUAGES C CXX

)

if(NOT CMAKE\_CXX\_STANDARD)

message(STATUS "[${PROJECT\_NAME}] setting c++ standard to c++23")

set(CMAKE\_CXX\_STANDARD 23)

set(CMAKE\_CXX\_STANDARD\_REQUIRED ON)

set(CMAKE\_CXX\_EXTENSIONS OFF)

endif()

add\_executable( ${PROJECT\_NAME}

${CMAKE\_CURRENT\_SOURCE\_DIR}/src/c++/main.cpp

${CMAKE\_CURRENT\_SOURCE\_DIR}/src/c++/core/bankAccount.cpp

${CMAKE\_CURRENT\_SOURCE\_DIR}/src/c++/core/bankAccount.h

)

target\_include\_directories(${PROJECT\_NAME}

PRIVATE

${CMAKE\_CURRENT\_SOURCE\_DIR}/src/c++

)

**bankAccount.cpp**

//

// Created by dokto on 10.06.2024.

//

#include "bankAccount.h"

#include <iostream>

namespace {

/\*\*

\* Outputs the corresponding account type to the console.

\*

\* @param type The account type to be outputted.

\*

\* @return void

\*

\* @throws None

\*/

auto out(const bank::AccountType type) -> void {

if(type == bank::AccountType::Checking) {

std::cout << "Checking" << std::endl;

} else if(type == bank::AccountType::Deposit) {

std::cout << "Deposit" << std::endl;

}

}

}

/\*\*

\* Constructor for the Core class. Initializes the account member variable with a unique pointer to a BankAccount object.

\*

\* @throws None

\*/

bank::Core::Core():

account(std::make\_unique<BankAccount>(1, 22, bank::AccountType::Checking))

{

}

/\*\*

\* Outputs the account number, balance, and account type of the bank account associated with the Core object.

\*

\* @throws None

\*/

auto bank::Core::out() const -> void {

std::cout << account->accNo << std::endl;

std::cout << account->accBal << std::endl;

::out(account->accType);

}**bankAccount.h**

//

// Created by dokto on 10.06.2024.

//

#pragma once

#include <memory>

namespace bank {

enum class AccountType{

Checking,

Deposit

};

struct BankAccount {

long accNo;

double accBal;

AccountType accType;

};

class Core {

public:

Core();

auto out () const -> void;

private:

std::unique\_ptr<BankAccount> account;

};

}

**main.cpp**

#include <Core/bankAccount.h>

int main()

{

const bank::Core core;

core.out();

return 0;

}

# **5. Полученные результаты**

В ходе выполнения данной лабораторной работы нами были получены следующие результаты:

* в ходе работы программы 1 в перечислимый тип enum были внесены значения cheking и deposite, а затем присвоены двум переменным, которые впоследствии выводятся на экран;
* в ходе работы программы 2 была создана структура, включающая три различных переменных, которым были своены различные значения, а затем выведены на экран.

# **6. Выводы**

В ходе выполнения данной лабораторной работы:

* был изучен перечислимый тип языка C++;
* был изучен тип структура языка C++.