**ДЕРЖАВНИЙ УНІВЕРСИТЕТ ІНТЕЛЕКТУАЛЬНИХ ТЕХНОЛОГІЙ І ЗВ’ЯЗКУ**

**Звіт**

**з дисципліни Проектний Практикум**

**Лабораторна робота №3**

**на тему: «Проектування і реалізація програми з масивами об’єктів, вказівниками та посиланням на них»»**

Виконав: студент групи ІПЗ-3.04

Бухта М.М

       \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Перевірив: Багачук Д.Г.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Одеса  2023**

**ВСТУП**

Мета роботи – засвоєння поняття масиву об’єктів, вказівників та посилань на об’єкти класів; набуття практичних навичок ‘х оголошення та використання.

**ЗАВДАННЯ**

Розрбити і реалізувати програмний додаток нарахування заробітної плати співробітникам фірми. Для збереження даних про співробітників використати масиви об’єктів, а для доступу до функцій – вказівники. Для цього:

1. Розробити і реалізувати програмно клас «Особа»
2. Розробити і реалізувати програмно функції, які враховують таки вимоги:
   1. Кожному співробітнику заробітна плата нараховується, виходячи з тарифної сітки, у якій закладено таки позиції: «Спеціаліст», «Спеціаліст 1 категорії», «Спеціаліст 2 категорії», «Спеціаліст вищої категорії», «Начальник відділу», «Директор».
   2. Кожному співробітнику призначається додатково до ставки премія за принципом: якщо заробітна плата менша від середньої заробітної плати усіх співробітників, то розмір премії становить 50% від заробітної плати, в інших випадках – 30% від заробітної плати.
   3. Реалізувати програмно функцію, яка визначає суму, що видається суму, що видається на руки (заробітна плата плюс премія).
3. Продемонструвати роботи створеного класу і його функцій в основній частині програми

**Порядок виконання роботи**

1. Для реалізації програмного коду оголосити та визначити клас особа. У цього класі слід оголосити закриті змінні, що зберігають такі дані: прізвище та ініціали співробітника, заробітну плату, премію, ставку, позицію з тарифної сітки.
2. Додати у клас відкриті конструктори. Один з конструкторів оголосити без параметрів, інший з параметрами для ініціалізації об’єктів класу. Конструктор не повинен ініціалізації значення ставки значення ставки, натомість ван має викликати функцію, яка її визначає.
3. Додати у клас деструктор.
4. Додати у клас функцію визначення ставки відповідно до позиції у тарифній сітці.
5. Додати у клас функцію призначення премій. Врахувати, що премія призначається за принципом: якщо заробітна плата менша, ніж середня зарплата усіх співробітників, то розмір премії становить 50% від заробітної плати, в інших випадках – 30% від заробітної плати.
6. Додати функцію визначення заробітної плати.
7. Додати функції виведення даних про співробітника.
8. В основній частині програми створити масив об’єктів. Доступ до елементів масиву показати через вказівник.
9. Програма повинна містить меню, яке дозволить перевірити основні функції, створені у програмі.

**КОД ПРОГРАМИ**

**main.cpp**

*/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\**

*\* Laboratory work #3;              \**

*\* Student Bukhta Mykyta;           \**

*\* Grade: 3;                        \**

*\* Group Software Engineering 3.04; \**

*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\**

*\*/*

#include "MainMenu.hpp"

#include <iostream>

using namespace lab\_3;

int main(int argc, char \*\*argv) {

    MainMenu main\_menu;

    main\_menu.draw();

    return 0;

}

**IMenu.hpp**

*/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\**

*\* Laboratory work #3;              \**

*\* Student Bukhta Mykyta;           \**

*\* Grade: 3;                        \**

*\* Group Software Engineering 3.04; \**

*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\**

*\*/*

#ifndef BUKHTAMYKYTA\_LAB3\_IMENU\_HPP

#define BUKHTAMYKYTA\_LAB3\_IMENU\_HPP

#include <inttypes.h>

namespace lab\_3 {

class IMenu {

public:

    virtual void draw(void) = 0;

};

} *// !lab\_3;*

#endif *// !BUKHTAMYKYTA\_LAB3\_IMENU\_HPP;*

**MainMenu.hpp**

*/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\**

*\* Laboratory work #3;              \**

*\* Student Bukhta Mykyta;           \**

*\* Grade: 3;                        \**

*\* Group Software Engineering 3.04; \**

*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\**

*\*/*

#ifndef BUKHTAMYKYTA\_LAB3\_MAINMENU\_HPP

#define BUKHTAMYKYTA\_LAB3\_MAINMENU\_HPP

#include "IMenu.hpp"

#include "Employee.hpp"

#include <deque>

#include <memory>

namespace lab\_3 {

class MainMenu : public IMenu {

private:

    enum class TAB\_ENUM {

        ADD\_WORKER = 1,

        OUTPUT\_WORKER\_INFO,

        OUTPUT\_WORKER\_BONUS\_INFO

    };

public:

    MainMenu(void);

    virtual ~MainMenu(void) = default;

    void draw(void) override;

private:

    void init\_employees(void);

    void handle\_tab(int16\_t tab);

    void open\_add\_worker\_submenu(void);

    void output\_workers\_info(void) const;

    void output\_workers\_bonus\_info(void) const;

private:

    std::deque<std::shared\_ptr<Employee>> m\_employees;

};

} *// !lab\_3;*

#endif *// !BUKHTAMYKYTA\_LAB3\_MAINMENU\_HPP;*

**MainMenu.cpp**

#include "MainMenu.hpp"

#include "SalaryCalculator.hpp"

#include <iostream>

#include <limits>

namespace lab\_3 {

MainMenu::MainMenu(void) {

    init\_employees();

}

void MainMenu::draw(void) {

    static int16\_t tab{0};

    std::cout << "1) Add worker;" << std::endl;

    std::cout << "2) Output worker info;"  << std::endl;

    std::cout << "3) Output worker info with bonuses;" << std::endl;

    std::cout << "Input: "; std::cin >> tab;

    handle\_tab(tab);

}

void MainMenu::init\_employees(void) {

    m\_employees.resize(10);

    m\_employees[0] = std::make\_shared<Employee>(std::move(common\_types::FullName{"Worker", "#1"}),

        common\_types::POSITION::COMMON\_SPECIALIST);

    m\_employees[1] = std::make\_shared<Employee>(std::move(common\_types::FullName{"Worker", "#2"}),

        common\_types::POSITION::COMMON\_SPECIALIST);

    m\_employees[2] = std::make\_shared<Employee>(std::move(common\_types::FullName{"Worker", "#3"}),

        common\_types::POSITION::COMMON\_SPECIALIST);

    m\_employees[3] = std::make\_shared<Employee>(std::move(common\_types::FullName{"Worker", "#4"}),

        common\_types::POSITION::FIRST\_CATEGORY\_SPECIALST);

    m\_employees[4] = std::make\_shared<Employee>(std::move(common\_types::FullName{"Worker", "#5"}),

        common\_types::POSITION::FIRST\_CATEGORY\_SPECIALST);

    m\_employees[5] = std::make\_shared<Employee>(std::move(common\_types::FullName{"Worker", "#6"}),

        common\_types::POSITION::SECOND\_CATEGORY\_SPECIALST);

    m\_employees[6] = std::make\_shared<Employee>(std::move(common\_types::FullName{"Worker", "#7"}),

        common\_types::POSITION::SECOND\_CATEGORY\_SPECIALST);

    m\_employees[7] = std::make\_shared<Employee>(std::move(common\_types::FullName{"Worker", "#8"}),

        common\_types::POSITION::HIGH\_CATEGORY\_SPECIALIST);

    m\_employees[8] = std::make\_shared<Employee>(std::move(common\_types::FullName{"Worker", "#9"}),

        common\_types::POSITION::BRANCH\_MANAGER);

    m\_employees[9] = std::make\_shared<Employee>(std::move(common\_types::FullName{"Worker", "#10"}),

        common\_types::POSITION::DIRECTOR);

}

void MainMenu::handle\_tab(int16\_t tab) {

    switch(static\_cast<TAB\_ENUM>(tab)) {

    case TAB\_ENUM::ADD\_WORKER:

        open\_add\_worker\_submenu();

        break;

    case TAB\_ENUM::OUTPUT\_WORKER\_INFO:

        output\_workers\_info();

        break;

    case TAB\_ENUM::OUTPUT\_WORKER\_BONUS\_INFO:

        output\_workers\_bonus\_info();

        break;

    default:

        std::wcout << "Unknown value!" << std::endl;

    }

    this->draw();

}

void MainMenu::open\_add\_worker\_submenu(void) {

    common\_types::FullName full\_name;

    uint16\_t position;

    std::cout << "Input First Name:\n";

    std::getline(std::cin, full\_name.first\_name).ignore(std::numeric\_limits<std::streamsize>::max(), '\n');

    std::cout << "Input Second Name:\n";

    std::getline(std::cin, full\_name.second\_name);

    std::cout << "Input Middle Name:\n";

    std::getline(std::cin, full\_name.middle\_name);

    do { *// Input position number;*

        std::cout << "Input position list:" << std::endl

        << "0 - COMMON\_SPECIALIST" << std::endl

        << "1 - FIRST\_CATEGORY\_SPECIALST" << std::endl

        << "2 - SECOND\_CATEGORY\_SPECIALST" << std::endl

        << "3 - HIGH\_CATEGORY\_SPECIALIST" << std::endl

        << "4 - BRANCH\_MANAGER" << std::endl

        << "5 - DIRECTOR" << std::endl

        << "Input: ";

    std::cin >> position;

    }   while (position >= static\_cast<uint16\_t>(common\_types::POSITION::SIZE));

    m\_employees.push\_back(std::make\_shared<Employee>(full\_name, static\_cast<common\_types::POSITION>(position)));

}

void MainMenu::output\_workers\_info(void) const{

    for (const auto &item : m\_employees) {

        std::cout << item->full\_name() << '\t' << common\_types::to\_string(item->position())

            << "\tsalary: " << item->salary() << std::endl;

    }

}

void MainMenu::output\_workers\_bonus\_info(void) const{

    auto salary\_calculator = SalaryCalculator::instance();

    for (const auto &item : m\_employees) {

        std::cout << item->full\_name() << '\t' << common\_types::to\_string(item->position())

            << "\tsalary with bonus: " << salary\_calculator->calculate\_with\_bonus(\*item, m\_employees) << std::endl;

    }

}

} *// !lab\_3;*

**Employee.hpp**

*/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\**

*\* Laboratory work #3;              \**

*\* Student Bukhta Mykyta;           \**

*\* Grade: 3;                        \**

*\* Group Software Engineering 3.04; \**

*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\**

*\*/*

#ifndef BUKHTAMYKYTA\_LAB3\_EMPLOYEE\_HPP

#define BUKHTAMYKYTA\_LAB3\_EMPLOYEE\_HPP

#include "CommonTypes.hpp"

#include <inttypes.h>

namespace lab\_3 {

class Employee {

public:

    Employee(common\_types::POSITION position = common\_types::POSITION::COMMON\_SPECIALIST);

    Employee(const common\_types::FullName &full\_name, common\_types::POSITION position = common\_types::POSITION::COMMON\_SPECIALIST);

    virtual ~Employee(void) = default;

*// Getters starts;*

    virtual uint32\_t salary(void) const noexcept;

    virtual common\_types::POSITION position(void) const noexcept;

    virtual common\_types::FullName full\_name(void) const;

*// Getters ends;*

*// Setters starts;*

    virtual void set\_salary(uint32\_t val) noexcept;

    virtual void set\_position(common\_types::POSITION val) noexcept;

    virtual void set\_full\_name(const common\_types::FullName &val);

    virtual void set\_full\_name(common\_types::FullName &&val);

*// Setters ends;*

private:

    uint32\_t m\_salary;

    common\_types::POSITION m\_position;

    common\_types::FullName m\_full\_name;

};

} *// !lab\_3;*

#endif *// !BUKHTAMYKYTA\_LAB3\_EMPLOYEE\_HPP;*

**Employee.cpp**

#include "Employee.hpp"

#include "SalaryCalculator.hpp"

namespace lab\_3 {

Employee::Employee(common\_types::POSITION position)

    : Employee({}, position)

{

}

Employee::Employee(const common\_types::FullName &full\_name, common\_types::POSITION position)

    : m\_full\_name(full\_name), m\_position(position)

{

    auto salary\_calculator = SalaryCalculator::instance();

    this->set\_salary(salary\_calculator->calculate(\*this));

}

*// Getters starts;*

uint32\_t Employee::salary(void) const noexcept {

    return m\_salary;

}

common\_types::POSITION Employee::position(void) const noexcept {

    return m\_position;

}

common\_types::FullName Employee::full\_name(void) const {

    return m\_full\_name;

}

*// Getters ends;*

*// Setters starts;*

void Employee::set\_salary(uint32\_t val) noexcept {

    m\_salary = val;

}

void Employee::set\_position(common\_types::POSITION val) noexcept {

    m\_position = val;

}

void Employee::set\_full\_name(const common\_types::FullName &val) {

    m\_full\_name = val;

}

void Employee::set\_full\_name(common\_types::FullName &&val) {

    m\_full\_name = std::move(val);

}

*// Setters ends;*

} *// !lab\_3;*

**SalaryCalculator.hpp**

*/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\**

*\* Laboratory work #3;              \**

*\* Student Bukhta Mykyta;           \**

*\* Grade: 3;                        \**

*\* Group Software Engineering 3.04; \**

*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\**

*\*/*

#ifndef BUKHTAMYKYTA\_LAB3\_SALARYCALCULATOR\_HPP

#define BUKHTAMYKYTA\_LAB3\_SALARYCALCULATOR\_HPP

#include "Employee.hpp"

#include <unordered\_map>

#include <deque>

#include <memory>

#include <inttypes.h>

namespace lab\_3 {

class SalaryCalculator {

public:

    static SalaryCalculator\* instance(void);

    uint32\_t calculate(const Employee &employee) const;

    uint32\_t calculate\_with\_bonus(const Employee &employee, const std::deque<std::shared\_ptr<Employee>> employees) const;

private:

    SalaryCalculator(void);

private:

    std::unordered\_map<common\_types::POSITION, int32\_t> m\_salaries;

};

} *// !lab\_3;*

#endif *// !BUKHTAMYKYTA\_LAB3\_SALARYCALCULATOR\_HPP;*

**SalaryCalculator.cpp**

#include "SalaryCalculator.hpp"

namespace lab\_3 {

SalaryCalculator\* SalaryCalculator::instance(void) {

    static SalaryCalculator instance;

    return &instance;

}

uint32\_t SalaryCalculator::calculate(const Employee &employee) const {

    auto salary\_rate = m\_salaries.find(employee.position());

    if (salary\_rate == m\_salaries.end()) {

        return 0;

    }

    return salary\_rate->second;

}

uint32\_t SalaryCalculator::calculate\_with\_bonus(const Employee &employee, const std::deque<std::shared\_ptr<Employee>> employees) const {

    uint64\_t monthly\_salary{0};

    uint32\_t avarage\_salary{0};

    uint32\_t bonus{0};

    uint32\_t employee\_salary{employee.salary()};

    for (const auto &empl : employees) {

        monthly\_salary += empl->salary();

    }

    avarage\_salary = monthly\_salary / employees.size();

    employee\_salary < avarage\_salary

        ? bonus = employee\_salary / 2

        : bonus = employee\_salary / 100 \* 30;

    return employee\_salary + bonus;

}

SalaryCalculator::SalaryCalculator(void) {

    m\_salaries[common\_types::POSITION::COMMON\_SPECIALIST] = 10000;

    m\_salaries[common\_types::POSITION::FIRST\_CATEGORY\_SPECIALST] = 12500;

    m\_salaries[common\_types::POSITION::SECOND\_CATEGORY\_SPECIALST] = 15000;

    m\_salaries[common\_types::POSITION::HIGH\_CATEGORY\_SPECIALIST] = 17500;

    m\_salaries[common\_types::POSITION::BRANCH\_MANAGER] = 20000;

    m\_salaries[common\_types::POSITION::DIRECTOR] = 22500;

}

} *// !lab\_3;*

**CommonTypes.hpp**

*/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\**

*\* Laboratory work #3;              \**

*\* Student Bukhta Mykyta;           \**

*\* Grade: 3;                        \**

*\* Group Software Engineering 3.04; \**

*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\**

*\*/*

#ifndef BUKHTAMYKYTA\_LAB3\_COMMONTYPES\_HPP

#define BUKHTAMYKYTA\_LAB3\_COMMONTYPES\_HPP

#include <string>

#include <ostream>

namespace lab\_3 {

namespace common\_types {

struct FullName {

    std::string first\_name;

    std::string second\_name;

    std::string middle\_name;

    friend std::ostream& operator<<(std::ostream &out, const FullName &value);

};

enum class POSITION {

    COMMON\_SPECIALIST,

    FIRST\_CATEGORY\_SPECIALST,

    SECOND\_CATEGORY\_SPECIALST,

    HIGH\_CATEGORY\_SPECIALIST,

    BRANCH\_MANAGER,

    DIRECTOR,

    SIZE

};

std::string to\_string(POSITION val);

} *// !common\_types;*

} *// !lab\_3;*

#endif *// !BUKHTAMYKYTA\_LAB3\_COMMONTYPES\_HPP;*

**CommonTypes.cpp**

#include "CommonTypes.hpp"

namespace lab\_3 {

namespace common\_types {

std::ostream& operator<<(std::ostream &out, const FullName &value) {

    out << value.second\_name << ' ' << value.first\_name;

    if (value.middle\_name.size()) {

        out << ' ' << value.middle\_name;

    }

    return out;

}

std::string to\_string(POSITION val) {

    switch(val) {

    case POSITION::COMMON\_SPECIALIST:

        return "COMMON\_SPECIALIST";

    case POSITION::FIRST\_CATEGORY\_SPECIALST:

        return "FIRST\_CATEGORY\_SPECIALST";

    case POSITION::SECOND\_CATEGORY\_SPECIALST:

        return "SECOND\_CATEGORY\_SPECIALST";

    case POSITION::HIGH\_CATEGORY\_SPECIALIST:

        return "HIGH\_CATEGORY\_SPECIALIST";

    case POSITION::BRANCH\_MANAGER:

        return "BRANCH\_MANAGER";

    case POSITION::DIRECTOR:

        return "DIRECTOR";

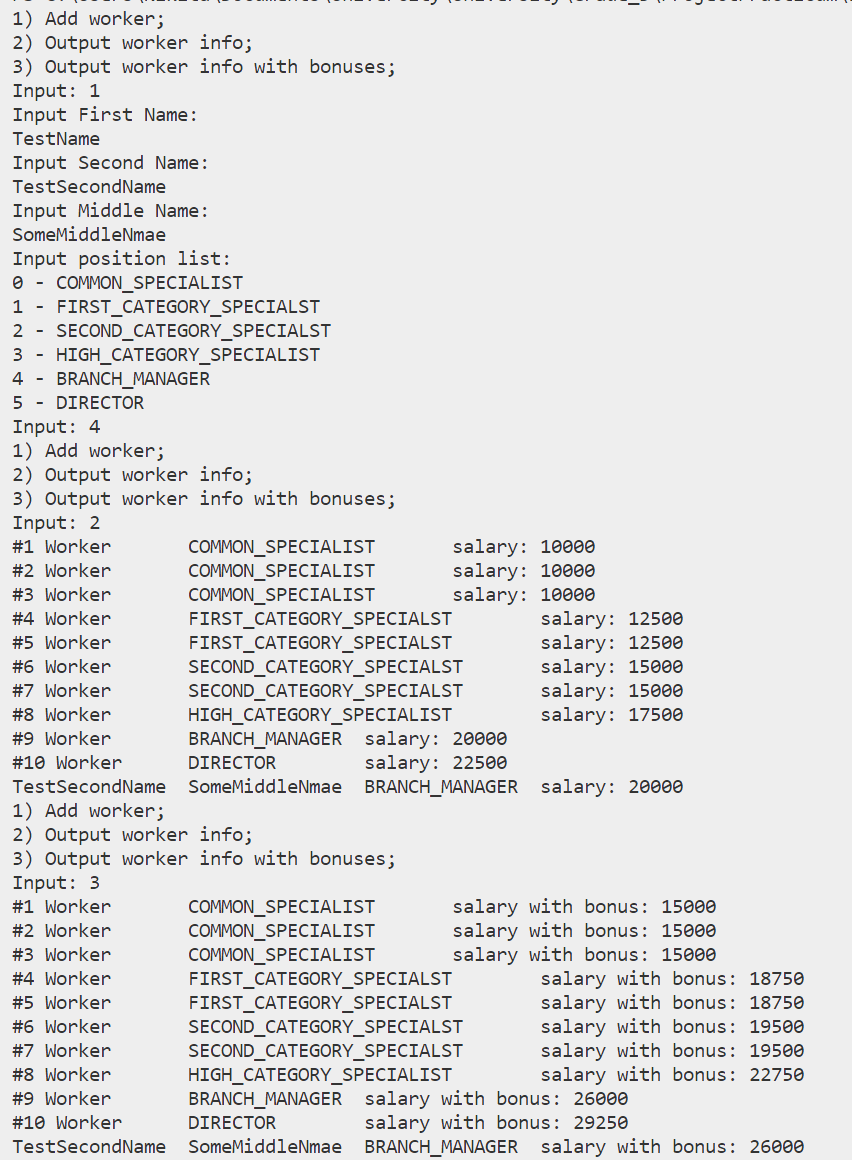
    }

    return {};

}

} *// !common\_types;*

} *// !lab\_3;*

**РЕЗУЛЬТАТ ВИКОНАННЯ**  


**КОНТРОЛЬНІ ЗАВДАННЯ ТА ЗАПИТАННЯ**

1. Як оголосити вказівник на об’єкт?

Type \*obj\_ptr = &obj; // Декларація та ініціалізація вказівника

1. Що таке вказівник this?

Це вказівник на осередок пам’яті, де ініціалізований об’єкт типу класа

1. Як звернутися у програмі до елементів масиву об’єктів.

Щоб звернутися до елементу масиву любого об’єкта, треба узяти вказівник на перший елемент масиву та додати до нього індекс елемента:

Type type\_array[10];

(type\_array + 3)->do\_some\_method(); // звернення до четвертого елемента масиву

\*(type\_array + 3).do\_some\_method(); // aбо

type\_array[3].so\_some\_method // або

1. Як отримати доступ елементу масива об’єктів до відкритих членів класу.

Щоб отримати доступ до відкрити членів класу масива, треба звернутися до потрібного елементу масиву, як показано у питанні 3 та через крапку або стрілку (відповідно до типу об’єкту: вказівник чи об’єкт) обратитися до відкритого члену класу. Приклад у питання 3.

1. Якими способами можна виконати ініціалізацію масиву об’єктів?
   1. Під час декларації масиву:

int32\_t array[5] = {1, 2, 3, 4, 5};

* 1. Через цикл

int32\_t array[5];

for (int32\_t i{0}; i < 5; ++i) { array[i] = i; }

* 1. memset();

int32\_t array[5];

memset(array, 0, 5 \* sizeof(\*array));

1. Як оголосити двомірний масив об’єктів?

Потрібно два рази указати розмір масиву. Перший – кількість рядків, другий – кількість стовпців

int32\_t array[5][2]; // масив 5х2

1. Дано прототип конструктора myclass(int, char, char\*). Оголосить масив чотирьох об’єктів

myclass myclass\_arr[4];