**ЕКЗАМЕНАЦІЙНА РОБОТА**

**з дисципліни “Алгоритмічні мови та програмування”**

**ст.гр. ПМ-21-1**

Сиваш Катерина

**Білет №** 25

**Питання № 1:**

Поняття масиву та їх різновиди. Поняття динамічного масиву. Оголошення двомірного динамічного масиву мовою С++. Вказівник на вказівник. Індексація в масивах за допомогою вказівників мовою С++.

**Відповідь на перше питання білету:**

**Масив** - це упорядкована послідовність змінних одного типу . В залежності від сенсу інформації яку необхідно буде зберігати розрізняють: одномірні та багатомірні масиви. Динамічним називають такий масив, розмір якого можна змінювати під час виконання програми. Динамічні масиви надають змогу більш гнучко працювати з даними, оскільки дозволяють вводити довільний розмір. При створенні двовимірного динамічного масиву , різні його частини (наприклад , рядки) можуть перебувати в різних областях пам'яті.

Використання масивів покажчиків і засобів динамічного розподілення пам'яті при роботі з двовимірними масивами дозволяє раціонально розподіляти пам'ять.

При створенні двовимірного динамічного масиву спочатку виділяється пам'ять під одновимірний масив покажчиків , в якому будуть зберігатися адреси одновимірних масивів . Адреса цього масиву покажчиків запам'ятовується у змінній , яка оголошена як покажчик на покажчик . Потім виділяється пам'ять під кожний з одновимірних масивів, при цьому їх адреси заносяться у відповідні елементи масиву покажчиків .  
Покажчик на покажчик - це вказівник, який містить адресу іншого вказівника. В інших словах, це вказівник, який вказує на пам'ять, де знаходиться інший вказівник. Використовуючи покажчик на покажчик, можно створювати більш складні структури даних, такі як двовимірні масиви або списки вказівників.

Елементи масиву розташовані в пам’яті послідовно.Також фіксований масив може конвертуватися на вказівник, який вказує на перший елемент (елемент під індексом 0) масиву,додавання одиниці до вказівника повертає адресу в пам’яті наступного об’єкта цього ж типу даних.

Отже, можна припустити, що додавання одиниці до ідентифікатора масиву призведе до повернення адреси другого елемента (елементу під індексом (1) масиву в пам’яті.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14 | #include <iostream>    int main()  {       int array [5] = { 7, 8, 2, 4, 5 };         std::cout << &array[1] << '\n'; // виведеться адреса в пам'яті елементу під індексом 1       std::cout << array+1 << '\n'; // виведеться адреса в пам'яті вказівника на масив + 1         std::cout << array[1] << '\n'; // виведеться 8       std::cout << \*(array+1) << '\n'; // виведеться 8        return 0;  } |

Коли компілятор бачить оператор індексу [], він, насправді, конвертує його у вказівник з операцією додавання і розіменування

**Приклад масиву:**

#include <iostream>

int main()

{

int array[5]; // масив з п'яти чисел

array[0] = 3; // індексом першого елементу є 0 (нульовий елемент)

array[1] = 2;

array[2] = 4;

array[3] = 8;

array[4] = 12; // індексом останнього елементу є 4

std::cout << "The lowest number is " << array[0] << "\n";

std::cout << "The sum of the first 5 numbers is " << array[0] + array[1] + array[2] + array[3] + array[4] << "\n";

return 0;

}

приклад використання двовимірного масиву:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25 | #include <iostream>    int main()  {      // Оголошуємо масив 10x10      const int numRows = 10;      const int numCols = 10;      int product[numRows][numCols] = { 0 };        // Створюємо таблицю множення      for (int row = 0; row < numRows; ++row)          for (int col = 0; col < numCols; ++col)              product[row][col] = row \* col;        // Виводимо таблицю множення      for (int row = 1; row < numRows; ++row)      {          for (int col = 1; col < numCols; ++col)              std::cout << product[row][col] << "\t";            std::cout << '\n';      }        return 0;  } |

**Питання № 2:**

Поняття об’єктно-орієнтованого програмування. Поняття класу та об’єкту. Основні принципи об’єктно-орієнтованого програмування.

**Відповідь на друге питання білету:**

ООП, або об'єктно-орієнтоване програмування, є парадигмою програмування, яка базується на концепції "об'єктів". Основні принципи ООП включають принципи абстракції, інкапсуляції, спадкування та поліморфізму.

Клас — це визначений користувачем тип або структура даних, оголошена ключовим словом class , яка містить дані і функції як свої члени, доступ до якої регулюється трьома специфікаторами доступу: private, public, protected (за умовчанням - private).

Об’єкт – це базове поняття в ООП, це конкретна реалізація, екземпляр класу.

**Приклад:**

Клас Date з методом виводу дати:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12 | class DateClass  {  public:      int m\_day;      int m\_month;      int m\_year;        void print() // визначаємо метод      {          std::cout << m\_day << "/" << m\_month << "/" << m\_year;      }  }; |

Далі цей клас можно використовувати у ході програми.

**Приклад використання об'єктів:**

#include <iostream>

#include <string>

// Клас "Автомобіль"

class Car {

public:

// Конструктор класу

Car(std::string make, std::string model, int year) {

make\_ = make;

model\_ = model;

year\_ = year;

}

// Метод для отримання інформації про автомобіль

void displayInfo() {

std::cout << "Марка: " << make\_ << "\nМодель: " << model\_ << "\nРік: " << year\_ << std::endl;

}

private:

// Закриті поля класу

std::string make\_;

std::string model\_;

int year\_;

};

int main() {

// Створення об'єкта класу "Автомобіль"

Car myCar("Toyota", "Camry", 2022);

// Виклик методу для відображення інформації про автомобіль

myCar.displayInfo();

return 0;

}

**Питання № 3:**

Розробити власний клас з двома полями, що містить конструктор за замовчуванням, методи для ініціалізації даних, методи введення та виведення на екран. Розробити метод, що перевіряє чи кратне перше поле другому. Перевантажте операцію порівняння / (a,b)/(c,d)=(a/b, c/d). В головній функції продемонструвати роботу з масивом об’єктів.

**Відповідь на третє питання білету:**

**Текст програми:**

#include <iostream>

class Point

{

public:

// Конструктор за замовчуванням

Point() = default;

// Конструктор з параметрами для ініціалізації точки з вказаними координатами

Point(int x, int y) : x(x), y(y) {}

// Методи для встановлення та отримання значень координат x та y

void setX(int x) { this->x = x; }

void setY(int y) { this->y = y; }

int getX() const { return x; }

int getY() const { return y; }

// Метод, який повертає true, якщо x кратне y

bool isMultiple() const { return x % y == 0; }

// Перевантаження оператора виводу для виведення точки у форматі (x, y)

friend std::ostream& operator<<(std::ostream& os, const Point& point)

{

os << "(" << point.x << ", " << point.y << ")";

return os;

}

// Перевантаження оператора ділення для точкових об'єктів

Point operator/(const Point& other) const

{

return Point(x / other.x, y / other.y);

}

private:

// Закриті поля класу для координат x та y

int x;

int y;

};

int main()

{

// Створення об'єктів класу Point

Point point1(10, 20);

Point point2(20, 40);

// Виведення значень точок

std::cout << "point1: " << point1 << std::endl;

std::cout << "point2: " << point2 << std::endl;

// Використання оператора / для точкових об'єктів та виведення результату

std::cout << "point1 / point2: " << point1 / point2 << std::endl;

// Масив точок

Point points[] = {

Point(40, 20),

Point(20, 40),

Point(30, 60),

Point(40, 80),

};

// Перевірка та виведення точок, де x кратне y

for (const Point& point : points)

{

if (point.isMultiple())

{

std::cout << point << std::endl;

}

}

return 0;

}

**Результати роботи програми:**

point1: (10, 20)

point2: (20, 40)

point1 / point2: (0, 0)

(40, 20)

