**Билет 1.**

**1. Абстрактные классы: назначение и примеры**

Абстрактный класс в C++ — это класс, который не может быть создан (нельзя создать его объект), но от него можно наследоваться. Он нужен, когда есть базовый класс с общим функционалом, но часть методов должны быть реализованы в дочерних классах.

## ****Как определить абстрактный класс?****

Если хотя бы один метод в классе является **чисто виртуальным** (имеет = 0 в объявлении), то класс становится абстрактным.

#include <iostream>

**using** **namespace** std**;**

class Shape **{** // Абстрактный класс

public**:**

virtual void draw**()** **=** 0**;** // Чисто виртуальная функция

**};**

class Circle **:** public Shape **{**

public**:**

void draw**()** override **{**

cout **<<** "Рисую круг" **<<** endl**;**

**}**

**};**

int main**()** **{**

// Shape s; // Ошибка! Нельзя создать объект абстрактного класса

Circle c**;**

c**.**draw**();** // Выведет: "Рисую круг"

**return** 0**;**

**}**

Здесь Shape — абстрактный класс, потому что у него есть virtual void draw() = 0;, а Circle реализует этот метод.

Теперь напишем программу на C++, которая:

А) Создаст массив bool, где каждый элемент хранится в одном бите. Код:

#include <iostream>

#include <vector>

class BoolArray **{**

private**:**

std**::**vector**<**unsigned int**>** data**;** // Вектор для хранения битов

size\_t size**;** // Размер логического массива

public**:**

// Конструктор: выделяем память под динамический массив int-ов

BoolArray**(**size\_t n**)** **:** size**(**n**)** **{**

size\_t intCount **=** **(**n **+** 31**)** **/** 32**;** // Количество int-ов (каждый хранит 32 бита)

data**.**resize**(**intCount**,** 0**);** // Заполняем нулями

**}**

// Установка значения бита

void set**(**size\_t index**,** bool value**)** **{**

**if** **(**index **>=** size**)** **return;**

size\_t intIndex **=** index **/** 32**;** // Определяем, в каком числе хранится бит

size\_t bitIndex **=** index **%** 32**;** // Определяем позицию бита в числе

**if** **(**value**)**

data**[**intIndex**]** **|=** **(**1 **<<** bitIndex**);** // Устанавливаем бит

**else**

data**[**intIndex**]** **&=** **~(**1 **<<** bitIndex**);** // Очищаем бит

**}**

// Получение значения бита

bool get**(**size\_t index**)** const **{**

**if** **(**index **>=** size**)** **return** **false;**

size\_t intIndex **=** index **/** 32**;**

size\_t bitIndex **=** index **%** 32**;**

**return** **(**data**[**intIndex**]** **>>** bitIndex**)** **&** 1**;**

**}**

// Вывод массива

void print**()** const **{**

**for** **(**size\_t i **=** 0**;** i **<** size**;** i**++)**

std**::**cout **<<** get**(**i**);**

std**::**cout **<<** std**::**endl**;**

**}**

**};**

// Пример работы

int main**()** **{**

BoolArray arr**(**10**);** // Создаем логический массив на 10 элементов

arr**.**set**(**0**,** 1**);**

arr**.**set**(**3**,** 1**);**

arr**.**set**(**9**,** 1**);**

arr**.**print**();** // Выведет 1001000001

**return** 0**;**

**}**

Б) Реализует двоичный счётчик на этом массиве. Код:

#include <iostream>

#include <vector>

class BinaryCounter **{**

private**:**

std**::**vector**<**unsigned int**>** data**;** // Массив битов

size\_t size**;** // Количество разрядов

public**:**

// Конструктор

BinaryCounter**(**size\_t n**)** **:** size**(**n**)** **{**

size\_t intCount **=** **(**n **+** 31**)** **/** 32**;**

data**.**resize**(**intCount**,** 0**);** // Заполняем массив нулями

**}**

// Установка бита

void set**(**size\_t index**,** bool value**)** **{**

**if** **(**index **>=** size**)** **return;**

size\_t intIndex **=** index **/** 32**;**

size\_t bitIndex **=** index **%** 32**;**

**if** **(**value**)**

data**[**intIndex**]** **|=** **(**1 **<<** bitIndex**);**

**else**

data**[**intIndex**]** **&=** **~(**1 **<<** bitIndex**);**

**}**

// Получение значения бита

bool get**(**size\_t index**)** const **{**

**if** **(**index **>=** size**)** **return** **false;**

size\_t intIndex **=** index **/** 32**;**

size\_t bitIndex **=** index **%** 32**;**

**return** **(**data**[**intIndex**]** **>>** bitIndex**)** **&** 1**;**

**}**

// Увеличение счетчика на 1 (аналогично прибавлению 1 в двоичной системе)

void increment**()** **{**

**for** **(**size\_t i **=** 0**;** i **<** size**;** i**++)** **{**

**if** **(!**get**(**i**))** **{** // Если бит равен 0, ставим 1 и выходим (аналог +1)

set**(**i**,** 1**);**

**return;**

**}** **else** **{** // Если бит равен 1, обнуляем его и переходим к следующему (перенос)

set**(**i**,** 0**);**

**}**

**}**

**}**

// Вывод состояния счетчика

void print**()** const **{**

**for** **(**size\_t i **=** 0**;** i **<** size**;** i**++)**

std**::**cout **<<** get**(**size **-** 1 **-** i**);** // Выводим старшие разряды первыми

std**::**cout **<<** std**::**endl**;**

**}**

**};**

// Пример работы счетчика

int main**()** **{**

size\_t N **=** 5**;** // Количество разрядов

BinaryCounter counter**(**N**);**

// Выведем 5 первых значений счетчика

**for** **(**int i **=** 0**;** i **<** 10**;** i**++)** **{**

counter**.**print**();**

counter**.**increment**();**

**}**

**return** 0**;**

**}**

1. **Часть (а)**:
   * Реализует класс BoolArray, который хранит bool-значения в битах unsigned int.
   * Использует битовые операции для установки (set) и получения (get) значений.
   * Позволяет напечатать массив битов в виде строки.
2. **Часть (б)**:
   * Расширяет идею BoolArray и делает **N-разрядный двоичный счетчик**.
   * Реализует метод increment(), который имитирует работу двоичного счетчика.
   * Позволяет напечатать текущее состояние счетчика.