Вариант 1.

Реализовать класс-контейнер «Односвязный список (List)», состоящий из объектов произвольного (но одного и того же) типа. Методы класса должны принимать указатель на абстрактный класс IObject. В классе IObject необходимо определить чисто виртуальные функции bool equal(IObject*) – для сравнения объектов на равенство и IObject* clone() – для создания копии объекта при добавлении в контейнер. Класс IObject должен быть базовым для классов различных типов объектов, которые может содержать список (IntObject, StringObject). Все эти классы должны переопределять чисто виртуальные функции класса IObject. Реализовать в классе «Список» несколько конструкторов, в том числе конструктор копирования, методы для поиска элемента, добавления элемента в список, удаления элемента из списка. Проиллюстрировать работу с классом «Список» для хранения различных типов объектов.

Вариант 2.

Реализовать класс-контейнер «Отображение (Map)», состоящий из объектов (пары ключ-значение) произвольного типа (но тип ключей один и тот же для одного отображения, тип значений также один и тот же в рамках одного отображения, но может отличаться от типа ключей). Методы класса должны принимать указатель на абстрактный класс IObject. В классе IObject необходимо определить чисто виртуальные функции bool less(IObject*) – для сравнения объектов на меньше и IObject* clone() – для создания копии объекта при добавлении в контейнер. Класс IObject должен быть базовым для классов различных типов ключей и значений, которые может содержать отображение (IntObject, CharObject, StringObject). Все эти классы должны переопределять чисто виртуальные функции класса IObject. Реализовать в классе «Отображение» несколько конструкторов, в том числе конструктор копирования, методы для добавления нового элемента в отображение, удаления элемента, поиска по ключу и вывода на экран. Элементы отображения хранить в упорядоченном массиве структур, поиск элементов осуществлять с помощью бинарного поиска. Проиллюстрировать работу с классом «Отображение» для хранения различных типов ключей и значений.

Вариант 3.

Реализовать класс-контейнер «Множество (Set)», состоящее из объектов произвольного (но одного и того же) типа. Методы класса должны принимать указатель на абстрактный класс IObject. В классе IObject необходимо определить чисто виртуальные функции bool equal(IObject*) – для сравнения объектов на равенство и IObject* clone() – для создания копии объекта при добавлении в контейнер. Класс IObject должен быть базовым для

классов различных типов объектов, которые может содержать множество (IntObject, DoubleObject, StringObject). Все эти классы должны переопределять чисто виртуальные функции класса IObject. Реализовать в классе «Множество» несколько конструкторов, в том числе конструктор копирования, методы для определения принадлежности заданного элемента множеству, добавления, удаления элемента из множества, пересечения, объединения, разности двух множеств. Элементы множества хранить в неупорядоченном массиве (содержащем значения типа IObject*), поиск элементов осуществлять с помощью перебора. Продемонстрировать работу с классом «Множество» для хранения различных типов объектов.

Вариант 4.

Реализовать класс-контейнер «Стек (Stack)», состоящий из объектов произвольного (но одного и того же) типа. Методы класса должны принимать указатель на абстрактный класс IObject (содержащий чисто виртуальную функцию IObject* clone() для создания копии объекта при добавлении в контейнер). Класс IObject должен быть базовым для классов различных типов объектов, которые может содержать стек (IntObject, DoubleObject, StringObject). Все эти классы должны переопределять чисто виртуальную функцию класса IObject. Реализовать в классе «Стек» несколько конструкторов, в том числе конструктор копирования, методы для добавления элемента в стек и извлечения элемента из стека. Проиллюстрировать работу с классом «Стек» для хранения различных типов объектов.

Вариант 5.

Реализовать класс-контейнер «Множество (Set)», состоящее из объектов произвольного (но одного и того же) типа. Методы класса должны принимать указатель на абстрактный класс IObject. В классе IObject необходимо определить чисто виртуальные функции bool less(IObject*) – для сравнения объектов на меньше и IObject* clone() – для создания копии объекта при добавлении в контейнер. Класс IObject должен быть базовым для классов различных типов объектов, которые может содержать множество (IntObject, StringObject). Все эти классы должны переопределять чисто виртуальные функции класса IObject. Реализовать в классе «Множество» несколько конструкторов, в том числе конструктор копирования, методы для определения принадлежности заданного элемента множеству, добавления, удаления элемента из множества, пересечения, объединения, разности двух множеств. Элементы множества хранить в упорядоченном массиве (содержащем значения типа IObject*), поиск элементов осуществлять на

основе бинарного поиска. Продемонстрировать работу с классом «Множество» для хранения различных типов объектов.

Вариант 6.

Реализовать класс-контейнер «Мультимножество (Multiset)», состоящее из объектов произвольного (но одного и того же) типа. Методы класса должны принимать указатель на абстрактный класс IObject. В классе IObject необходимо определить чисто виртуальные функции bool less(IObject*) – для сравнения объектов на меньше и IObject* clone() – для создания копии объекта при добавлении в контейнер. Класс IObject должен быть базовым для классов различных типов объектов, которые может содержать множество (IntObject, DoubleObject, StringObject). Все эти классы должны переопределять чисто виртуальные функции класса IObject. Реализовать в классе «Мультимножество» несколько конструкторов, в том числе конструктор копирования, методы для определения принадлежности заданного элемента множеству, добавления, удаления элемента из множества, пересечения, объединения, разности двух множеств. Элементы множества хранить в упорядоченном массиве (содержащем значения типа IObject*), поиск элементов осуществлять на основе бинарного поиска. Продемонстрировать работу с классом «Мультимножество» для хранения различных типов объектов.

Вариант 7.

Реализовать класс-контейнер «Упорядоченное бинарное дерево (Tree)», состоящий из объектов произвольного (но одного и того же) типа. Методы класса должны принимать указатель на абстрактный класс IObject. В классе IObject необходимо определить чисто виртуальные функции bool less(IObject*) – для сравнения объектов на меньше и IObject* clone() – для создания копии объекта при добавлении в контейнер. Класс IObject должен быть базовым для классов различных типов объектов, которые может содержать дерево (IntObject, StringObject). Все эти классы должны переопределять чисто виртуальные функции класса IObject. Реализовать в классе «Дерево» несколько конструкторов, в том числе конструктор копирования, методы для добавления нового элемента, удаления элемента, поиска элемента и вывода на экран. Проиллюстрировать работу с классом «Дерево» для хранения различных типов элементов.

Вариант 8.

Реализовать класс-контейнер «Двусвязный список (List)», состоящий из объектов произвольного (но одного и того же) типа. Методы класса должны принимать указатель на абстрактный класс IObject. В классе IObject необходимо определить чисто виртуальные функции bool equal(IObject*) – для сравнения объектов на равенство и IObject* clone() – для создания копии объекта при добавлении в контейнер. Класс IObject должен быть базовым для классов различных типов объектов, которые может содержать список (IntObject, DoubleObject, StringObject). Все эти классы должны переопределять чисто виртуальные функции класса IObject. Реализовать в классе «Двусвязный список» несколько конструкторов, в том числе конструктор копирования, методы для поиска элемента, добавления элемента в список, удаления элемента из списка. Проиллюстрировать работу с классом «Двусвязный список» для хранения различных типов объектов.

Вариант 9.

Реализовать класс-контейнер «Отображение (Map)», состоящий из объектов (пары ключ-значение) произвольного типа (но тип ключей один и тот же для одного отображения, тип значений также один и тот же в рамках одного отображения, но может отличаться от типа ключей). Методы класса должны принимать указатель на абстрактный класс IObject. В классе IObject необходимо определить чисто виртуальные функции bool equal(IObject*) – для сравнения объектов на равенство и IObject* clone() – для создания копии объекта при добавлении в контейнер. Класс IObject должен быть базовым для классов различных типов ключей и значений, которые может содержать отображение (IntObject, StringObject). Все эти классы должны переопределять чисто виртуальные функции класса IObject. Реализовать в классе «Отображение» несколько конструкторов, в том числе конструктор копирования, методы для добавления нового элемента в отображение, удаления элемента, поиска по ключу и вывода на экран. Элементы отображения хранить в неупорядоченном массиве структур, поиск элементов осуществлять с помощью перебора. Проиллюстрировать работу с классом «Отображение» для хранения различных типов ключей и значений.

Вариант 10.

Реализовать класс-контейнер «Мультимножество (Multiset)», состоящее из объектов произвольного (но одного и того же) типа. Методы класса должны принимать указатель на абстрактный класс IObject. В классе IObject необходимо определить чисто виртуальные функции bool equal(IObject*) – для сравнения объектов на равенство и IObject* clone() – для создания копии объекта при добавлении в контейнер. Класс IObject должен быть базовым для классов различных типов объектов, которые может содержать множество

(IntObject, DoubleObject, StringObject). Все эти классы должны переопределять чисто виртуальные функции класса IObject. Реализовать в классе «Мультимножество» несколько конструкторов, в том числе конструктор копирования, методы для определения принадлежности заданного элемента множеству, добавления, удаления элемента из множества, пересечения, объединения, разности двух множеств. Элементы множества хранить в неупорядоченном массиве (содержащем значения типа IObject*), поиск элементов осуществлять с помощью перебора. Продемонстрировать работу с классом «Мультимножество» для хранения различных типов объектов.

Вариант 11.

Реализовать класс-контейнер «Очередь (Queue)», состоящий из объектов произвольного (но одного и того же) типа. Методы класса должны принимать указатель на абстрактный класс IObject (содержащий чисто виртуальную функцию IObject* clone() для создания копии объекта при добавлении в контейнер). Класс IObject должен быть базовым для классов различных типов объектов, которые может содержать очередь (IntObject, DoubleObject, StringObject). Все эти классы должны переопределять чисто виртуальную функцию класса IObject. Реализовать в классе «Очередь» несколько конструкторов, в том числе конструктор копирования, методы для добавления элемента в очередь и извлечения элемента из очереди. Проиллюстрировать работу с классом «Очередь» для хранения различных типов объектов.

Вариант 12.

Реализовать класс-контейнер «Отображение с неуникальными ключами (Multimap)», состоящий из объектов (пары ключ-значение) произвольного типа (но тип ключей один и тот же для одного отображения, тип значений также один и тот же в рамках одного отображения, но может отличаться от типа ключей). Методы класса должны принимать указатель на абстрактный класс IObject. В классе IObject необходимо определить чисто виртуальные функции bool equal(IObject*) – для сравнения объектов на равенство и IObject* clone() – для создания копии объекта при добавлении в контейнер. Класс IObject должен быть базовым для классов различных типов ключей и значений, которые может содержать отображение (IntObject, CharObject, StringObject). Все эти классы должны переопределять чисто виртуальные функции класса IObject. Реализовать в классе «Отображение» несколько конструкторов, в том числе конструктор копирования, методы для добавления нового элемента в отображение, удаления элемента, поиска по ключу и вывода на экран. Элементы отображения хранить в

неупорядоченном массиве структур, поиск элементов осуществлять с помощью перебора. Проиллюстрировать работу с классом «Отображение» для хранения различных типов ключей и значений.

Вариант 13.

Реализовать класс-контейнер «Отображение с неуникальными ключами (Multimap)», состоящий из объектов (пары ключ-значение) произвольного типа (но тип ключей один и тот же для одного отображения, тип значений также один и тот же в рамках одного отображения, но может отличаться от типа ключей). Методы класса должны принимать указатель на абстрактный класс IObject. В классе IObject необходимо определить чисто виртуальные функции bool less(IObject*) – для сравнения объектов на меньше и IObject* clone() – для создания копии объекта при добавлении в контейнер. Класс IObject должен быть базовым для классов различных типов ключей и значений, которые может содержать отображение (IntObject, CharObject, StringObject). Все эти классы должны переопределять чисто виртуальные функции класса IObject. Реализовать в классе «Отображение» несколько конструкторов, в том числе конструктор копирования, методы для добавления нового элемента в отображение, удаления элемента, поиска по ключу и вывода на экран. Элементы отображения хранить в упорядоченном массиве структур, поиск элементов осуществлять с помощью бинарного поиска. Проиллюстрировать работу с классом «Отображение» для хранения различных типов ключей и значений.

Вариант 14.

Реализовать класс-контейнер «Односвязный список (List)», состоящий из объектов произвольного (но одного и того же) типа. Методы класса должны принимать указатель на абстрактный класс IObject. В классе IObject необходимо определить чисто виртуальные функции bool equal(IObject*) – для сравнения объектов на равенство и IObject* clone() – для создания копии объекта при добавлении в контейнер. Класс IObject должен быть базовым для классов различных типов объектов, которые может содержать список (DoubleObject, StringObject). Все эти классы должны переопределять чисто виртуальные функции класса IObject. Реализовать в классе «Список» несколько конструкторов, в том числе конструктор копирования, методы для поиска элемента, добавления элемента в список, удаления элемента из списка. Проиллюстрировать работу с классом «Список» для хранения различных типов объектов.

Вариант 15.

Реализовать класс-контейнер «Упорядоченное бинарное дерево (Tree)», состоящий из объектов произвольного (но одного и того же) типа. Методы класса должны принимать указатель на абстрактный класс IObject. В классе IObject необходимо определить чисто виртуальные функции bool less(IObject*) – для сравнения объектов на меньше и IObject* clone() – для создания копии объекта при добавлении в контейнер. Класс IObject должен быть базовым для классов различных типов объектов, которые может содержать дерево (CharObject, StringObject). Все эти классы должны переопределять чисто виртуальные функции класса IObject. Реализовать в классе «Дерево» несколько конструкторов, в том числе конструктор копирования, методы для добавления нового элемента, удаления элемента, поиска элемента и вывода на экран. Проиллюстрировать работу с классом «Дерево» для хранения различных типов элементов.

Вариант 16.

Реализовать класс-контейнер «Отображение (Map)», состоящий из объектов (пары ключ-значение) произвольного типа (но тип ключей один и тот же для одного отображения, тип значений также один и тот же в рамках одного отображения, но может отличаться от типа ключей). Методы класса должны принимать указатель на абстрактный класс IObject. В классе IObject необходимо определить чисто виртуальные функции bool equal(IObject*) – для сравнения объектов на равенство и IObject* clone() – для создания копии объекта при добавлении в контейнер. Класс IObject должен быть базовым для классов различных типов ключей и значений, которые может содержать отображение (CharObject, StringObject). Все эти классы должны переопределять чисто виртуальные функции класса IObject. Реализовать в классе «Отображение» несколько конструкторов, в том числе конструктор копирования, методы для добавления нового элемента в отображение, удаления элемента, поиска по ключу и вывода на экран. Элементы отображения хранить в неупорядоченном массиве структур, поиск элементов осуществлять с помощью перебора. Проиллюстрировать работу с классом «Отображение» для хранения различных типов ключей и значений.

Вариант 17.

Реализовать класс-контейнер «Множество (Set)», состоящее из объектов произвольного (но одного и того же) типа. Методы класса должны принимать указатель на абстрактный класс IObject. В классе IObject необходимо определить чисто виртуальные функции bool less(IObject*) – для сравнения объектов на меньше и IObject* clone() – для создания копии объекта при добавлении в контейнер. Класс IObject должен быть базовым для классов различных типов объектов, которые может содержать множество (DoubleObject, StringObject). Все эти классы должны переопределять чисто виртуальные функции класса IObject. Реализовать в классе «Множество»

несколько конструкторов, в том числе конструктор копирования, методы для определения принадлежности заданного элемента множеству, добавления, удаления элемента из множества, пересечения, объединения, разности двух множеств. Элементы множества хранить в упорядоченном массиве (содержащем значения типа IObject*), поиск элементов осуществлять на основе бинарного поиска. Продемонстрировать работу с классом «Множество» для хранения различных типов объектов.