

|  |
| --- |
| Министерство образования и науки Российской Федерации |
| Федеральное государственное бюджетное образовательное учреждение  высшего образования  **«МИРЭА – Российский технологический университет»**  **РТУ МИРЭА**  **филиал РТУ МИРЭА в г. Ставрополе** |

**МЕТОДИЧЕСКИЕ УКАЗАНИЯ**

к лабораторным работам и самостоятельной работе

по дисциплине «Объектно-ориентированное программирование»

для студентов направления подготовки 09.03.01 Информатика и вычислительная техника

Часть 1

Ставрополь

Методические указания составлены в соответствии с Федеральным государственным образовательным стандартом высшего образования и программой дисциплины «Объектно-ориентированное программирование» для студентов направления подготовки 09.03.01 «Информатика и вычислительная техника».

Составители: Н.Г. Касьяненко, к.т.н., доцент

Ю.В. Рокотов, к.т.н., доцент

СОДЕРЖАНИЕ

|  |  |
| --- | --- |
| Указания по технике безопасности………………………………………… | 5 |
| Лабораторная работа 1. Разработка консольных приложений в C#. Часть 1……………………………………………………………………….. | 6 |
| Лабораторная работа 2. Разработка консольных приложений в С#. Часть 2……………………………………………………………………….. | 11 |
| Лабораторная работа  3. Использование предопределенных типов данных, переменных, констант. Часть 1………………………………….. | 13 |
| Лабораторная работа  4. Использование предопределенных типов данных, переменных, констант. Часть 2…………………………………… | 19 |
| Лабораторная работа  5. Использование возможностей консольного ввода-вывода. Часть 1………………………………………………………. | 22 |
| Лабораторная работа  6. Использование возможностей консольного ввода-вывода. Часть 2………………………………………………………. | 26 |
| Лабораторная работа  7. Использование операторов ветвления if, switch. Часть 1……………………………………………………………………….. | 29 |
| Лабораторная работа  8. Использование операторов ветвления if, switch. Часть 2……………………………………………………………………….. | 33 |
| Лабораторная работа  9. Использование оператора цикла for. Часть 1….. | 43 |
| Лабораторная работа  10. Использование оператора цикла for. Часть 2… | 46 |
| Лабораторная работа  11. Использование оператора цикла while. Часть 1 | 50 |
| Лабораторная работа  12. Использование оператора цикла while. Часть 2 | 52 |
| Лабораторная работа  13. Использование оператора цикла do …while. Часть 1……………………………………………………………………….. | 55 |
| Лабораторная работа  14. Использование оператора цикла do …while. Часть 2……………………………………………………………………….. | 57 |
| Лабораторная работа  15. Использование классов и структур. Часть 1…. | 60 |
| Лабораторная работа  16. Использование классов и структур. Часть 2…. | 68 |
| Лабораторная работа  17. Конструктор класса. Перегрузка конструкторов класса. Часть 1……………………………………………… | 73 |
| Лабораторная работа  18. Конструктор класса. Перегрузка конструкторов класса. Часть 2……………………………………………… | 76 |
| Список рекомендуемой литературы……………………………………….. | 78 |
| Приложение 1…………………………………………………………… | 79 |

**Указания по технике безопасности**

Перед началом занятия необходимо:

1. Проверить правильность установки стола, стула, угол наклона экрана монитора, положение клавиатуры и, при необходимости, произвести корректировку их установки в целях исключения неудобных поз и длительных напряжений тела. Особо обратить внимание на то, что дисплей должен находиться на расстоянии не менее 50 см от глаз (оптимально 60 – 70 см), плоскость его экрана должна быть перпендикулярна направлению взгляда и центр экрана должен быть ниже уровня (или на уровне) глаз.
2. Подготовить к работе рабочее место, разместив необходимые принадлежности таким образом, чтобы исключить их падение и опрокидывание.
3. Проверить работу компьютера, обо всех замеченных нарушениях, неисправностях и поломках немедленно доложить преподавателю или лаборанту.

Запрещается приступать к работе в случае обнаружения несоответствия рабочего места установленным в данном разделе требованиям, а также при невозможности выполнить указанные в данном разделе подготовительные к работе действия.

Обо всех неполадках в работе оборудования необходимо ставить в известность преподавателя или лаборанта. Запрещается самостоятельное устранение любых неисправностей используемого оборудования.

При обнаружении неисправности в работе оборудования (нагревании, появлении искрения, запаха горелой изоляции, появлении посторонних звуков и т.п.) немедленно прекратить работу и сообщить об этом учителю (иному лицу, проводящему занятия) или лаборанту и действовать в соответствии с его указаниями.

По окончании занятия следует:

1. Произвести закрытие всех выполняемых задач.
2. Привести в порядок рабочее место.
3. При обнаружении неисправности мебели, оборудования проинформировать об этом преподавателя и лаборанта.
4. С разрешения преподавателя организованно покинуть кабинет.

**лабораторная работа 1**

**разработка консольных приложений в C#. Часть 1**

**Цель работы:** научиться разрабатывать консольные приложения в C# с организацией простейшего вывода.

Задачи лабораторной работы:

– изучить структуру консольного приложения C#;

– научиться реализовывать простейший вывод в консольных приложениях C#.

**1. Теоретическая часть**

Рассмотрим структуру консольного приложения на языке C#, созданного с использованием средств MS Visual Studio (рис. 1.1).

|  |
| --- |
| 4. Объявление метода Main  3. Объявление класса  2. Пользовательское пространство имен  1. Подключение пространств имен |
| Рисунок 1.1 – Исходный код консольного приложения |

В интерактивном редакторе кода среды разработки код может быть свернут/ развернут с использованием кнопок +/-, внедренных в код.

На рисунке 1.1 в области 1 показано подключение пространств имен (библиотек, содержащих стандартные инструменты) с использованием зарезервированного слова using.

Программист сам создает свое пространство имен с именем LR\_One (2), в котором объявляется класс с именем Program.

В классе Program объявлен один метод – функция Main (параметры функции не рассматриваем).

**2. Материалы и оборудование**

Для выполнения лабораторной работы необходимо использовать:

1) персональный компьютер;

2) программное обеспечение:

- операционную систему windows 7 и выше,

- Microsoft Visual Studio 2010 и выше.

**3. Методика и порядок выполнения работы**

1. Создайте консольное приложение, для этого выполните следующие действия:
   1. Выберите команду главного меню 
   2. В открывшемся диалоговом окне (рис. 1.2) выберите необходимые настройки для создаваемого проекта: язык Visual C#; фреймворк: .NET Framework 4; шаблон: Console Application.

|  |
| --- |
|  |
| Рисунок 1.2 – Создание нового проекта консольного приложения |

* 1. В текстовом поле Name введите имя проекта (например, LR\_One).
  2. В текстовом поле Location выберите место сохранения нового проекта.
  3. Установите флажок-переключатель «Create directory for solution».
  4. Нажмите кнопку «OK».

1. После выполнения пункта 1 в среде разработки откроется новый созданный проект (рис. 1.3).

|  |
| --- |
|  |
| Рисунок 1.3 – Новый проект, загруженный в среду разработки: вкладка «Solution Explorer» отображает состав проекта (нас интересует только файл Program.cs); в левой части окна (сверху) открыт файл Program.cs в редакторе кода |

1. На данном этапе необходимо ознакомиться со структурой исходного файла консольного приложения (рис. 1.4).

|  |
| --- |
|  |
| Рисунок 1.4 – Исходный файл консольного приложения |

1. Весь код программы необходимо писать внутри функции Main.
2. Для построения сборки (исполняемого exe-файла) выполните команду главного меню  (или использовать горячую клавишу ). После этого сборка создана, но приложение не будет запущено автоматически.
3. Для создания сборки и последующего запуска программы можно воспользоваться командой  главного меню среды разработки или нажать кнопку панели инструментов  . Можно также использовать горячую клавишу .
4. Запустите приложение на выполнение одним из методов, указанных в пункте 6. Окно консольного приложения появится и исчезнет. Это означает, что приложение выполнило все команды, написанные программистом, и завершило свою работу.
5. Для удержания окна на экране измените исходный файл в соответствии с рисунком 1.5. В функции Main добавлен вызов только одной команды Console.ReadKey() – эта функция останавливает выполнение программы и ждет, когда пользователь нажмет любую клавишу.

|  |
| --- |
|  |
| Рисунок 1.5 – Исходный файл консольного приложения для предотвращения закрытия окна консольного приложения |

1. Запустите измененное приложение, убедитесь, что окно удерживается на экране.
2. Добавьте несколько строк кода в исходный файл (рис. 1.6).

|  |
| --- |
|  |
| Рисунок 1.6 – Исходный файл консольного приложения для вывода информации на экран |

1. Внимательно изучите исходный код примера на рисунке 1.6. Запустите приложение и убедитесь, что отсутствуют ошибки и информация выводится.

**4. Задания к лабораторной работе**

1. Изучите теоретическую часть к лабораторной работе, изложенную в методических указаниях.
2. Выполните задания 1-11, сформулированные в разделе **3. Методика и порядок выполнения работы** методических указаний.
3. Оформите отчет по лабораторной работе (Приложение 1).

**5. Задания для самостоятельной работы**

1. Опишите структуру консольного приложения C#. Какая функция имеет особенное значение при выполнении программы на языках C, C++, C#?

2. Поясните термин «точка входа» программы.

3. Опишите возможные варианты предотвращения закрытия окна консольного приложения.

4. Опишите метод для вывода текста, использованный в программе.

**Вопросы к лабораторной работе**

1. Как создать консольное приложение в Microsoft Visual Studio?

2. Что представляет собой команда console.readkey()?

**лабораторная работа 2**

**разработка консольных приложений в C#. Часть 2**

**Цель работы:** научиться работать с переменными и константами простых типов в C#.

Задачи лабораторной работы:

– научиться объявлять переменные простых типов в языке C#;

– научиться объявлять константы простых типов в языке C#;

– научиться выполнять простейшие действия с переменными и константами.

**1. Теоретическая часть**

Функция Main имеет особенное значение в программировании на языках C, C++ и C#.

Функцию Main называют «точкой входа», то есть началом выполнения программы. Далее мы рассмотрим приложения, содержащие множество функций. Операционная система знает, с какой именно функции начать выполнение программы – с функции Main. Очевидно, что имя этой функции менять нельзя. Это должен быть статический метод класса (или структуры), возвращающий либо значение типа int, либо void. Хотя нередко модификатор public указывается явно, поскольку по определению этот метод должен быть вызван извне программы, на самом деле неважно, какой уровень доступа вы назначите методу точки входа. Он запустится, даже если вы пометите его как private.

Когда компилируется консольное или Windows-приложение С#, по умолчанию компилятор ищет в точности один метод Main() с описанной выше сигнатурой в любом классе и делает его точкой входа программы. Если существует более одного метода Main(), компилятор возвращает сообщение об ошибке.

Обратите внимание на вложенность конструкций на рисунке 1.1 лабораторной работы 1: в пространство имен LR\_One вложен класс Program, в класс Program вложена функция Main. В свою очередь, в функции Main содержатся инструкции на языке C#-код, который начнет выполняться при старте программы.

В С#, как и в других С-подобных языках, большинство операторов завершаются точкой с запятой (;) и могут продолжаться в нескольких строках без необходимости указания знака переноса. Операторы могут быть объединены в блоки с помощью фигурных скобок ({ }). Однострочные комментарии начинаются с двойного слеша (//), а многострочные – со слеша со звездочкой (/\*) и заканчиваются противоположной комбинацией (\*/). В этих аспектах язык С# идентичен C++ и Java.

Следует также помнить о том, что язык С# чувствителен к регистру символов. Это значит, что переменные с именами myVar и MyVar являются разными.

Причина присутствия оператора using в файле Program.cs связана с использованием библиотечных классов.

Весь код С# должен содержаться внутри класса. Объявление класса состоит из ключевого слова class, за которым следует имя класса и пара фигурных скобок. Весь код, ассоциированный с этим классом, размещается между этими скобками.

**2. Задания к лабораторной работе**

1. Изучите теоретическую часть к лабораторной работе, изложенную в методических указаниях.
2. Выполните задание: измените приложение, созданное в ходе выполнения предыдущей лабораторной работы, таким образом, чтобы программа выводила на экран следующую информацию (каждый студент должен использовать персональную информацию о себе):

– название и номер лабораторной работы;

– ФИО студента;

– группа студента и шифр специальности;

– дата рождения студента;

– населенный пункт постоянного места жительства студента;

– любимый предмет в школе;

– краткое описание увлечений, хобби, интересов.

1. Оформите отчет по лабораторной работе (Приложение 1).

**4. Задания для самостоятельной работы**

1. Поясните термины «пространства имен», «класс», «метод», «функция».

2. Опишите отличия структуры консольного приложения от приложения, построенного с использованием технологий Windows Forms.

3. Опишите вложенность конструкций консольного приложения.

**Вопросы к лабораторной работе**

1. Какой тип данных использован для выводимой информации?

2. Как обратиться к методу-члену класса?

**Лабораторная работа 3**

**использование Предопределенных типов данных, переменных, констант. Часть 1**

**Цель работы**: научиться работать с переменными и константами простых типов в C#.

Задачи лабораторной работы:

– научиться объявлять переменные и константы простых типов в языке C#;

– научиться выполнять простейшие действия с переменными и константами.

**1. Теоретическая часть**

Перед выполнением заданий необходимо изучить материалы лекций. Следует понимать принцип деления типов .NET на типы значений и ссылочные типы.

На данной лабораторной работе необходимо освоить приемы работы с предопределенными типами значений.

**1.1 Типы значений C#**

Язык C# поддерживает восемь предопределенных целочисленных типов (таблица 3.1).

Таблица 3.1 – Целочисленные типы C#

|  |  |  |  |
| --- | --- | --- | --- |
| Имя типа | Тип CTS | Описание | Диапазон (минимум : максимум) |
| sbyte | System.SByte | 8-битное целое со знаком | -128 : 127 |
| short | System.Int16 | 16-битное целое со знаком | -32 768 : 32 767 |
| int | System.Int32 | 32-битное целое со знаком | -2 147 483 648: 2 147 483 647 |
| long | System.Int64 | 64-битное целое со знаком | -263 : 263-1 |
| byte | System.Byte | 8-битное целое без знака | 0 : 255 |
| ushort | System.UInt16 | 16-битное целое без знака | 0 : 65 535 |
| uint | System.UInt32 | 32-битное целое без знака | 0 : 232-1 |
| ulong | System.UInt64 | 64-битное целое без знака | 0 : 264-1 |

Язык C# также поддерживает и типы с плавающей точкой (таблица 3.2).

Таблица 3.2 – Типы с плавающей точкой C#

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Имя типа | Тип CTS | Описание | Кол-во знаков | Диапазон (минимум : максимум) |
| float | System.Single | 32-битное с плавающей точкой одинарной точности | 7 | от  до |
| double | System.Double | 64-битное с плавающей точкой двойной точности | 15/16 | от  до |

В таблице 3.3 представлен десятичный тип C#. Данный тип реализован для финансовых операций.

Таблица 3.3 – Десятичный тип C#

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Имя типа | Тип CTS | Описание | Кол-во знаков | Диапазон (минимум : максимум) |
| decimal | System.Decimal | 128-битное с плавающей точкой в десятичной нотации с высокой точностью | 28 | от  до |

Как и во многих языках программирования, существует булевский тип (таблица 3.4).

Таблица 3.4 – Булевский тип

|  |  |  |
| --- | --- | --- |
| Имя типа | Тип CTS | Значения |
| bool | System.Boolean | true или false |

Для хранения одиночных символов в языке C# используется тип char (таблица 3.5).

Таблица 3.5 – Булевский тип

|  |  |  |
| --- | --- | --- |
| Имя типа | Тип CTS | Значения |
| char | System.Char | Представляет отдельный 16-битный (Unicode) символ |

Литералы типа char записываются как одиночные, заключенные в одинарные кавычки символы: ‘F’, ‘w’, ‘ц’, ‘Я’ и т.д.

В переменных типа char можно хранить и специальные символы в виде управляющих последовательностей (таблица 3.6).

Таблица 3.6 – Представление символов в виде управляющих последовательностей.

|  |  |
| --- | --- |
| Управляющая последовательность | Символ |
| \’ | Одиночная кавычка |
| \” | Двойная кавычка |
| \\ | Обратный слэш |
| \0 | Пусто |
| \a | Предупреждение (звуковой сигнал) |
| \b | Забой |
| \f | Подача формы |
| \n | Новая строка |
| \r | Возврат каретки |
| \t | Символ табуляции |
| \v | Вертикальная табуляция |

Если отдельные символы объединены в строку, то необходимо использовать тип string, который отображается на тип CTS – System.String.

**1.2 Объявление и инициализация переменных в C#**

Синтаксис объявления переменных в C# выглядит следующим образом:

Тип\_данных Идентификатор\_переменной

Например:

**int a;**

Этот код объявляет переменную типа int с именем a. Компилятор не позволит использовать эту переменную до тех пор, пока она не будет инициализирована (т.е. пока ей не будет присвоено значение).

Для инициализации переменной a нужно написать следующий код:



Переменную можно инициализировать во время объявления:



или



Синтаксис C# позволяет объявить несколько переменных (и инициализировать их) одного типа в одной синтаксической конструкции.

Например:



В данном примере объявляется пять переменных типа float, некоторые из них инициализируются в процессе объявления.

**1.3 Объявление и инициализация констант в C#**

Константа – это переменная, значение которой не меняется за время выполнения программы. Для объявления константы необходимо воспользоваться ключевым словом const. Например:



Очевидно, что при таком объявлении, поменять значения simv и pi в дальнейшем будет нельзя.

**2. Материалы и оборудование**

Для выполнения лабораторной работы необходимо использовать:

1) персональный компьютер;

2) программное обеспечение:

- операционную систему windows 7 и выше,

- Microsoft Visual Studio 2010 и выше.

**3. Методика и порядок выполнения работы**

1. Создайте консольное приложение в соответствии с алгоритмом, описанном в лабораторной работе 1.

2. Изучите материал раздела «Теоретическая часть» данной лабораторной работы.

3. Модифицируйте исходный файл как показано на рисунке 3.1.

|  |
| --- |
|  |
| Рисунок 3.1 – Объявление переменных в C# |

4. Рассмотрим приведенный пример подробнее:

4.1. В строке 12 объявляется переменная типа int с именем a.

4.2. В строке 13 объявляется переменная типа int с именем b, причем при объявлении для нее устанавливается начальное значение, равное 7.

4.3. В строке 15 объявляется переменная типа string с именем str и инициализируется строковым значением «Hello, World!!!».

4.4. В строке 17 переменной a присваивается целочисленное значение 123.

4.5. Запустите приложение на выполнение. У вас должно появиться пустое окно консольного приложения. Очевидно, что исходный код, представленный на рисунке 3.1, не предполагает вывода какой-либо информации на экран.

4.6. Для вывода информации на экран воспользуйтесь функцией Console.WriteLine, изученной на лабораторной работе 1. Добавьте в исходный файл следующие строки (строки 19 – 21 на рис. 3.2).

|  |
| --- |
|  |
| Рисунок 3.2 – Добавление строк кода для вывода значений объявленных переменных на экран |

4.7. Выполните программу.

**4. Задания к лабораторной работе**

1. Изучите теоретическую часть к лабораторной работе, изложенную в методических указаниях.
2. Выполните задания 1-4, изложенные в разделе **3. Методика и порядок выполнения работы** методических указаний.
3. Оформите отчет по лабораторной работе (Приложение 1).

**5. Задания для самостоятельной работы**

1. Определите понятие переменной. Опишите синтаксис объявления переменной.

2. Опишите синтаксис объявления константы и ее отличия от переменной.

3. Опишите разнообразие и особенности типов данных, применяемых в программах на языке C#.

**Вопросы к лабораторной работе**

1. Какая функция использована в программах для вывода информации на экран? Как выглядит обращение к ней и почему?

2. Что такое константа?

3. Как объявить и инициализировать константу?

**Лабораторная работа 4**

**использование Предопределенных типов данных, переменных, констант. Часть 2**

**Цель работы**: научиться работать с переменными и константами простых типов в C#.

Задачи лабораторной работы:

– научиться объявлять переменные и константы простых типов в языке C#;

– научиться выполнять простейшие действия с переменными и константами.

**1. Теоретическая часть**

В примере на рисунке 3.2 (ЛР 3) используется простой вывод, то есть имя переменной просто передается в качестве параметра функции Console.WriteLine.

Для выполнения форматного вывода (изменения формата представления выводимой информации) необходимо реализовать вывод в следующем виде (рис. 4.1: изменены строки 19-21).

|  |
| --- |
|  |
| Рисунок 4.1 – Вывод информации с использованием форматных строк |

Внимательно изучите код полученной программы, затем выполните ее. Поясните использование формата при выводе.

**2. Задания к лабораторной работе**

1. Изучите теоретическую часть к лабораторной работе, изложенную в методических указаниях.
2. Выполните задание, сформулированное в теоретической части.
3. Напишите программу, объявив требуемые переменные; присвойте им начальные значения (определите самостоятельно, значения какого типа могут принимать переменные), выведите на экран с использованием форматной строки значения переменных и результат вычисления выражения в соответствие с вариантом (1-25):

|  |  |
| --- | --- |
| Вариант | Выражение для вычисления |
| 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 |  |

1. Оформите отчет по лабораторной работе (Приложение 1).

**3. Задания для самостоятельной работы**

1. Проанализируйте, чем тип char отличается от типа string.

2. Опишите инициализацию переменных и констант. В чем разница между ними?

3. Опишите особенности применения при выводе строковых литералов, представляющих собой управляющие последовательности.

**Вопросы к лабораторной работе**

1. Как выполнить форматный вывод информации?

2. Опишите синтаксис и возможности форматирования выводимых данных.

3. Какие действия нужно выполнить для вычисления арифметического выражения в программе?

**Лабораторная работа 5**

**Использование возможностей консольного ввода-вывода. Часть 1**

**Цель работы**: изучить команды ввода и вывода данных в консольном приложении.

Задачи лабораторной работы:

– научиться применять функции Console.Write и Console.WriteLine;

– научиться применять функции Console.Read и Console.ReadLine;

– научиться использовать форматы вывода для различных типов данных.

**1. Теоретическая часть**

На предыдущих лабораторных работах уже использовалась команда Console.WriteLine. Рассмотрим ее синтаксис подробнее:



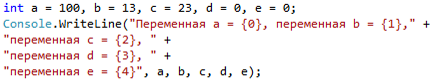
Эта запись означает вызов статического метода класса Console. В качестве параметра функции передается строка, которая выводится в консоль.

Существует также функция:



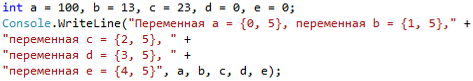
Этот метод также осуществляет вывод, только не переводит каретку на следующую строку.

Рассмотренные методы также позволяют осуществлять форматный вывод, аналогичный функции printf языка С, например:



В данном примере в качестве первого параметра передается строка, содержащая маркеры в фигурных скобках. При выводе на места маркеров будут подставлены параметры, которые следуют за строкой.

При форматном выводе можно задавать ширину поля вывода, для этого необходимо воспользоваться следующим приемом:



В данном примере видно: чтобы задать ширину вывода, необходимо использовать формат

{n, w},

где n – порядковый номер параметра,

w – ширина области вывода (знак w позволяет осуществлять выравнивание выводимого значения по левому или правому краю).

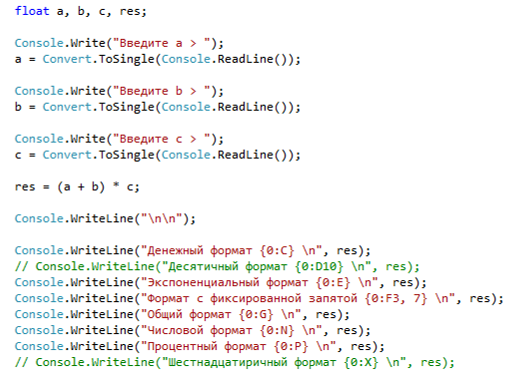
При выводе также можно добавлять строку формата вместе с необязательным значением точности (табл. 5.1).

Таблица 5.1 – Основные строки формата

|  |  |
| --- | --- |
| Строка | Описание |
| C | Локальный формат валюты |
| D | Десятичный формат. Преобразует целое к основанию 10 и снабжает ведущими нулями, если есть спецификатор точности. |
| E | Научный (экспоненциальный) формат. Спецификатор точности устанавливает количество десятичных разрядов (по умолчанию 6). |
| F | Формат с фиксированной запятой. Спецификатор точности задает количество десятичных разрядов. |
| G | Общий формат. Форматирование E или F. |
| N | Числовой формат. Форматирует число с разделителями тысяч – запятыми. |
| P | Процентный формат |
| X | Шестнадцатеричный формат. Спецификатор точности используется для указания ведущих нулей. |

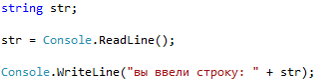
Приведем пример 1 использования формата.

Пример 1. Использование формата



Изучите представленный пример. Выполните данный код в среде выполнения.

В классе Console также существуют методы, позволяющие считывать информацию, вводимую пользователем:



В данном примере строка, введенная пользователем, считывается в переменную str, затее осуществляется ее вывод.

Наберите данный пример в среде VS, изучите код.

Выполните программу.

**2. Материалы и оборудование**

Для выполнения лабораторной работы необходимо использовать:

1) персональный компьютер;

2) программное обеспечение:

- операционную систему windows 7 и выше,

- Microsoft Visual Studio 2010 и выше.

**3. Методика и порядок выполнения работы**

1. Создайте консольное приложение в соответствии с алгоритмом, представленным в лабораторной работе 1.
2. Выполните задание примера 1 (Теоретическая часть), выполнив вывод результата с использованием всех возможных форматов, представленных в таблице 5.1.

**4. Задания к лабораторной работе**

1. Изучите теоретическую часть к лабораторной работе, изложенную в методических указаниях.
2. Ответьте на вопросы для самостоятельной работы, используя лекции и указанную в них литературу, теоретический материал к лабораторной работе.
3. Выполните задания, изложенные в разделе **3. Методика и порядок выполнения работы** методических указаний.
4. Оформите отчет по лабораторной работе (Приложение 1).

**5. Задания для самостоятельной работы**

1. Опишите варианты оформления комментариев к программе.
2. Проанализируйте возможности функций, используемых для вывода информации в консоль.
3. Проанализируйте возможности функций, используемых для считывания информации с консоли.

**Вопросы к лабораторной работе**

1. Как задать ширину области вывода?

2. Какие существуют строки формата?

3. Какую роль играют маркеры в осуществлении форматного вывода?

**Лабораторная работа 6**

**Использование возможностей консольного ввода-вывода. Часть 2**

**Цель работы**: изучить команды ввода и вывода данных в консольном приложении.

Задачи лабораторной работы:

– научиться применять функции Console.Write и Console.WriteLine;

– научиться применять функции Console.Read и Console.ReadLine;

– научиться использовать форматы вывода для различных типов данных.

**1. Теоретическая часть**

МетодConsole.WriteLine позволяет реализовать форматный вывод, позволяющий:

- подставить параметры, имена которых включены в список после строки, на места маркеров в фигурных скобках в строке, передаваемой в качестве первого параметра;

- задать ширину вывода переменной при использовании формата:

{n, w},

где n – порядковый номер параметра,

w – ширина области вывода;

- задать тип строки формата:

C – локальный формат валюты;

D – десятичный формат, преобразующий целое к основанию 10 (снабжает ведущими нулями, если есть спецификатор точности);

E – научный (экспоненциальный) формат; спецификатор точности устанавливает количество десятичных разрядов (по умолчанию 6);

F – формат с фиксированной запятой; спецификатор точности задает количество десятичных разрядов;

G – общий формат; форматирование E или F;

N – числовой формат; форматирует число с разделителями тысяч – запятыми.

P – процентный формат;

X – шестнадцатеричный формат; спецификатор точности используется для указания ведущих нулей.

**2. Материалы и оборудование**

Для выполнения лабораторной работы необходимо использовать:

1) персональный компьютер; 2) программное обеспечение: операционную систему windows, Microsoft Visual Studio 2010 и выше.

**3. Задания к лабораторной работе**

1. Изучите теоретическую часть к лабораторной работе, изложенную в методических указаниях.
2. Выполните задание: объявите требуемые переменные, значения переменным пользователь должен присваивать в процессе выполнения программы (считываются с консоли), выведите на экран с использованием форматной строки значения переменных и результат вычисления выражения во всех возможных форматах для варианта выражения (1-25):

|  |  |
| --- | --- |
| **Вариант** | **Выражение для вычисления** |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

1. Оформите отчет по лабораторной работе (Приложение 1).

**4. Задания для самостоятельной работы**

1. Опишите известные вам форматы вывода.
2. Сравните статические методы для конвертирования значений и приведения их к требуемому типу. Какой класс их содержит?
3. Опишите возможности использования статических методов – математических функций. Какой класс их содержит?

**Вопросы к лабораторной работе**

1. Как задать тип строки формата?
2. Какой методпозволяет реализовать форматный вывод информации?
3. Сколько параметров нужно передать методу console.writeline и с какой целью?

**Лабораторная работа 7**

**использование операторов ветвления if, switch. Часть 1**

**Цель работы:** изучить операторы, позволяющие организовывать непоследовательное выполнение программного кода.

Задачи лабораторной работы:

– научиться применять условныq оператор if;

– научиться применять операторы выбора switch.

**1. Теоретическая часть**

**1.1 Условный оператор**

На предыдущих лабораторных занятиях были рассмотрены вопросы программирования на языке C# с использованием только последовательного выполнения операторов программы.

В языке C# (как и в большинстве языков) можно вводить условия, циклы, ветвления.

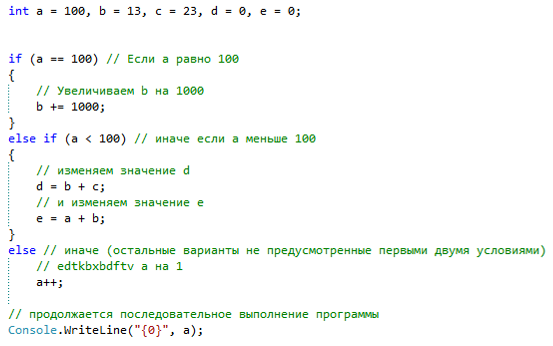
Условный оператор в С# позволяет организовать условие типа «если …, то …, иначе …».

Синтаксис условного оператора:



В данном синтаксисе *оператор* может быть составным оператором (группа операторов языка, заключенные в фигурные скобки). Условный оператор содержит только одно обязательное зарезервированное слово – if. Все остальные конструкции не являются обязательными (в синтаксическом описании на это указывают скобки [ ]).

**Пример 1** использования условного оператора:



В представленном примере сначала проверяется, является ли значение переменной a равным 100.

1. Если «Да», то выполняется единственный оператор (увеличение b на 1000). На этом выполнение всего условного оператора завершено и программа переходит к выполнению операторов, следующих за ним, то есть к выводу значения a.

2. Если первое условие не выполнилось, проверяется второе – является ли a меньше 100 (конструкция else if). Если это утверждение истинно, то выполняется составной оператор, состоящий из двух – изменение значений переменных d и e. Затем осуществляется переход к выводу переменной a.

3. Если не выполнилось ни одно условие с оператором if, то выполняется оператор, указанный после зарезервированного слова else. В данном случае это несоставной оператор (a увеличивается на 1), поэтому фигурные скобки можно не писать.

Во всей этой конструкции только оператор if является обязательным. Конструкций else if может быть любое количество, а if и else – только по одному.

**1.2 Оператор выбора**

Синтаксис оператора:

switch (*перем*)

{case *константа\_*1: *оператор\_*1; break;

case *константа\_*2: *оператор\_*2; break;

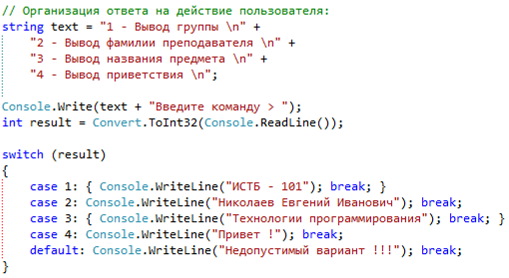
. . .

default: *оператор\_*n; break;

}

В данном операторе осуществляется выбор оператора (который может быть составным) в зависимости от значения переменной *перем*. Если *перем*  равна значению *константа\_*1, то выполнится *оператор\_*1, если *константа\_*2 – *оператор*\_2, и т.д. Если ни одно значение, указанное в каком-либо операторе case, не совпало со значением *перем*, то выполнится оператор, указанный в секции default. Внутри каждой секции case следует указать оператор break, который предотвращает проверку других условий после выполнения данного оператора. Следует обратить внимание, что в секциях case следует использовать только константы (переменные не допускаются).

**Пример 2** использования оператора switch … case:



Изучите представленный пример самостоятельно.

**2. Материалы и оборудование**

Для выполнения лабораторной работы необходимо использовать:

1) персональный компьютер;

2) программное обеспечение:

- операционную систему windows 7 и выше,

- Microsoft Visual Studio 2010 и выше.

**3. Задания к лабораторной работе**

1. Создайте консольное приложение в соответствии с алгоритмом, представленным в лабораторной работе 1.
2. Напишите программы примеров 1 и 2, представленных в теоретической части методических указаний, откомпилируйте и выполните. Сформулируйте выводы.
3. Оформите отчет по лабораторной работе (Приложение 1).

**4. Задания для самостоятельной работы**

1. Проанализируйте возможности С#-программ, позволяющие организовывать непоследовательное выполнение программного кода, т.е. переходы.

2. Опишите синтаксис оператора выбора switch. Поясните, почему следует использовать оператор break внутри каждого выбора.

3. Опишите синтаксис условного оператора. Какие части данного оператора являются обязательными?

**Вопросы к лабораторной работе**

1. Каким может быть тип выражения, стоящего в скобках после ключевого слова if в условном операторе?
2. Как определить в программе правильность использования вложенных условных операторов, включающих пары if..else?
3. Каким может быть тип выражения, стоящего в скобках после ключевого слова switch в переключателе?

**Лабораторная работа 8**

**использование операторов ветвления if, switch.**

**Часть 2**

**Цель работы:** изучить операторы, позволяющие организовывать непоследовательное выполнение программного кода, а также принципов отладки и тестирования и их применение к программам.

Задачи лабораторной работы:

– научиться применять условный оператор if;

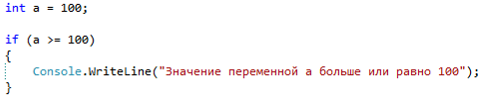
– научиться применять операторы выбора switch;

– научиться тестировать и отлаживать программы.

**1. Теоретическая часть**

**1.1**. **Применение условного оператора**. Итак, одной из возможностей создания переходов в С#-программах является использование условного оператора (синтаксис оператора if приведен в Теоретической части лабораторной работы 7).

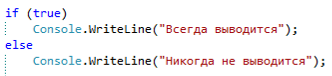
Приведем еще один пример использования условного оператора:



В данном случае вывод в консоль выполнится, только если a больше либо равно 100. Иначе условный оператор ничего не выведет. Фигурные скобки в данном примере можно опустить.

Обратите внимание, что для проверки на равенство используется оператор ==, а не =. Знак «равно» используется в C# для присваивания значений. Следует понимать, что условное выражение, стоящее в конструкции if, должно возвращать булево значение.

Например, следующий условный оператор всегда будет выполняться:



**1.2. Тестирование и отладка программ**

При использовании любой известной модели жизненного цикла ПП роль процессов тестирования и отладки невозможно переоценить.

При использовании *каскадной модели*жизненного цикла программного продукта (ПП) процессы программирования, тестирования, отладки программ происходят во время осуществления *этапа* ***реализации***.

Только ПП, многократно прошедшие тестирование с последующей отладкой на этапе реализации программного комплекса, допускаются к внедрению в *эксплуатацию*.

Тестирование и отладка важны также на *этапе* ***сопровождения***ПП, особенно при модификации программ по требованию заказчика, что может происходить:

- при внесении *дополнительных функций* в готовый ПП,

- при *обнаружении ошибок*, не выявленных на предыдущих этапах проведенного тестирования.

При использовании *спиральной модели*итерационный способ разработки ПП предполагает активное использование тестирования и отладки программ, причем приступить к этим процессам можно *значительно раньше*, чем при использовании каскадного подхода.

*Раннее и постоянное тестирование с последующей отладкой* – одна из основных черт и преимуществ спиральной модели проектирования и реализации сложных программных продуктов.

Как только готов *первый работоспособный прототип* ПП (а это при спиральной модели происходит очень быстро, т.к. для этого подхода характерно *раннее прототипирование* ПП), его *уже можно подвергнуть тестированию*.

Причем в процессе тестирования может принимать участие *заказчик*, согласование с которым происходит на ранних этапах жизненного цикла ПП.

При ***развитии функциональности*** разработки *роль постоянного тестирования еще возрастает*.

Приращение функциональных возможностей ПП не должно приводить:

- к снижению его качественных характеристик,

- корректности выполняемых программой функций.

Тестирование является одним из этапов ЖЦ ПП, направленным на повышение качественных характеристик.

При создании типичного ПП ***от 30% до 60%*** *общей трудоемкости отводится на тестирование*.

Применительно к ПП, *тестирование* – это процесс многократного выполнения программы с целью обнаружения ошибок.

*Цель* тестирования – выявить как можно больше ошибок.

*Принципы тестирования*. Из правильного определения тестирования вытекает ряд *принципов*.

1. *Процесс тестирования более эффективен, если его проводит не автор*.

2.*Описание предполагаемых значений результатов тестовых прогонов должно быть необходимой частью тестового набора данных***.**

Тестовый прогон должен содержать два компонента:

- описание *входных данных*;

- описание *точного и корректного результата*, соответствующего набору входных данных.

3.*Необходимо досконально изучать результаты применения каждого теста.*

4.*Тесты для неправильных и непредусмотренных входных данных должны разрабатываться так же тщательно, как и для правильных, предусмотренных.*

5. *Необходимо проверять не только, делает ли программа то, для чего она предназначена, но не делает ли она то, что не должна делать*.

6. *Вероятность наличия не обнаруженных ошибок в части программы пропорциональна числу ошибок, уже обнаруженных в программе*.

*Методы тестирования программ*.Тестирование программ является одной из составных частей более общего понятия «отладка программ».

Если тестирование – это процесс, направленный на выявление ошибок, то *целью отладки* являются *локализация и исправление**выявленных в процессе тестирования ошибок*.

 Под *отладкой* понимается *процесс, позволяющий получить программу, функционирующую с требующимися характеристиками в заданной области изменения входных данных.* Процесс отладки *включает*:

- действия, направленные на выявление ошибок (тестирование);

- диагностику и локализацию ошибок (определение характера и местонахождение ошибок);

- внесение исправлений в программу с целью устранения ошибок.

*Тестирование ПО*– важный этап жизненного цикла ПО, определяющее качество и надежность системы.

*Тестирование как* процесс выполнения программ с намерением найти ошибки включает в себя следующие ***этапы***:

- автономное тестирование;

- тестирование сопряжений;

- тестирование функций;

- комплексное тестирование;

- тестирование полноты и корректности документации;

- тестирование конфигураций.

 Надежность ПО повышается также с помощью применения различных *методов тестирования*.

Полное тестирование ПО невозможно.

Обычно применяют следующие *виды тестирования*:

- тестирование *ветвей*;

- математическое *доказательство правильности* алгоритма решения задачи (в некоторых работах именно в этом смысле употребляется слово верификация);

- *символическое* тестирование (или с помощью специально подобранных тестовых наборов), еще называется *статическим* тестированием. Удобно при локализации ошибки, проявление которой выявлено при конкретном узком или строго заданном диапазоне входных значений;

- *динамическое* тестирование (с помощью динамически генерируемых входных данных), что удобно при быстром тестировании во всем широком диапазоне входных параметров;

- тестирование *путей выполнения* программы;

- *функциональное* тестирование;

- *проверки по времени* выполнения программы;

- проверка *по использованию ресурсов;*

- *стрессовое* тестирование.

Тестирование может состоять из *трех отдельных фаз*:

- системный тест или лабораторные испытания – *альфа-тест*;

- опытная эксплуатация – *бета-тестирование*;

- приемочный тест.

**2. Материалы и оборудование**

Для выполнения лабораторной работы необходимо использовать:

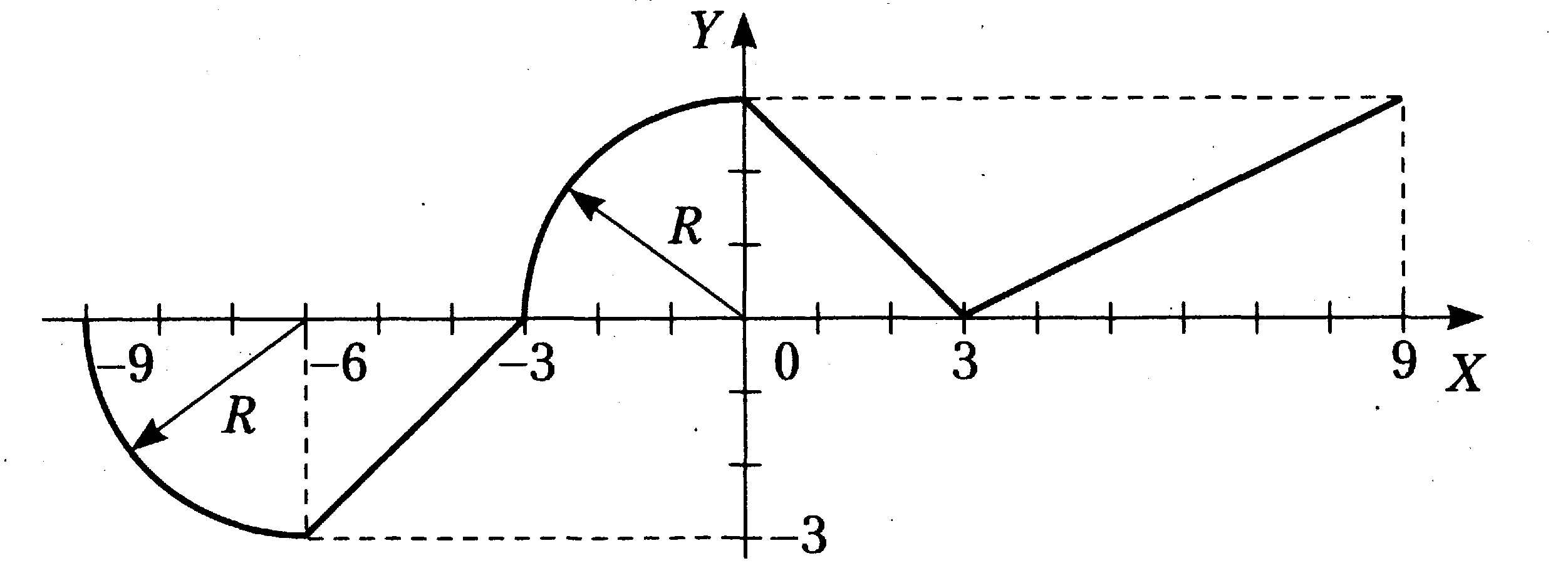
1) персональный компьютер;

2) программное обеспечение: операционную систему windows 7 и выше, Microsoft Visual Studio 2010 и выше.

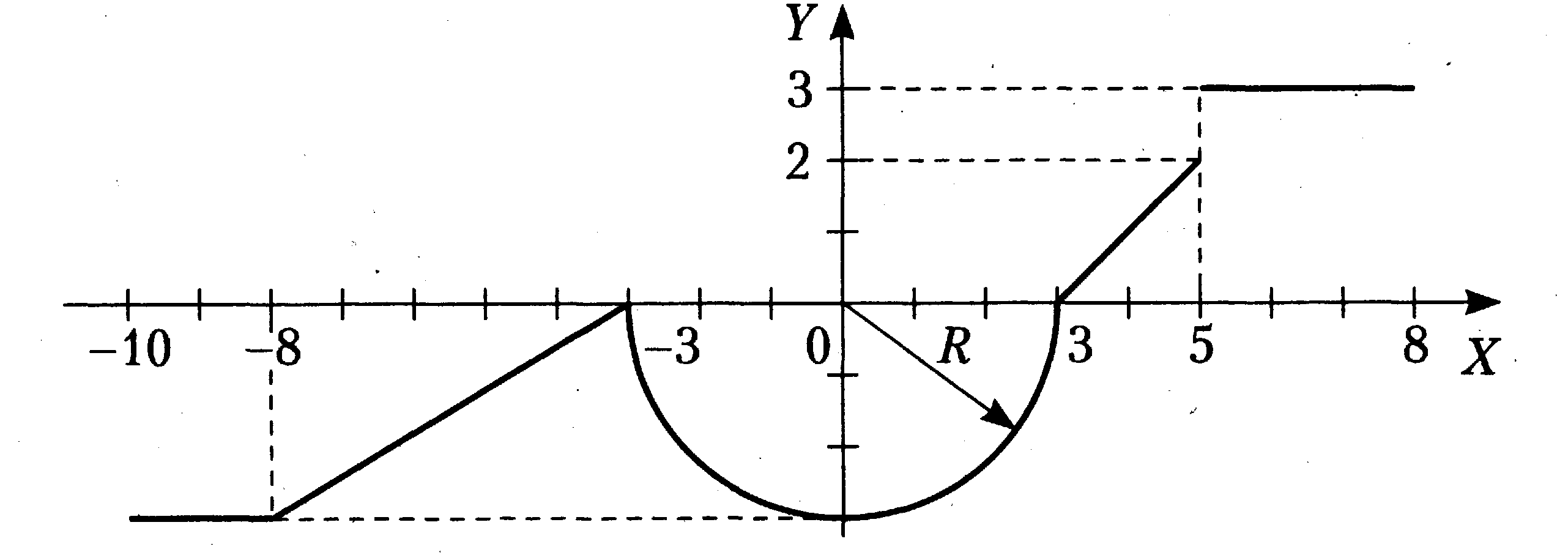
**3. Задания к лабораторной работе**

1. Изучите теоретическую часть к лабораторной работе, изложенную в методических указаниях.
2. Выполните задание согласно индивидуальному варианту (1-20): напишите программу, которая по введенному значению аргумента вычисляет значение функции, заданной в виде графика. Параметр R вводится с клавиатуры.

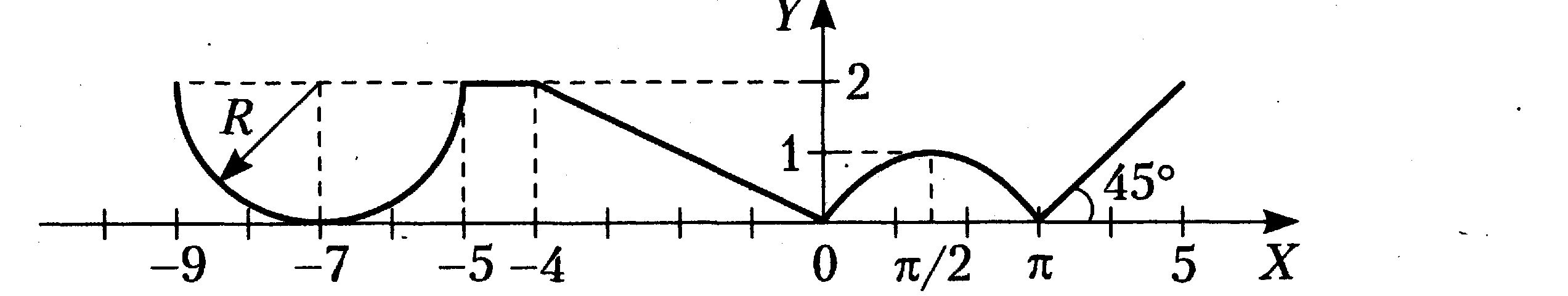
**Вариант 1**



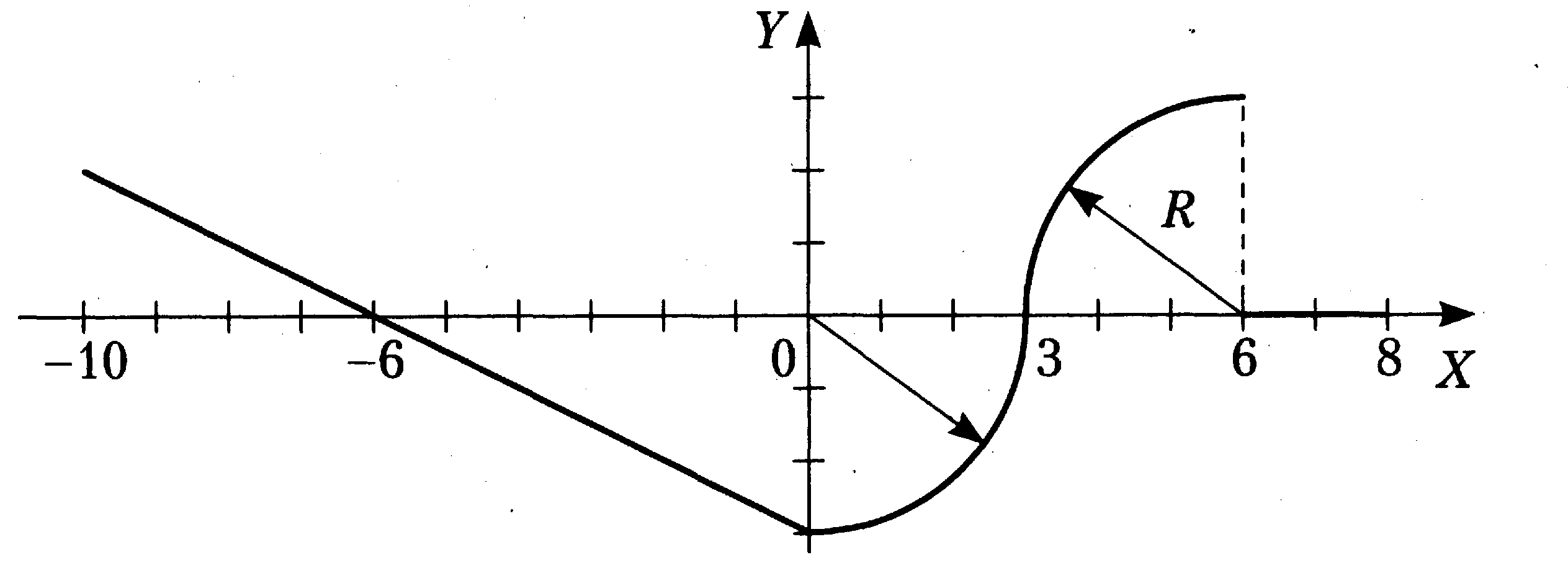
**Вариант 2**



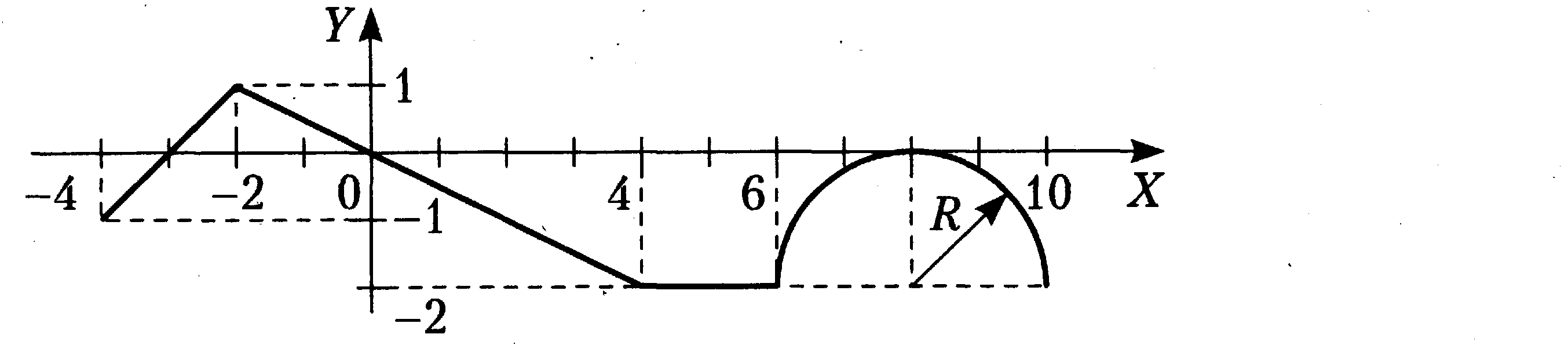
**Вариант 3**



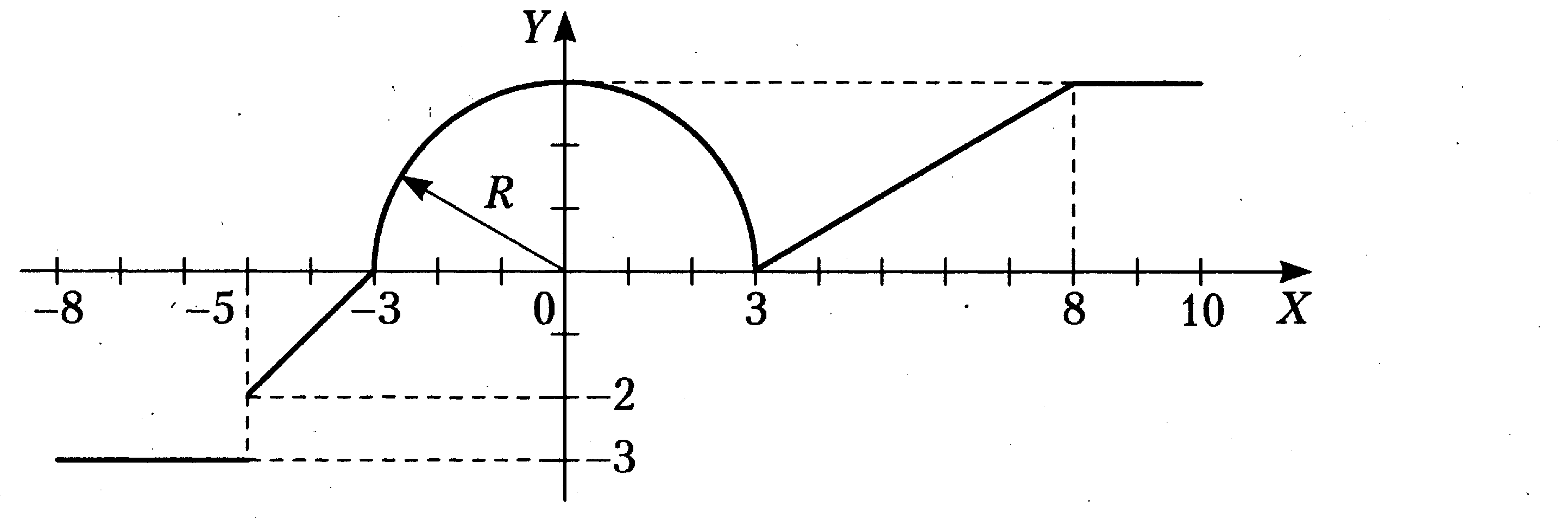
**Вариант 4**



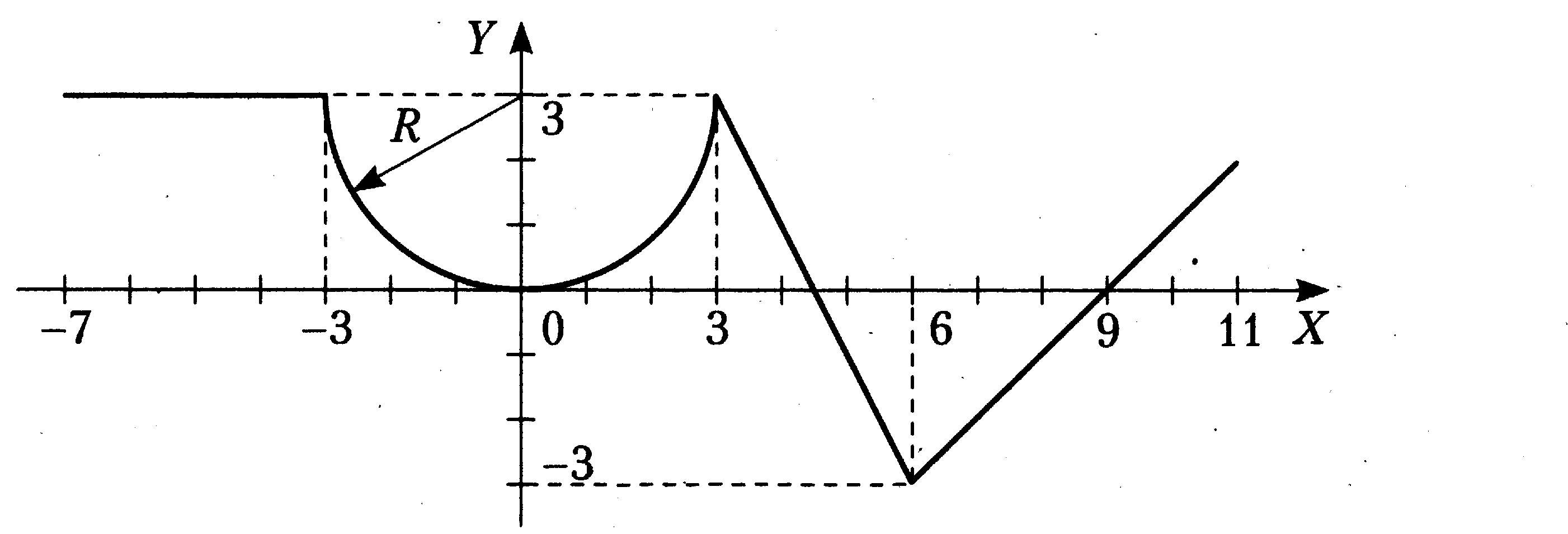
**Вариант 5**



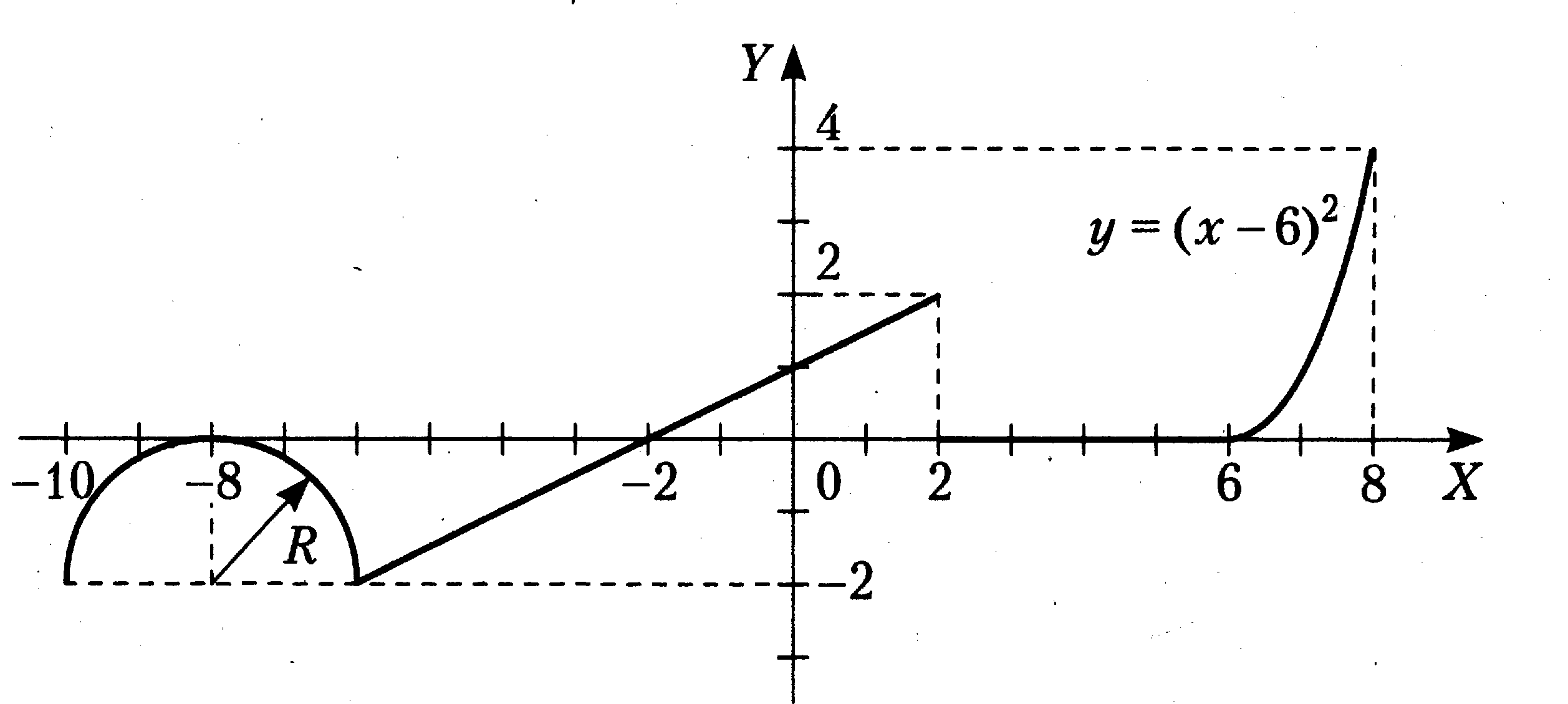
**Вариант 6**



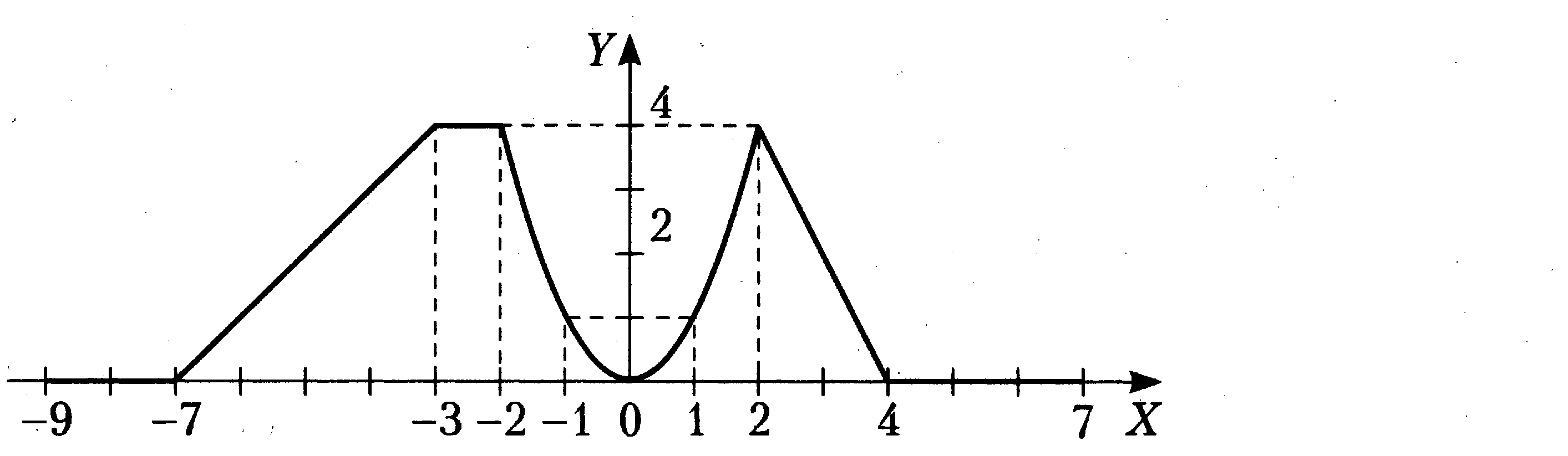
**Вариант 7**



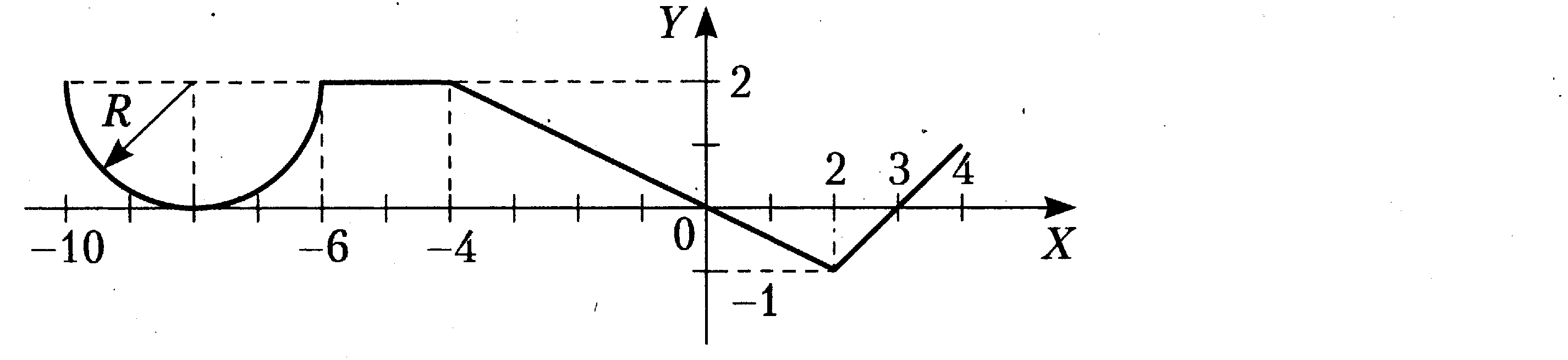
**Вариант 8**



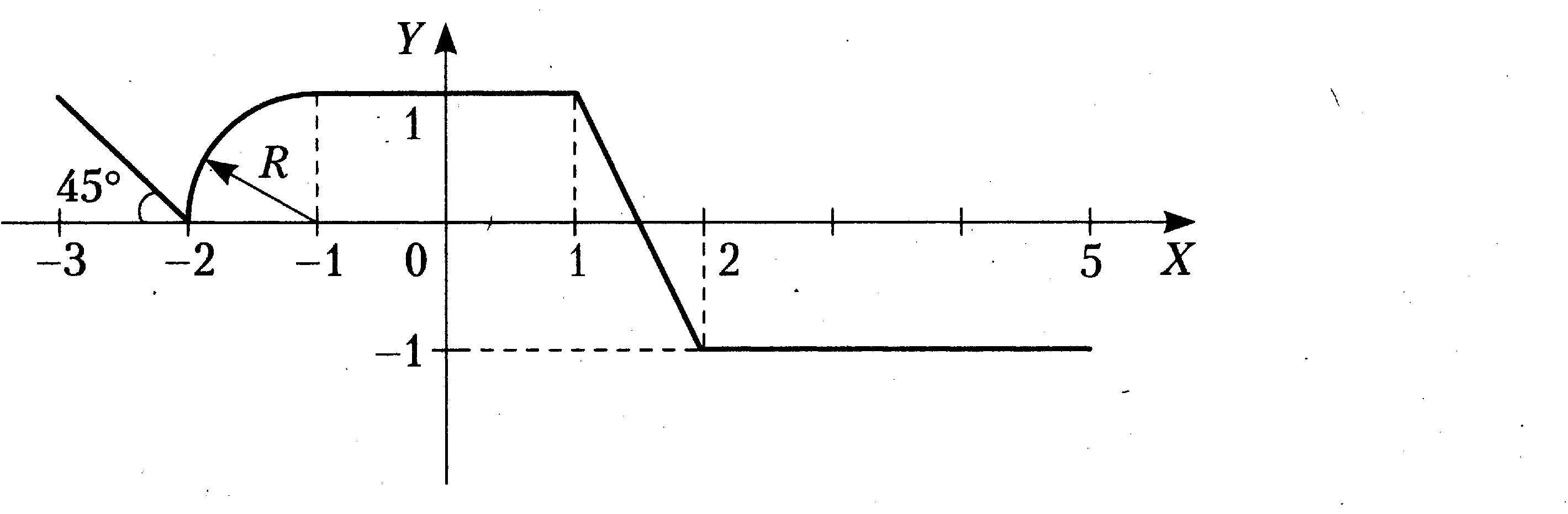
**Вариант 9**



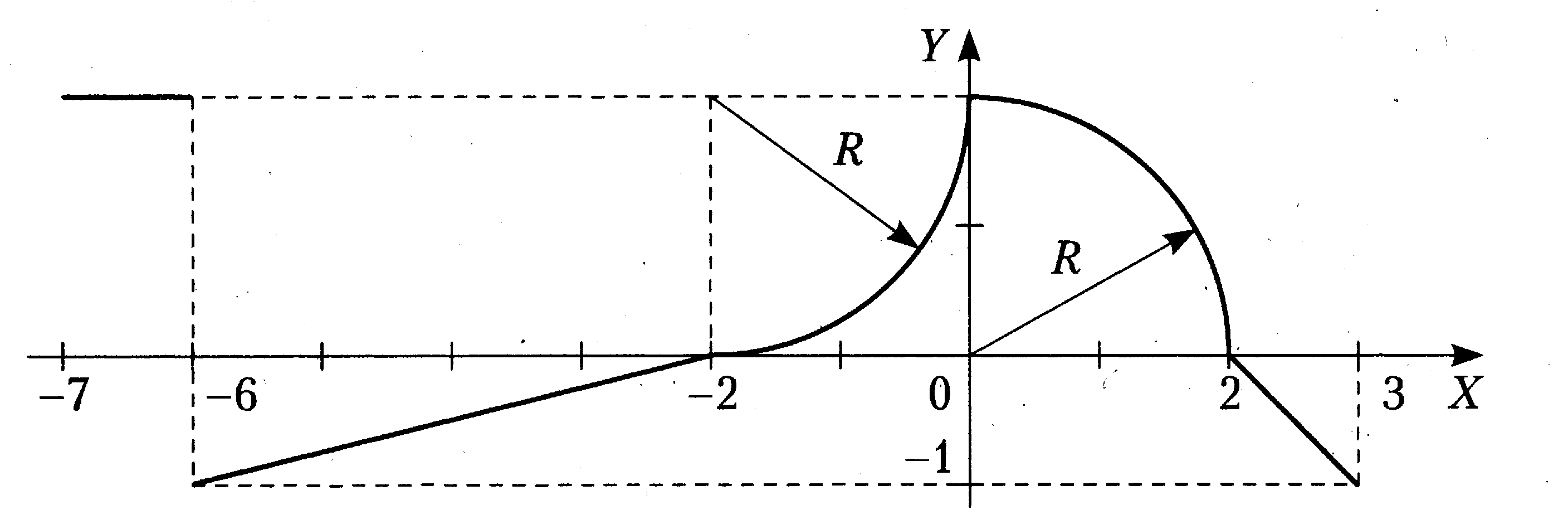
**Вариант 10**



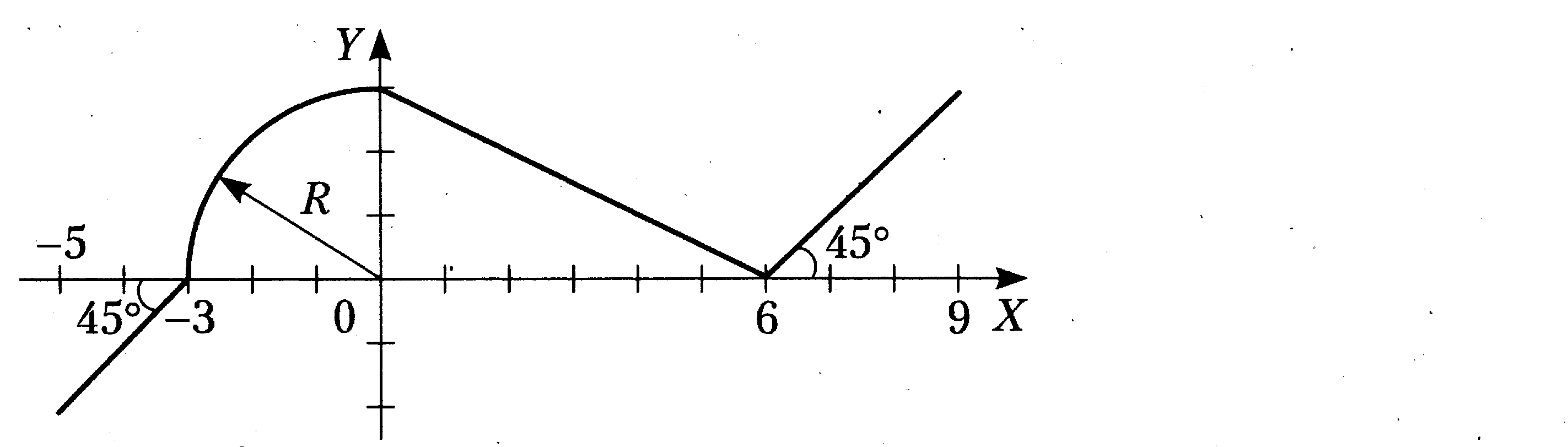
**Вариант 11**



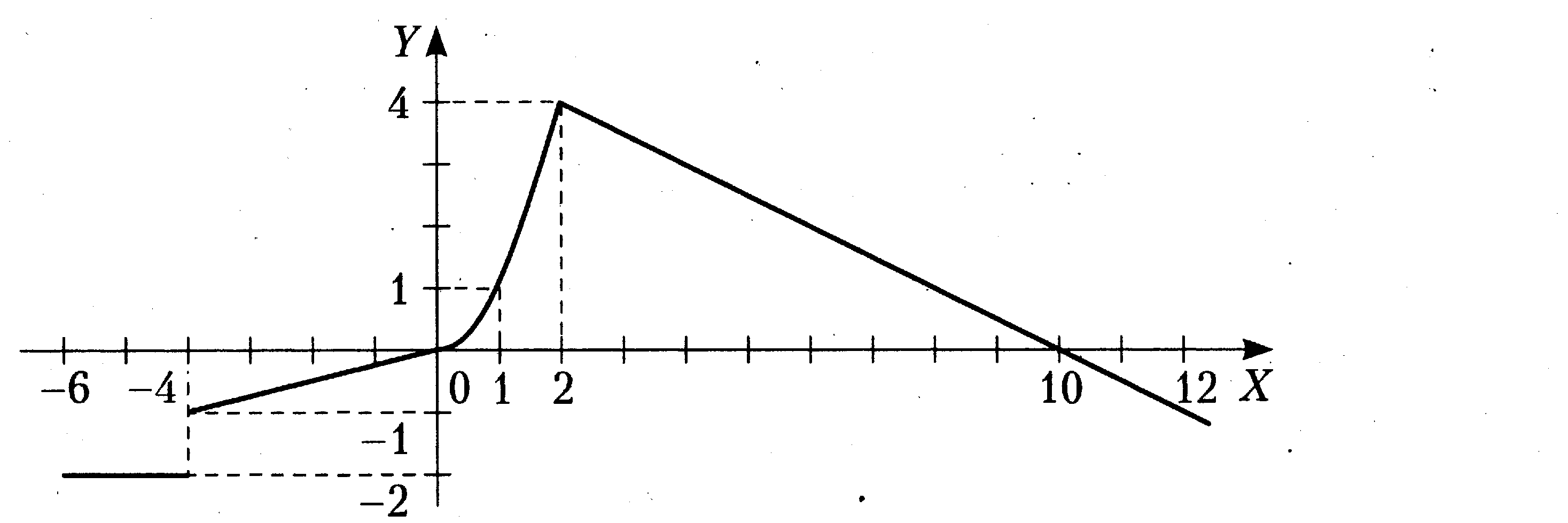
**Вариант 12**



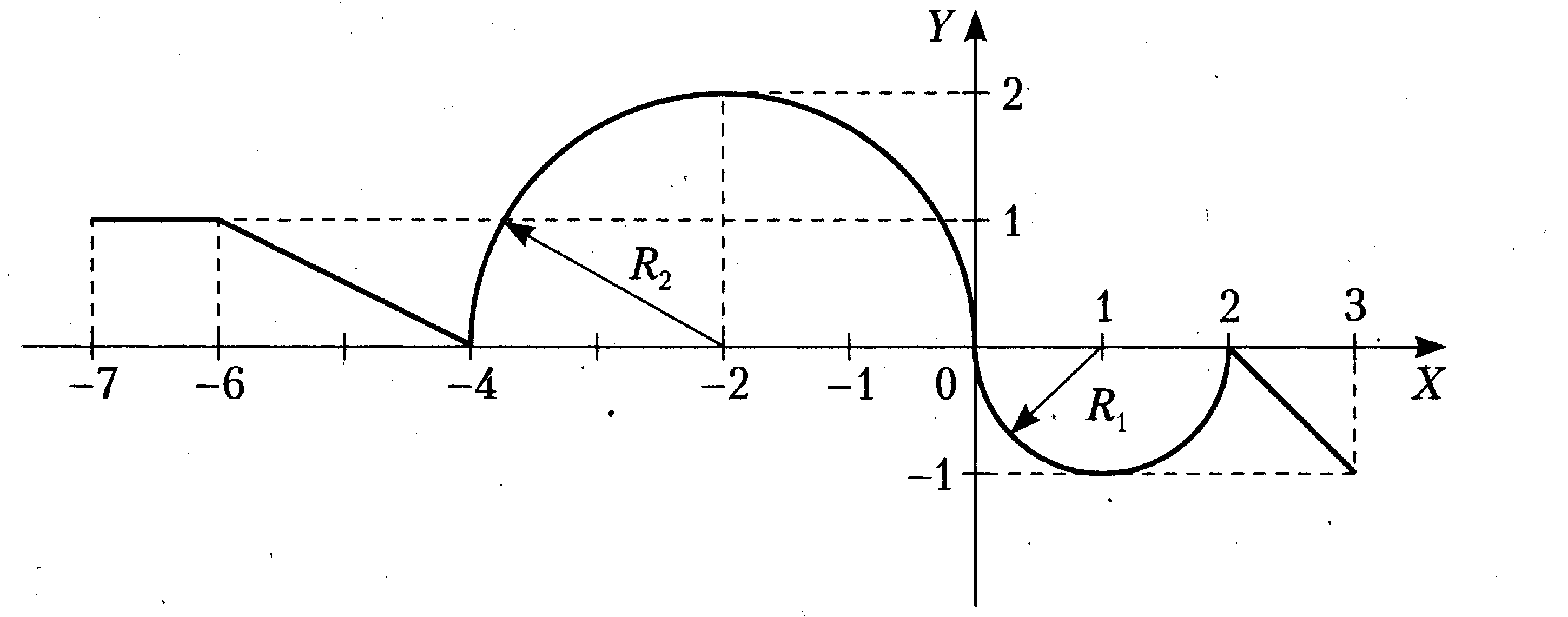
**Вариант 13**



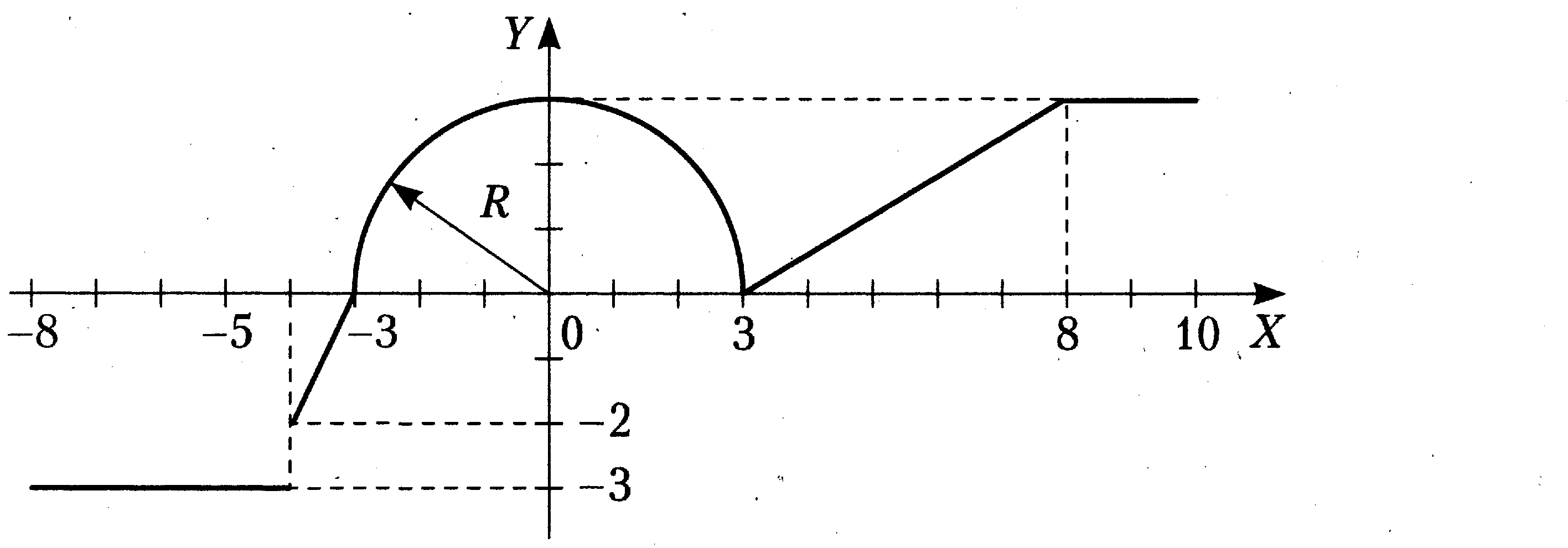
**Вариант 14**



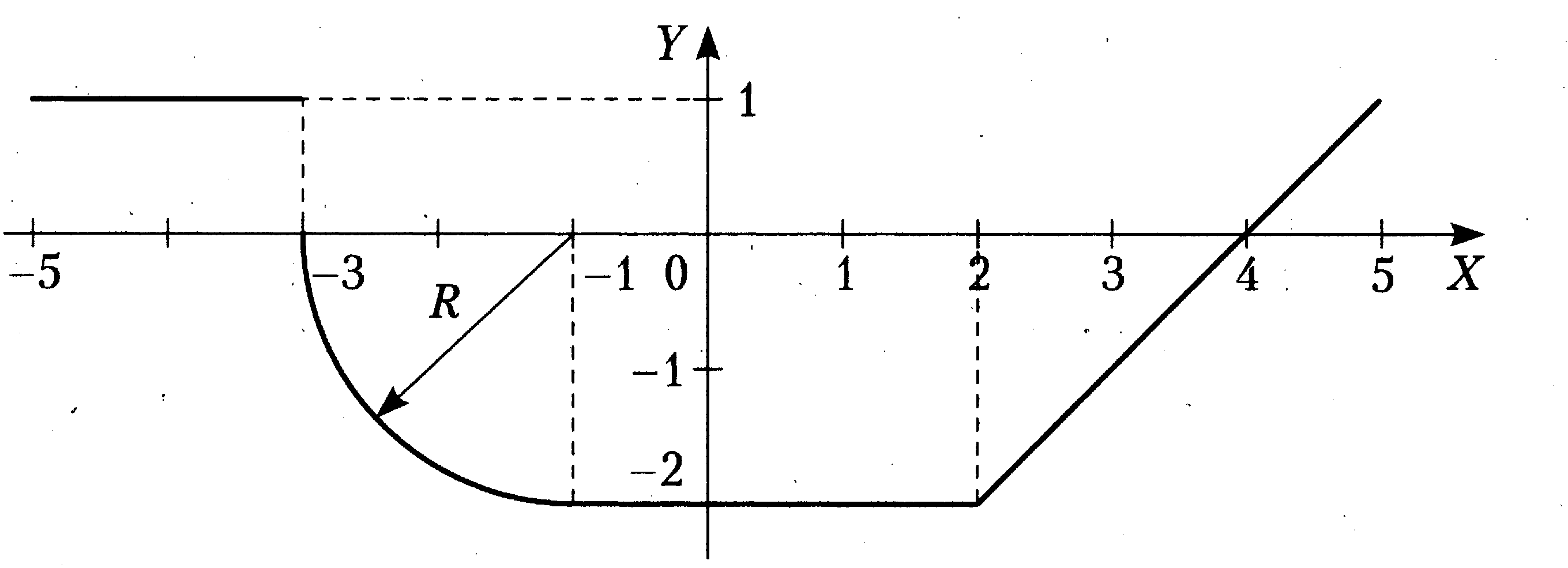
**Вариант 15**



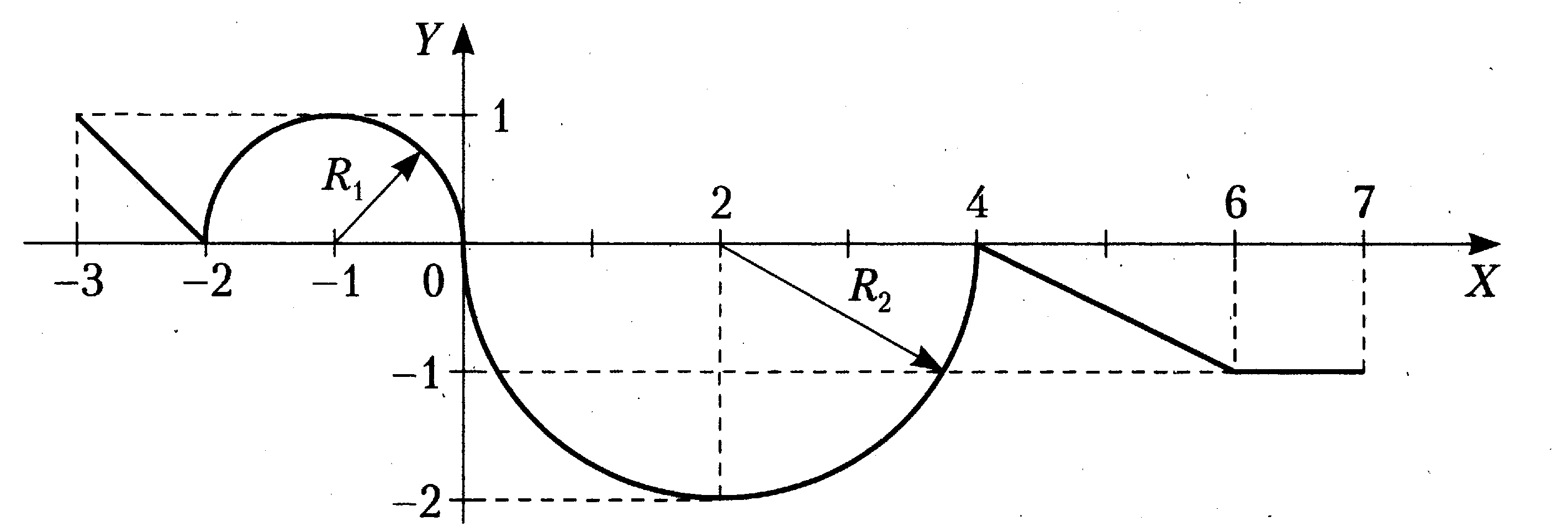
**Вариант 16**



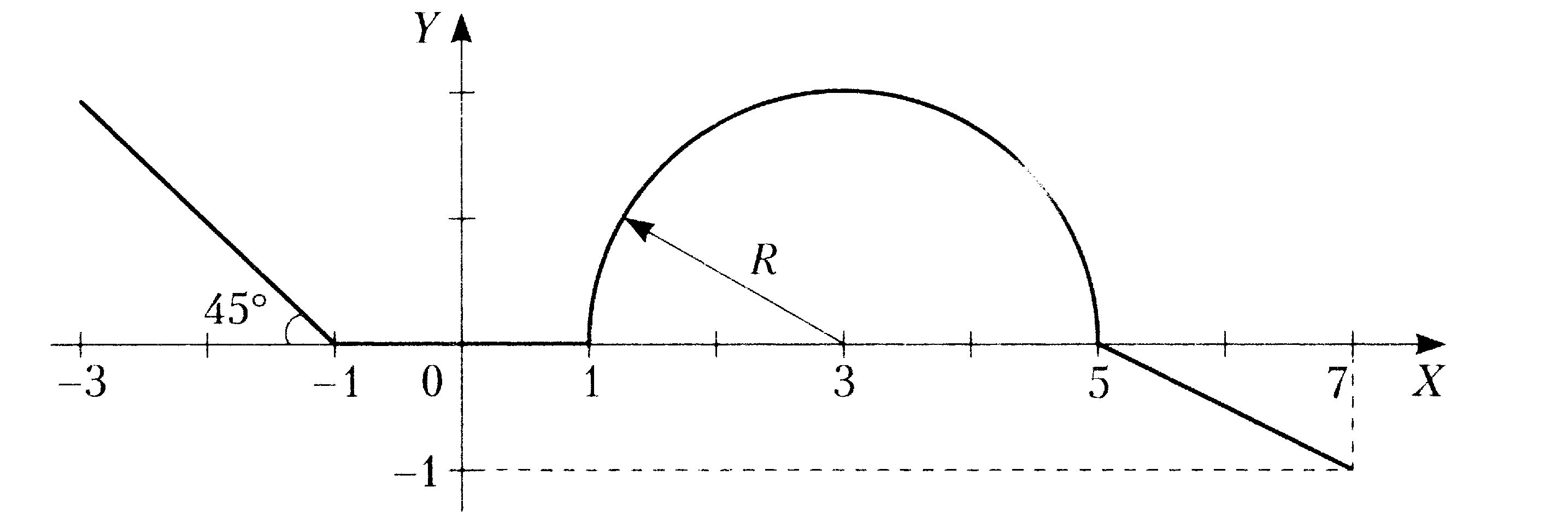
**Вариант 17**



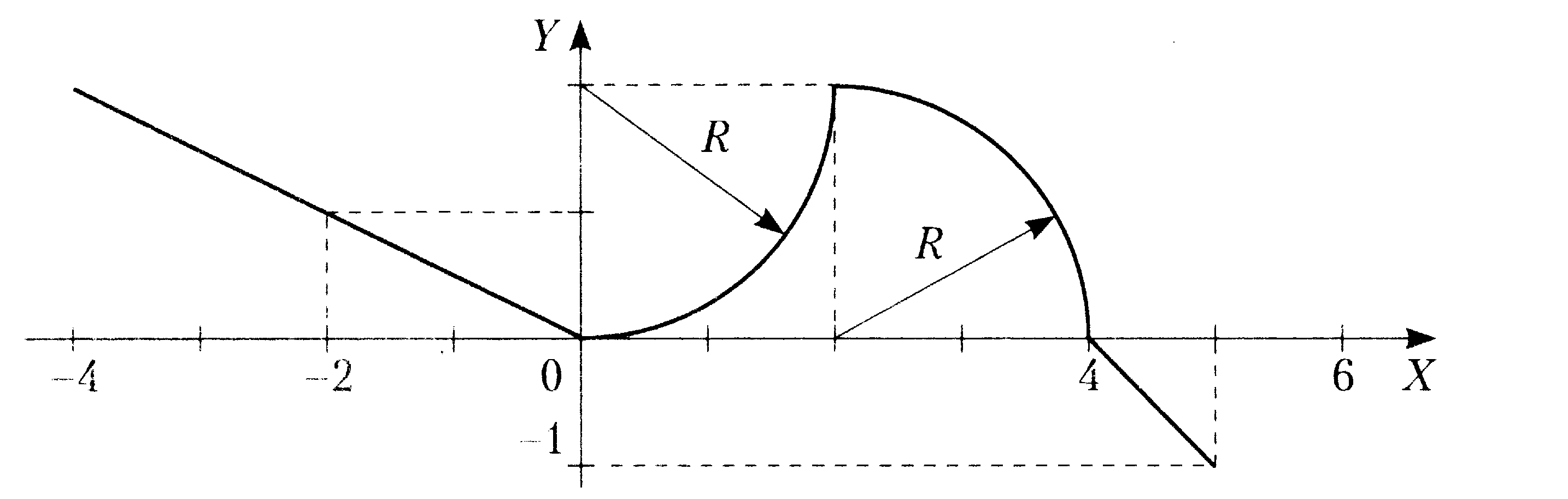
**Вариант 18**



**Вариант 19**



**Вариант 20**



1. Протестируйте работу приложения на заранее сформулированных тестовых примерах для своего варианта, позволяющих констатировать правильную работу программы для всех участков существования функций, описываемых разными выражениями.
2. Оформите отчет по лабораторной работе (Приложение 1).

**4. Задания для самостоятельной работы**

1. Опишите синтаксис условного оператора. Поясните особенности использования пар if..else.

2. Опишите смысл, цель тестирования программ и его виды. Поясните выбранные варианты контрольных примеров (тестовых заданий) для проверки правильной работы программы при выполнении задания 2 к лабораторной работе.

**Вопросы к лабораторной работе**

1. Какие принципы используют при составлении контрольных тестовых примеров для проверки работоспособности программы?
2. Каким образом использован условный оператор при выполнении задания 2 к лабораторной работе?
3. В чем заключается цель тестирования? отладки?

**Лабораторная работа 9**

**использование** **оператора цикла for. Часть 1**

**Цель работы**: изучить операторы, позволяющие организовывать циклическое выполнение программного кода.

Задача лабораторной работы:

– научиться применять оператор цикла for во вложенных циклах.

**1. Теоретическая часть**

Циклы позволяют выполнять определенную последовательность операторов до тех пор, пока выполняется некоторое условие. Каждый «круг» выполнения этого блока называется итерацией.

Синтаксис оператора:

**for** (*инициализатор; условие; итератор*)

*оператор*

где

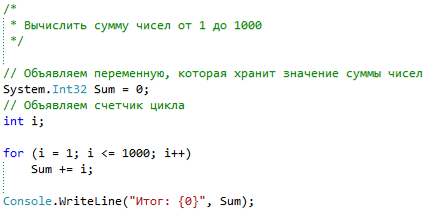
*инициализатор* – выражение, выполняемое перед первой итерацией цикла;

*условие* – выражение булевого типа, которое проверяется перед каждой итерацией. Если оно истинно, то итерация выполняется, иначе – цикл завершается;

*итератор* – выражение, вычисляемое после каждой итерации.

Приведем пример применения оператора цикла for (пример 1).

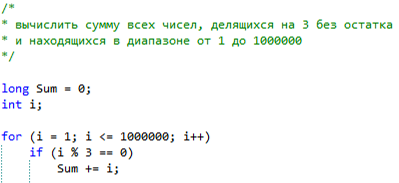
Пример 1.



Изучите код самостоятельно (создайте такую программу в VS).

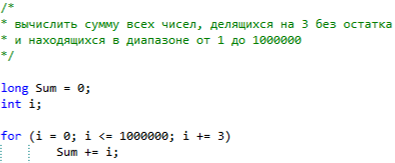
Приведем еще один пример использования цикла **for** (пример 2).

Пример 2.



Пример 3 отображает возможность применения итератора.

Пример 3.



Обратите внимание, что примеры 2 и 3 решают одну и ту же задачу.

**2. Материалы и оборудование**

Для выполнения лабораторной работы необходимо использовать:

1) персональный компьютер;

2) программное обеспечение:

- операционную систему windows 7 и выше,

- Microsoft Visual Studio 2010 и выше.

**3. Задания к лабораторной работе**

1. Изучите теоретическую часть к лабораторной работе.
2. Наберите, откомпилируйте и выполните программы примеров 1 и 3 раздела «Теоретическая часть» методических указаний. Приведите результаты выполнения и поясните их.
3. Модифицируйте приложение, разработанное в рамках задания 2 лабораторной работы 8, следующим образом:

3.1. Программа должна запросить исходные данные для вычисления выражения.

3.2. Программа вычисляет выражение и выводит результат.

3.3. Затем программа снова запрашивает ввод исходных данных и цикл повторяется.

С использованием цикла for организуйте указанное поведение программы и самостоятельно определите команду, которая приведет к завершению работы программы (например, нажата клавиша «Escape», пользователь должен ответить на вопрос «Продолжить?» и т.п.).

1. Оформите отчет по лабораторной работе (Приложение 1).

**Задания для самостоятельной работы**

1. Опишите возможности организации и цели использования циклов в C#-программах.

2. Опишите синтаксис цикла for.

3. В разделе «Теоретическая часть» приведены примеры 1, 2 и 3. Почему в примерах 2 и 3 для переменной Sum выбран тип long, хотя в примере 1 для Sum выбран тип int? Обоснуйте ответ.

**Вопросы к лабораторной работе**

1. Обязательно ли присутствие всех трех частей в заготовке цикла for?
2. Какие существуют варианты выхода из цикла for?
3. Что представляет собой бесконечный цикл for?

**Лабораторная работа 10**

**использование** **оператора цикла for. Часть 2**

**Цель работы**: изучить операторы, позволяющие организовывать циклическое выполнение программного кода.

Задача лабораторной работы:

– научиться применять оператор цикла for во вложенных циклах.

**1. Теоретическая часть**

В предыдущей лабораторной работе был приведен синтаксис оператора **for**:

**for** (*инициализатор; условие; итератор*)

*оператор*

Существует синтаксическое ограничение, требующее наличия одного оператора в теле цикла. Снять его достаточно легко: нужно вспомнить о возможности использования составного оператора в качестве одного оператора. Поэтому чаше всего *оператор* в теле цикла представляет собой блок, т.е. любое количество операторов, обрамленное парой фигурных скобок {}.

Заголовок оператора **for** разделен на три части: *инициализатор; условие; итератор.*

Обычно при написании программ *инициализатор* задает начальное значение переменной-счетчика цикла*; условие* представляет собой ограничение на изменение счетчика цикла*; итератор* задает величину, на которую изменяется значение счетчика цикла на каждой итерации, т.е. при каждом выполнении тела цикла.

Любая из частей заголовка может быть:

- опущена;

- состоять из выражений не для одной, а нескольких переменных.

Обязательным является лишь присутствие разделителей заголовка – «**;**».

Этими возможностями можно пользоваться, но не забывать о том, что при этом велика опасность создания бесконечного цикла и предусматривать возможность выхода из цикла.

**2. Материалы и оборудование**

Для выполнения лабораторной работы необходимо использовать:

1) персональный компьютер;

2) программное обеспечение:

- операционную систему windows 7 и выше,

- Microsoft Visual Studio 2010 и выше.

**3. Задания к лабораторной работе**

1. Изучите теоретическую часть к лабораторной работе, изложенную в методических указаниях.
2. Разработайте программу вычисления суммы последовательности согласно варианту задания (1-25) к лабораторной работе. Во всех вариантах заданий переменные X, Y являются вещественными и вводятся пользователем. Количество слагаемых также вводится пользователем. Программа должна вывести сумму заданного числа членов последовательности (варианты 1-25):

|  |  |
| --- | --- |
| Вариант | Выражение для вычисления |
| 1 |  |
| 2 |  |
| 3 |  |
| 4 |  |
| 5 |  |
| 6 | (в знаменателе факториал) |
| 7 |  |
| 8 |  |
| 9 |  |
| 10 |  |
| 11 |  |
| 12 |  |
| 13 |  |
| 14 |  |
| 15 |  |
| 16 |  |
| 17 |  |
| 18 |  |
| 19 |  |
| 20 |  |
| 21 |  |
| 22 |  |
| 23 |  |
| 24 |  |
| 25 |  |

1. Измените код программы, как предложено в задании 3 к лабораторной работе 9 для варианта индивидуального задания, предоставив пользователю возможность вычислений с несколькими значениями X, Y и количеством слагаемых.
2. Оформите отчет по лабораторной работе (Приложение 1).

**4. Задания для самостоятельной работы**

1. Опишите синтаксис заголовка оператора цикла for. Возможен ли пропуск одной из трех частей заголовка? Ответ поясните на примерах.

2. Приведите пример реализации бесконечного цикла for. Поясните работу программы примера.

**Вопросы к лабораторной работе**

1. Как снять синтаксическое ограничение, требующее наличия одного оператора в теле цикла for?
2. Где и каким образом можно использовать бесконечные циклы for?
3. Как использован цикл for при выполнении варианта индивидуального задания?

**Лабораторная работа 11**

**использование оператора цикла while. часть 1**

**Цель работы**: изучить операторы, позволяющие организовывать непоследовательное выполнение программного кода.

Задача лабораторной работы:

– научиться применять оператор цикла с предусловием while.

**1. Теоретическая часть**

Цикл while похож на for тем, что является конструкцией с предварительной проверкой условия продолжения цикла. Но синтаксис цикла while более лаконичен:

while (*условие*)

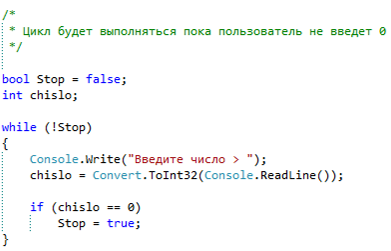
*оператор*

В данном синтаксисе *оператор* может быть составным оператором (группа операторов языка, заключенные в фигурные скобки).

Следует понимать, что while используется для заранее неизвестного количества повторных выполнений операторов.

Приведем пример выполнения цикла (пример 1).

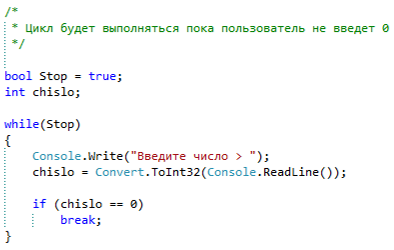
Пример 1.



Оператор break уже встречался в конструкции case оператора switch для выхода из case. Но break часто применяется внутри циклов for, while и do … while. Данный оператор прерывает выполнение цикла и передает управление оператору, следующему за циклом.

Пример 2 выполняет то же самое, что и пример 1.

Пример 2.



Следует обратить внимание, что цикл является бесконечным (логическая переменная Stop всегда является true), но выполнение цикла все равно прервется, если пользователь введет 0.

Оператор continue также применяется внутри цикла – он останавливает выполнение текущей итерации и немедленно переходит к выполнению следующей итерации.

**2. Материалы и оборудование**

Для выполнения лабораторной работы необходимо использовать:

1) персональный компьютер;

2) программное обеспечение: операционную систему windows 7 и выше, Microsoft Visual Studio 2010 и выше.

**3. Задания к лабораторной работе**

1. Изучите теоретическую часть к лабораторной работе.
2. Создайте консольное приложение в соответствии с алгоритмом, представленным в лабораторной работе 1.Напишите, откомпилируйте и выполните программы примеров 1 и 2 теоретической части. Проанализируйте результаты выполнения.
3. Оформите отчет по лабораторной работе (Приложение 1).

**4. Задания для самостоятельной работы**

1. Проанализируйте преимущества и недостатки циклов while и for.

2. Опишите синтаксис оператора while.

**Вопросы к лабораторной работе**

1. Чем цикл while похож на цикл for?
2. С какой целью оператор continue применяется внутри цикла?
3. Как действуетоператор break внутри цикла?

**Лабораторная работа 12**

**использование оператора цикла while. часть 2**

**Цель работы**: изучить операторы, позволяющие организовывать непоследовательное выполнение программного кода.

Задача лабораторной работы:

– научиться применять оператор цикла с предусловием while.

**1. Теоретическая часть**

Формат оператора while прост:

while (выражение) оператор

выражение должно быть логического типа: например, это может быть операция отношения или просто логическая переменная.

Если результат вычисления выражения равен true, выполняется простой или составной оператор (блок).

Эти действия повторяются до того момента, пока результатом выражения не станет значение false.

После окончания цикла управление передается следующему за ним оператору.

Выражение вычисляется перед каждой итерацией цикла. Если при первой проверке выражение равно false, цикл не выполнится ни разу.

ВНИМАНИЕ – Если в теле цикла необходимо выполнить *более одного оператора*, необходимо заключить их в *блок* с помощью фигурных скобок.

**2. Материалы и оборудование**

Для выполнения лабораторной работы необходимо использовать:

1) персональный компьютер;

2) программное обеспечение:

- операционную систему windows 7 и выше,

- Microsoft Visual Studio 2010 и выше.

**3. Задания к лабораторной работе**

1. Изучите теоретическую часть к лабораторной работе.
2. Создайте консольное приложение в соответствии с алгоритмом, представленным в лабораторной работе 1.
3. Выполните индивидуальное задание по варианту (1-25). Во всех вариантах переменные X, Y являются вещественными и вводятся пользователем. Отличие от задания 2 к лабораторной работе 10: количество слагаемых пользователем *не вводится*! Программа должна работать следующим образом:

– пользователю выводится приглашение на ввод X и Y;

– пользователь вводит X и Y;

– программа начинает расчет суммы, при этом выводится результат расчета, полученный на каждой итерации;

– сначала выводится номер итерации (1) и сумма 1-го слагаемого, затем программа останавливается и ждет ввода команды пользователя (1 – продолжить, 0 – прекратить расчет);

– если пользователь продолжает, выводится номер итерации (2) и сумма 2-х слагаемых, и снова программа ждет команды пользователя, и т.д.

Программа должна использовать цикл while.

4. Оформите отчет по лабораторной работе (Приложение 1).

|  |  |
| --- | --- |
| **Вариант** | **Выражение для вычисления** |
| 1 | (в знаменателе факториал) |
| 2 |  |
| 3 |  |
| 4 |  |
| 5 |  |
| 6 |  |
| 7 |  |
| 8 |  |
| 9 |  |
| 10 |  |
| 11 |  |
| 12 |  |
| 13 |  |
| 14 |  |
| 15 |  |
| 16 |  |
| 17 |  |
| 18 |  |
| 19 |  |
| 20 |  |
| 21 |  |
| 22 |  |
| 23 |  |
| 24 |  |
| 25 |  |

**4. Задания для самостоятельной работы**

1. Напишите программу заполнения целочисленного массива данными, а затем реализуйте в циклеwhile перебор введенных значений и их распечатку.

2. Модифицируйте программу, позволив пользователю вводить элементы массива в цикле while до набора условленного значения (например, 0).

**Вопросы к лабораторной работе**

1. Каким должен быть тип выражения, стоящего в скобках после ключевого слова while?
2. Как реализовать выход из бесконечного цикла while?
3. Как реализована программа варианта индивидуального задания?

**Лабораторная работа 13**

**использование оператора цикла do …while. Часть 1**

**Цель работы**: изучить операторы, позволяющие организовывать непоследовательное выполнение программного кода.

Задачи лабораторной работы:

– научиться применять оператор цикла с постусловием do … while.

– изучить особенности оператора do…while;

– научиться использовать оператор цикла do…while для решения практических задач.

**1. Теоретическая часть**

Цикл do … while полностью повторяет функциональные возможности while, но предполагает проверку условия окончания цикла *после* выполнения тела цикла. То есть операторы тела цикла всегда выполнятся хотя бы один раз.

Синтаксис оператора:

do

{

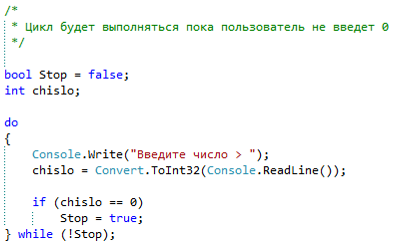
оператор (операторы)

}

while (*условие*);

Приведем пример использования цикла (пример 1).

Пример 1.



**2. Материалы и оборудование**

Для выполнения лабораторной работы необходимо использовать:

1) персональный компьютер;

2) программное обеспечение:

- операционную систему windows 7 и выше,

- Microsoft Visual Studio 2010 и выше.

**3. Задания к лабораторной работе**

1. Изучите теоретическую часть к лабораторной работе, изложенную в методических указаниях.
2. Создайте консольное приложение в соответствии с алгоритмом, представленным в лабораторной работе 1. Наберите, откомпилируйте и выполните программу примера 1, приведенного в теоретической части методических указаний. Сформулируйте выводы, проанализировав результаты выполнения программы.
3. Оформите отчет по лабораторной работе (Приложение 1).

**4. Задания для самостоятельной работы**

1. Опишите преимущества и недостатки циклов do…while и while.

2. Опишите синтаксис оператора do…while.

**Вопросы к лабораторной работе**

1. В каком цикле тело цикла всегда выполняется хотя бы один раз?

2. Каковы преимущества и недостатки использования трех типов цикла в программах?

3. При каком значении условного выражения происходит выход из цикла do …while?

**Лабораторная работа 14**

**использование оператора цикла do …while. Часть 2**

**Цель работы**: изучить операторы, позволяющие организовывать непоследовательное выполнение программного кода.

Задачи лабораторной работы:

– научиться применять оператор цикла с постусловием do … while.

– изучить особенности оператора do…while;

– научиться использовать оператор цикла do…while для решения практических задач.

**1. Теоретическая часть**

Цикл с постусловием do...while реализует структурную схему, приведенную на рисунке 14.1. и имеет вид:

do оператор while выражение;

Сначала выполняется простой или составной оператор, образующий тело цикла, а затем вычисляется выражение, которое должно иметь тип bool.

Если выражениеравно true (истинно), тело цикла выполняется еще раз и проверка повторяется.



Рисунок 14.1 – Структурная схема, реализующая цикл с постусловием do...while

Цикл *завершается*, когда выражение станет равным false или в теле цикла будет выполнен какой-либо оператор передачи управления.

Этот вид цикла применяется в тех случаях, когда тело цикла необходимо обязательно выполнить хотя бы один раз, например, если в цикле вводятся данные и выполняется их проверка.

Если же такой необходимости нет, предпочтительнее пользоваться циклом с предусловием.

Пример программы, выполняющей проверку ввода:

using System;

namespace ConsoleApplication1

{ class Class1

{ static void Main()

{

char answer;

do

{

Console.WriteLine("Купи слоника, a?");

answer = (char) Console.Read();

Console.ReadLine();

} while (answer != 'y');

}

}

}

**2. Материалы и оборудование**

Для выполнения лабораторной работы необходимо использовать:

1) персональный компьютер;

2) программное обеспечение:

- операционную систему windows 7 и выше,

- Microsoft Visual Studio 2010 и выше.

**3. Задания к лабораторной работе**

1. Изучите теоретическую часть к лабораторной работе.
2. Создайте консольное приложение в соответствии с алгоритмом, представленным в лабораторной работе 1.
3. Выполните индивидуальное задание, используя методику выполнения задания 3 для варианта индивидуального задания из предыдущей лабораторной работы. Во всех заданиях переменные X, Y являются вещественными и вводятся пользователем. Количество слагаемых пользователем не вводится. Программа должна работать следующим образом:

– пользователю выводится приглашение на ввод X и Y;

– пользователь вводит X и Y;

– программа начинает расчет суммы, при этом выводится результат расчета, полученный на каждой итерации;

– сначала выводится номер итерации (1) и сумма 1-го слагаемого, затем программа останавливается и ждет ввода команды пользователя (1 – продолжить, 0 – прекратить расчет);

– если пользователь продолжает, выводится номер итерации (2) и сумма двух слагаемых, и снова программа ждет команды пользователя, и т.д.

Программа должна использовать цикл do … while.

1. Оформите отчет к лабораторной работе (Приложение 1).

**4. Задания для самостоятельной работы**

1. Напишите программу заполнения целочисленного массива данными, а затем реализуйте в циклеdo…while перебор введенных значений и их распечатку.

2. Модифицируйте программу, позволив пользователю вводить элементы массива в цикле do…while до набора условленного значения (например, 0).

**Лабораторная работа 15**

**использование Классов и Структур. Часть 1**

**Цель работы**: изучить структуру и принципы объявления классов, освоить технологию создания экземпляров классов (объектов).

Задачи лабораторной работы:

– научиться объявлять классы;

– научиться создавать объекты классов;

– научиться работать с полями данных и методами классов.

**1. Теоретическая часть**

**1.1. Классы и структуры**

Класс – это тип данных, объединяющий данные и методы их обработки.

Класс – это пользовательский шаблон, в соответствии с которым можно создавать объекты. То есть класс – это правило, по которому будет строиться объект. Сам класс не содержит данных.

Объект класса (экземпляр класса) – переменная типа класс.

Объект содержит данные и методы, манипулирующие этими данными. Класс определяет, какие данные содержит объект и каким образом он ими манипулирует.

*Структуры*. Все, что справедливо для классов, можно распространить и на структуры.

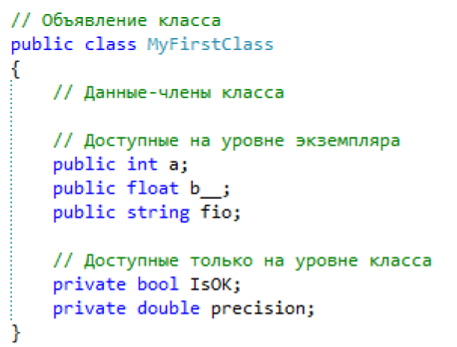
Отличие состоит в методе хранения объектов данных типов в оперативной памяти: структуры – это типы по значению, они размещаются в стеке; классы – это ссылочные типы, объекты классов размещаются в куче. Структуры не поддерживают наследование.

Структуры применяются для представления небольших объемов данных.

Объявление структур происходит с использованием ключевого слова struct, объявление классов – с помощью ключевого слова class.

Приведем пример объявления класса (пример 1).

**Пример 1**. Объявление класса MyFirstClass



При создании как классов, так и структур используется ключевое слово new, например:



**1.2. Структура класса**

Данные и функции, объявленные внутри класса, называются членами класса (class members). Доступность членов класса может быть описана как public, private, protected, internal или internal protected.

**1.2.1. Данные-члены**. Данные-члены – это те структуры внутри класса, которые содержат данные класса – поля, константы события.

Поля – это любые переменные, ассоциированные с классом. После создания экземпляра класса к полям можно обращаться с использованием синтаксиса, например: ИмяПоля.ИмяОбъекта.



Аналогичным образом с классом ассоциируются константы.

События будут рассмотрены в следующих лабораторных работах.

**1.2.2. Функции-члены**. Функции-члены – это члены, которые обеспечивают некоторую функциональность для манипулирования данными классов. Они делятся на следующие виды: методы, свойства, конструкторы, финализаторы, операции и индексаторы.

Методы (method) – это функции, ассоциированные с определенным классом. Как и данные-члены, по умолчанию они являются членами экземпляра. Они могут быть объявлены статическими с помощью модификатора static.

Свойства (property) – это наборы функций, которые могут быть доступны клиенту таким же способом, как общедоступные поля класса. В С# предусмотрен специальный синтаксис для реализации чтения и записи свойств для классов, поэтому писать собственные методы с именами, начинающимися на Set и Get, не понадобится. Поскольку не существует какого-то отдельного синтаксиса для свойств, который отличал бы их от нормальных функций, создается иллюзия объектов как реальных сущностей, предоставляемых клиентскому коду.

Конструкторы (constructor) – это специальные функции, вызываемые автоматически при инициализации объекта. Их имена совпадают с именами классов, которым они принадлежат, и они не имеют типа возврата. Конструкторы полезны для инициализации полей класса.

Финализаторы (finalizer) похожи на конструкторы, но вызываются, когда среда CLR определяет, что объект больше не нужен. Они имеют то же имя, что и класс, но с предшествующим символом тильды (~). Предсказать точно, когда будет вызван финализатор, невозможно.

Операции (operator) – это простейшие действия вроде + или -. Когда вы складываете два целых числа, то, строго говоря, применяете операцию + к целым. Однако С# позволяет указать, как существующие операции будут работать с пользовательскими классами (так называемая перегрузка операций).

Индексаторы (indexer) позволяют индексировать объекты таким же способом, как массив или коллекцию.

В данной лабораторной работе рассматриваются только методы класса – это функции, ассоциированные с определенным классом.

В C# объявление метода класса состоит из спецификатора доступности, возвращаемого значения, имени метода, списка формальных параметров и тела метода:



**Пример 2**. Добавим методы для объявленного ранее класса MyFirstClass:



Синтаксис вызова методов аналогичен синтаксису обращения к данным-членам:



В данном примере метод InitClassMembers не возвращает никаких данных, но требует передачи ему фактических параметров. В свою очередь, метод GetAbsA возвращает значение типа int и не предполагает никаких параметров.

В общем случае параметры могут передаваться методу либо по значению, либо по ссылке. Когда переменная передается по ссылке, вызываемый метод получает саму переменную, поэтому любые изменения, которым она подвергнется внутри метода, останутся в силе после его завершения. Но если переменная передается по значению, вызываемый метод получает копию этой переменной, а это значит, что все изменения в ней по завершении метода будут утеряны. Для сложных типов данных передача по ссылке более эффективна из-за большого объема данных, который приходится копировать при передаче по значению.

Если не указано обратное, то в С# все параметры передаются по значению. Тем не менее, можно принудительно передавать значения по ссылке, для чего используется ключевое слово ref. Если параметр передается в метод, и входной аргумент этого метода снабжен префиксом ref, то любые изменения этой переменной, которые сделает метод, отразятся на исходном объекте.

В С-подобных языках функции часто возвращают более одного значения. Это обеспечивается применением выходных параметров, за счет присваивания значений переменным, переданным в метод по ссылке. Часто первоначальное значение таких переменных не важно. Эти значения перезаписываются в функции, которая может даже не обращать внимания на то, что в них хранилось первоначально.

Было бы удобно использовать то же соглашение в С#. Однако в С# требуется, чтобы переменные были инициализированы каким-то начальным значением перед тем, как к ним будет выполнено обращение. Хотя можно инициализировать входные переменные какими-то бессмысленными значениями до передачи их в функцию, которая наполнит их осмысленными значениями, этот прием выглядит в лучшем случае излишним, а в худшем – сбивающим с толку. Тем не менее, существует способ обойти требование компилятора С# относительно начальной инициализации переменных.

Это достигается ключевым словом out. Когда входной аргумент снабжен префиксом out, этому методу можно передать неинициализированную переменную. Переменная передается по ссылке, поэтому любые изменения, выполненные методом в переменной, сохраняются после того, как он вернет управление. Ключевое слово out также должно указываться при вызове метода – так же, как при его определении.

**Частичные классы**. Ключевое слово partial (частичный) позволяет определить класс, структуру или интерфейс, распределенный по нескольким файлам. Но ситуации, когда множеству разработчиков требуется доступ к одному и тому же классу, или же в ситуации, когда некоторый генератор кода генерирует часть класса, такое разделение класса на несколько файлов может оказаться полезным. Ключевое слово partial просто помещается перед классом, структурой или интерфейсом.

**Статические классы**. Статический класс функционально представляет собой то же самое, что и класс с приватным статическим конструктором. Создать экземпляр такого класса невозможно. Если указать ключевое слово static в объявлении класса, компилятор будет гарантировать, что к этому классу никогда не будут добавлены нестатические члены.

**Класс Object**. Классы .NET изначально унаследованы от System.Object. Фактически, если при определении нового класса базовый класс не указан, компилятор автоматически предполагает, что он наследуется от Object.

Практическое значение этого в том, что помимо методов и свойств, которые программист определяет самостоятельно, также появляется доступ к множеству общедоступных и защищенных методов-членов, которые определены в классе Object. Эти методы присутствуют во всех определяемых классах.

**2. Материалы и оборудование**

Для выполнения лабораторной работы необходимо использовать:

1) персональный компьютер;

2) программное обеспечение:

- операционную систему windows 7 и выше,

- Microsoft Visual Studio 2010 и выше.

**3. Методика и порядок выполнения работы**

**Пример выполнения задания.** Разработаем класс для представления объекта «Прямоугольный параллелепипед». Реализуем все необходимые поля данных (закрытые) и методы, позволяющие:

– устанавливать и считывать значения полей данных;

– вычислять объем прямоугольного параллелепипеда;

– вычислять площадь поверхности прямоугольного параллелепипеда;

– выводить полную информацию об объекте в консоль.

Решение данной задачи состоит из двух этапов:

1) объявление класса Parallelepiped и

2) демонстрация использования объекта данного класса.

Полный листинг примера представлен на рисунках 15.1 – 15.2.





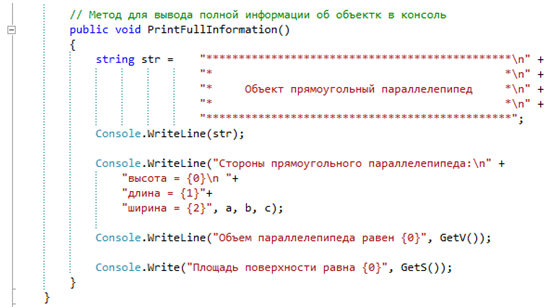


Рисунок 15.1 – Объявление класса Parallelepiped

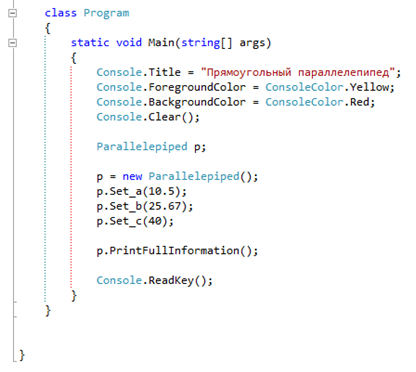


Рисунок 15.2 – Метод Main класса Program

В данном примере обратите внимание на тот факт, что все вычисления выполняются внутри класса.

Метод Main содержит только вызовы методов класса, то есть вся реализация скрыта.

В результате выполнения программы отобразится консольное окно с выводом информации (рис. 15.3).



Рисунок 15.3 – Консольное окно с выводом информации

**4. Задания к лабораторной работе**

1. Изучите примеры 1 и 2 теоретической части к лабораторной работе.
2. Изучите пример выполнения задания, представленный в части 3. Методика и порядок выполнения работы.
3. Создайте консольное приложение в соответствии с алгоритмом, представленным в лабораторной работе 1.
4. Объявите класс Parallelepiped (рис. 15.1).
5. Поместите вызовы методов класса в метод Main (рис. 15.2).
6. Отладьте и выполните программу.
7. Сопоставьте результат выполнения программы с результатом, приведенным в примере (рис. 15.3). Сформулируйте выводы.
8. Оформите отчет по лабораторной работе (Приложение 1).

**5. Задания для самостоятельной работы**

1. Дайте определение класса.

2. Дайте определение структуры. Чем структура отличается от класса?

3. Опишите разновидности членов класса.

4. Опишите типы членов-данных.

5. Объявите класс MyFirstClass (пример 1 теоретической части), добавьте методы (пример 2 теоретической части). Создайте метод Main для вызова методов класса для его объектов.

**Вопросы к лабораторной работе**

1. Как описать объект х класса Class1?
2. Чем переменные и методы, являющиеся членами класса, отличаются от внешних переменных и методов?
3. Какие методы имеют доступ к значениям переменных-членов класса? От чего это зависит?

**Лабораторная работа 16**

**использование Классов и Структур. Часть 2**

**Цель работы**: изучить структуру и принципы объявления классов, освоить технологию создания экземпляров классов (объектов).

Задачи лабораторной работы:

– научиться объявлять классы;

– научиться создавать объекты классов;

– научиться работать с полями данных и методами классов.

**1. Теоретическая часть**

Итак, известно, что ***класс*** – это тип данных, объединяющий данные и методы их обработки. Объектом класса (экземпляром класса) является переменная типа класс.

Класс объединяет данные-члены и функции-члены. Они могут обладать разной степенью видимости, наиболее употребительными из спецификаторов доступа являются:

public – общедоступные данные и методы.

private – частные (закрытые) данные и методы, доступные только методам класса (принята по умолчанию).

protected – защищенные данные и методы, доступные методам класса и их потомкам.

Общепринятой практикой является использование частных данных (переменных) класса и общедоступных методов класса, осуществляющих доступ к закрытым переменным.

Конструктор класса – это функция с именем, совпадающим с именем класса, которому он принадлежит. Они вызываются автоматически при создании объекта класса, поэтому обычно используются для инициализации полей класса.

Объявление метода класса:

**[спецификатор\_доступа] тип\_возвращаемого\_значения имя\_метода([параметры])**

**{**

**// тело метода**

**}**

Обращаться к методам и данным класса можно только после создания переменной класса, например, объекта (переменной класса) **obj** классаMyFirstClass. К методам и переменным класса можно для объекта этого класса через оператор «точку» (**.**):

MyFirstClass **obj** = new MyFirstClass();// создание объекта.

// Вызов метода InitClassMembers с передачей параметров:

**obj**.InitClassMembers(10, 0.8F, “Новиков П.Е.”);

// Присвоение переменной класса abs\_a значения,

// возвращаемого методом класса getAbsA():

int abs\_a = **obj**.getAbsA();

Обычно при разработке объектных приложений все необходимые вычисления выполняются внутри класса в его методах, а метод Main содержит только создание и инициализацию объектов (переменных) класса, а также вызовы методов класса, то есть вся реализация скрыта.

Учтите это замечание при выполнении заданий к лабораторной работе.

**2. Материалы и оборудование**

Для выполнения лабораторной работы необходимо использовать:

1) персональный компьютер;

2) программное обеспечение: операционную систему windows 7 и выше, Microsoft Visual Studio 2010 и выше.

**3. Задания к лабораторной работе**

1. Изучите теоретическую часть к лабораторной работе.
2. Выполните задание по варианту (1-25): спроектируйте класс, наполните его требуемой функциональностью, продемонстрируйте работоспособность класса.
3. Оформите отчет по лабораторной работе (Приложение 1).

|  |  |
| --- | --- |
| Вариант | Выражение для вычисления |
| 1. | Класс «Шар». Реализовать ввод и вывод полей данных, вычисление объема, диаметра и площади поверхности, а также вывод информации об объекте. |
| 2. | Класс «Куб». Реализовать ввод и вывод полей данных, вычисление объема, площади поверхности, длины диагонали, а также вывод информации об объекте. |
| 3. | Класс «Сфера». Реализовать ввод и вывод полей данных, вычисление объема, диаметра и площади поверхности, а также вывод информации об объекте. |
| 4. | Класс «Точка в пространстве». Реализовать ввод и вывод полей данных, вычисление расстояния до введенной пользователем точки, расстояния от начала координат, а также вывод информации об объекте. |
| 5. | Класс «График y=x». Реализовать ввод и вывод полей данных, вычисление интеграла функции от a до b (вводятся пользователем), длины отрезка функции от (a, y(a)) до (b, y(b)), а также вывод информации об объекте. |
| 6. | Класс «Шар». Реализовать ввод и вывод полей данных, вычисление объема, диаметра и площади поверхности, а также вывод информации об объекте. |
| 7. | Класс «Матрица M×N». Реализовать инициализацию элементов матрицы случайными числами, вывод матрицы, нахождение максимального и минимального элементов, а также вывод информации об объекте. |
| 8. | Класс «Прямоугольный треугольник». Реализовать ввод и вывод полей данных, вычисление гипотенузы, площади и периметра, а также вывод информации об объекте. |
| 9. | Класс «Отрезок». Реализовать ввод и вывод полей данных (координаты начала и координаты конца отрезка), вычисление длины, расстояний начала и конца отрезка от начала координат, а также вывод информации об объекте. |
| 10. | Класс «Цилиндр». Реализовать ввод и вывод полей данных, вычисление объема, площади поверхности, а также вывод информации об объекте. |
| 11. | Класс «Ромб». Реализовать ввод и вывод полей данных (диагонали ромба), вычисление площади, периметра, а также вывод информации об объекте. |
| 12. | Класс «Точка в пространстве». Реализовать ввод и вывод полей данных, вычисление расстояния до введенной пользователем точки, расстояния от начала координат, а также вывод информации об объекте. |
| 13. | Класс «График y=3x+5». Реализовать ввод и вывод полей данных, вычисление интеграла функции от a до b (вводятся пользователем), длины отрезка функции от (a, y(a)) до (b, y(b)), а также вывод информации об объекте. |
| 14. | Класс «Матрица M×N». Реализовать инициализацию элементов матрицы случайными числами, вывод транспонированной матрицы, нахождение среднего арифметического всех элементов, а также вывод информации об объекте. |
| 15. | Класс «Отрезок в пространстве». Реализовать ввод и вывод полей данных (координаты начала и координаты конца отрезка), вычисление длины, расстояний начала и конца отрезка от начала координат, а также вывод информации об объекте. |
| 16. | Класс «График y=x-10». Реализовать ввод и вывод полей данных, вычисление интеграла функции от a до b (вводятся пользователем), длины отрезка функции от (a, y(a)) до (b, y(b)), а также вывод информации об объекте. |
| 17. | Класс «Матрица M×N». Реализовать инициализацию элементов матрицы случайными числами, вывод транспонированной матрицы, нахождение и вывод среднего арифметического элементов в каждом столбце. |
| 18. | Класс «Куб». Реализовать ввод и вывод полей данных, вычисление объема, площади поверхности, длины диагонали, а также вывод информации об объекте. |
| 19. | Класс «Сфера». Реализовать ввод и вывод полей данных, вычисление объема, диаметра и площади поверхности, а также вывод информации об объекте. |
| 20. | Класс «Точка в пространстве». Реализовать ввод и вывод полей данных, вычисление расстояния до введенной пользователем точки, расстояния от начала координат, а также вывод информации об объекте. |
| 21. | Класс «График y=x». Реализовать ввод и вывод полей данных, вычисление интеграла функции от a до b (вводятся пользователем), длины отрезка функции от (a, y(a)) до (b, y(b)), а также вывод информации об объекте. |
| 22. | Класс «Точка в пространстве». Реализовать ввод и вывод полей данных, вычисление расстояния до введенной пользователем точки, расстояния от начала координат, а также вывод информации об объекте. |
| 23. | Класс «График y=-x». Реализовать ввод и вывод полей данных, вычисление интеграла функции от a до b (вводятся пользователем), длины отрезка функции от (a, y(a)) до (b, y(b)), а также вывод информации об объекте. |
| 24. | Класс «Цилиндр». Реализовать ввод и вывод полей данных, вычисление объема, площади поверхности, а также вывод информации об объекте. |
| 25. | Класс «Отрезок в пространстве». Реализовать ввод и вывод полей данных (координаты начала и координаты конца отрезка), вычисление длины, расстояний начала и конца отрезка от начала координат, а также вывод информации об объекте. |

**4. Задания для самостоятельной работы**

1. Опишите общепринятый подход к назначению модификаторов видимости переменным-членам класса и методам-членам класса.

2. Опишите последовательность действий при вызове метода класса. Как обратиться к переменной класса?

3. Охарактеризуйте известные модификаторы доступности членов класса.

4. Поясните процесс создания объекта класса.

**Вопросы к лабораторной работе**

1. Какие модификаторы видимости можно использовать в объявлении класса?
2. Как выполнялось индивидуальное задание?
3. Чем класс отличается от объекта?

**Лабораторная работа 17**

**Конструктор класса. Перегрузка конструкторов класса. Часть 1**

**Цель работы**: понять принципы работы конструктора.

Задачи лабораторной работы:

– научиться объявлять конструктор класса;

– научиться создавать перегруженные конструкторы.

**1. Теоретическая часть**

Конструктор предназначен для *инициализации объекта*. Он вызывается автоматически при создании объекта класса с помощью операции new.

*Имя конструктора* совпадает с *именем класса*.

*Свойства* конструкторов:

□ Конструктор ***не возвращает значение***, даже типа **void**.

□ Класс может иметь несколько конструкторов с разными параметрами для разных видов инициализации.

□ Если программист не указал ни одного конструктора или какие-то поля не были инициализированы, полям значимых типов присваивается нуль, полям ссылочных типов – значение null.

□ Конструктор, вызываемый без параметров, называется ***конструктором по умолчанию***.

Можно задавать *начальные значения полей класса при описании класса* (например, листинг 1). Это удобно в том случае, когда *для всех экземпляров класса начальные значения некоторого поля одинаковы*.

Если же при создании объектов требуется *присваивать полю разные значения*, это следует делать в конструкторе.

**Пример**. В классе Demo используем *конструктор*, *поля*сделаем*закрытыми*.

В программе создаются *два объекта с различными значениями полей*.

Листинг 1 – Класс с конструктором

using System;

namespace ConsoleApplication1

{class Demo

{public Demo(int a, double у) //конструктор с параметрами

{this.a = a;

this.у = у;

}

public double Gety() // метод получения поля у

{return у;

}

int a; // закрытые переменные

double у;

}

class Program

{static void Main(string[] args)

{Demo a = new Demo(300, 0.002); // вызов конструктора

Console.WriteLine(a.Gety()); // результат: 0.002

Demo b = new Demo(1, 5.71); // вызов конструктора

Console.WriteLine(b.Gety()); // результат: 5.71

}

}

}

В примере Demo(int a, double у) – конструктор, имя которого совпадает с именем класса (Demo), принимающий два параметра: целочисленный и вещественного типа.

Понятие перегрузки является проявлением принципа полиморфизма объектного подхода.

Конструктор – это метод класса, который не возвращает значения и имеет то же самое имя, что и класс. Если конструктор класса не определен программистом явно, то компилятор создаст конструктор по умолчанию.

Конструкторы подчиняются тем же правилам перегрузки, что и все методы.

В C# поддерживается перегрузка методов – то есть может существовать несколько версий одного метода, но с разными сигнатурами (методы отличаются количеством и/или типом параметров).

Чтобы перегрузить метод, просто объявляются методы с одинаковыми именами, но разными сигнатурами.

**2. Материалы и оборудование**

Для выполнения лабораторной работы необходимо использовать:

1) персональный компьютер;

2) программное обеспечение:

- операционную систему windows 7 и выше,

- Microsoft Visual Studio 2010 и выше.

**3. Задания к лабораторной работе**

1. Изучите теоретическую часть к лабораторной работе.
2. Модифицируйте приложение, полученное в результате выполнения индивидуального задания предыдущей лабораторной работы.

Модификация сводится к следующему: необходимо объявить и

продемонстрировать использование 3-4 перегруженных конструкторов

класса.

1. Оформите отчет по лабораторной работе (Приложение 1).

**4. Задания для самостоятельной работы**

1. Дайте определение класса. Опишите процедуру создания и инициализации переменных и методов класса.

2. Опишите сущность принципа полиморфизма и его проявления в перегрузке методов.

3. Опишите особенности конструктора класса. Каким требованиям он должен соответствовать?

4. Поясните возможность перегрузки конструктора класса и причины, по которым часто прибегают к перегрузке конструкторов.

**Вопросы к лабораторной работе**

1. Что такое перегрузка конструктора класса?
2. Какие функции может выполнять конструктор класса?
3. Можно ли напрямую вызвать конструктор класса в программе?

**Лабораторная работа 18**

**Конструктор класса. Перегрузка конструкторов класса. Часть 2**

**Цель работы**: понять принципы работы конструктора.

Задачи лабораторной работы:

– научиться объявлять конструктор класса;

– научиться создавать перегруженные конструкторы.

**1. Теоретическая часть**

Часто бывает удобно, чтобы методы, реализующие один и тот же алгоритм для различных типов данных, имели ***одно и то же имя***.

Если имя метода является осмысленным и несет нужную информацию, это делает программу более понятной, поскольку для каждого действия требуется помнить только одно имя.

Использование нескольких методов с одним и тем же именем, но различными типами параметров называется ***перегрузкой*** *методов*.

Компилятор определяет, *какой именно метод требуется вызвать, по типу фактических параметров*. Этот процесс называется ***разрешением*** (resolution) перегрузки.

*Механизм* *разрешения* основан на достаточно сложном наборе правил, смысл которых сводится к тому, чтобы *использовать метод с наиболее подходящими аргументами и выдать сообщение, если такой не найдется*.

Допустим, имеется четыре варианта метода, определяющего наибольшее значение:

// Возвращает наибольшее из двух целых:

int max(int a, int b)

// Возвращает наибольшее из трех целых:

int max(int a, int b, int c)

// Возвращает наибольшее из первого параметра и длины второго:

int max (int a, string b)

// Возвращает наибольшее из второго параметра и длины первого:

int max (string b, int a)

...

Console.WriteLine(max(1, 2));

Console.WriteLine(max(1, 2, 3));

Console.WriteLine(max( 1, "2")):

Console.WriteLine(max( "1", 2));

При вызове метода max компилятор выбирает вариант метода, соответствующий типу передаваемых в метод аргументов.

В приведенном примере будут последовательно вызваны все четыре варианта метода.

*Перегрузка* методов – проявление *полиморфизма*. Программисту удобнее помнить одно имя метода и использовать его для работы с различными типами данных, а решение о том, какой вариант метода вызвать, возложить на компилятор.

**2. Материалы и оборудование**

Для выполнения лабораторной работы необходимо использовать:

1) персональный компьютер;

2) программное обеспечение:

- операционную систему windows 7 и выше,

- Microsoft Visual Studio 2010 и выше.

**3. Задания к лабораторной работе**

1. Изучите теоретическую часть к лабораторной работе, изложенную в методических указаниях.
2. Выполните индивидуальное задание по перегрузке конструкторов класса для варианта индивидуального задания из лабораторной работы №8.
3. Оформите отчет по лабораторной работе (Приложение 1).

**4. Задания для самостоятельной работы**

1. Поясните понятие перегрузки методов. Распространяется ли перегрузка на конструкторы класса?

2. Попробуйте реализовать вызов конструктора другим конструктором класса в своем разработанном классе.

**Вопросы к лабораторной работе**

1. Может ли один конструктор класса вызывать другой конструктор?
2. Имеет ли конструктор возвращаемое значение?
3. Как компилятор определяет, какой из перегруженных конструкторов нужно использовать при создании объекта класса?

**СПИСОК РЕКОМЕНДУЕМОЙ ЛИТЕРАТУРЫ**

***Основная литература***

1. Хорев, П. Б. Объектно-ориентированное программирование с примерами на С#: учебное пособие / П.Б. Хорев. – М.: ФОРУМ: ИНФРА-М, 2020. – 200 с. – (Высшее образование: Бакалавриат). – ISBN 978-5-00091-680-3. – Текст: электронный. – URL: https://znanium.com/catalog/product/1069921. – Режим доступа: по подписке.

2. Пруцков, А. В. Математическая логика и теория алгоритмов: Учебник / Пруцков А.В., Волкова Л.Л. – М.: КУРС, НИЦ ИНФРА-М, 2018. - 152 с.: - (Бакалавриат). - ISBN 978-5-906818-74-4. - Текст: электронный. - URL: https://znanium.com/catalog/product/956763. – Режим доступа: по подписке.

***Дополнительная литература***

3. Ашарина, И. В. Язык С++ и объектно-ориентированное программирование в С++. Лабораторный практикум: Учебное пособие для вузов / Ашарина И.В., Крупская Ж.Ф. – М.: Гор. линия-Телеком, 2016. - 232 с. – (ВО - Бакалавриат) ISBN . - Текст: электронный. - URL: https://znanium.com/catalog/product/973780. – Режим доступа: по подписке.

4. Бедердинова, О. И. Программирование на языках высокого уровня: учеб. пособие / О.И. Бедердинова, Т.А. Минеева, Ю.А. Водовозова. – М.: ИНФРА-М, 2019. – 159 с. – Текст: электронный. – URL: https://znanium.com/catalog/product/1044396. – Режим доступа: по подписке.

# Приложение 1

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| |  |  |  |  | | --- | --- | --- | --- | | |  | | --- | |  | | МИНОБРНАУКИ РОССИИ | | Федеральное государственное бюджетное образовательное учреждение  высшего образования  **«МИРЭА – Российский технологический университет»**  **РТУ МИРЭА**  **Филиал РТУ МИРЭА в г. Ставрополе** | | | | |
|  | | |
| Кафедра \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  *(название кафедры* *полностью)* | | |

|  |  |
| --- | --- |
| **ОТЧЕТ ПО ЛАБОРАТОРНЫМ РАБОТАМ** | |
| **по дисциплине** | |
| **«**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**»**  *(наименование дисциплины)* | |
| Выполнил студент группы \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  *(учебная группа)* |  |
| Принял \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  *должность, звание, ученая степень* | *Фамилия И.О.* |
|  | *Фамилия И.О.* |

|  |  |  |
| --- | --- | --- |
| Лабораторные работы выполнены | «\_\_»\_\_\_\_\_\_\_202\_\_ г. |  |
|  |  | *(подпись студента)* |
| «Зачтено» | «\_\_»\_\_\_\_\_\_\_202\_\_ г. |  |

*(подпись руководителя)*

Ставрополь 20\_\_