Л а б о р а т о р н а я р а б о т а № 2

ПРОГРАММИРОВАНИЕ АЛГОРИТМОВ ЛИНЕЙНОЙ СТРУКТУРЫ. ЦЕЛОЧИСЛЕННАЯ АРИФМЕИИКА, ОПЕРАЦИИ НАД ЦЕЛОЧИСЛЕННЫМИ ПЕРЕМЕННЫМИ.

Цель работы:

- Ознакомиться с целочисленными типами,
- Освоить работу целочисленной арифметики,
- Повторить операции ввода/вывода.

Краткие теоретические сведения Типы данных.

Числовые типы

В следующей таблице (2.1) представлены размеры и диапазоны целых типов, которые составляют подмножество простых типов.

Таблица 2.1

type	Диапазон	Размер
sbyte	От -128 до 127	8-разрядное знаковое целое число
byte	От 0 до 255	8-разрядное целое число без знака
char	от U+0000 до U+ffff	16-разрядный символ Юни-кода
short	От -32768 до 32767	16-разрядное знаковое целое число
ushort	От 0 до 65535	16-разрядное целое число без знака
int (Целочис-ленное значе-ние)	От -2 147 483 648 до 2 147 483 647	32-разрядное знаковое целое число
uint	От 0 до 4 294 967 295	32-разрядное целое число без знака
long	От -9,223,372,036,854,775,808 до 9,223,372,036,854,775,807	64-разрядное целое число со знаком
ulong	От 0 до 18 446 744 073 709 551 615	64-разрядное целое число без знака

Приведение типов

При работе с числовыми типами данных довольно часто приходится сталкиваться с “расширением” и с “сужением” типа. Для начала рассмотрим несколько примеров:

```
byte b1 = 100;
short s1 = b1; // расширение типа
Console.WriteLine($"byte value: {b1}, short value: {s1}");
```

При компиляции этого кода никаких ошибок не будет несмотря на то, что мы присваиваем переменную одного типа (*byte*) переменной другого типа (*short*) для целых

чисел это делать можно без последствий, так как в данном случае происходит расширение типа. Переменная типа *byte* может принимать значения из диапазона от -128 до 127, а *short* из диапазона от -32 768 до 32 767, что значительно превышает ограничения для *byte*. Таким образом, при присваивании значения переменной типа *byte*, переменной типа *short* не происходит потери данных. В обратную сторону это сделать не получится, если вы напишите код, приведенный ниже, то компиляция будет прервана с ошибкой:

```
short s2 = 100;
byte b2 = s2; // приведет к ошибке компиляции
```

Для того, чтобы такая операция могла быть выполнена, необходимо использовать явное приведение, тем самым, мы как бы говорим компилятору, что в курсе того, что делаем. Для явного приведения необходимо тип, к которому приводится значение переменной, указать перед ней в круглых скобках. Перепишем наш второй пример с использованием явного приведения:

```
short s2 = 100;
byte b2 = (byte)s2;
Console.WriteLine($"byte value: {b2}, short value: {s2}");
```

Но имейте ввиду, что такая операция, по своей сути, не является безопасной, так как в этом случае возможно переполнение, что приведет к получению результата, который скорее всего вы не ожидаете:

```
short s3 = 150;
short s4 = 150;
byte b3 = (byte)(s3 + s4);
Console.WriteLine($"Result: {b3}");
```

В результате на консоль будет выведено следующее:

```
Result: 44
```

Это произошло вследствие того, что сумма *s3* и *s4* равна 300, а максимальное значение, которое может храниться в *byte* – это 127.

Проверка переполнения

Если переполнение является критичным моментом для некоторого участка кода вашего приложения, то можете использовать проверку переполнения с помощью ключевого слова *checked*. Суть его работы заключается в том, что если в рамках контекста, обозначенного через *checked* происходит переполнение, то будет выброшено исключение *OverflowException*. Пример использования для одного оператора приведен ниже:

```
try
{
short s5 = 150;
short s6 = 150;
byte b4 = checked((byte)(s5 + s6));
}
catch(OverflowException)
{
Console.WriteLine("Overflow is detected!");
}
```

Если необходимо провести проверку для группы операторов, то используйте *checked* следующим образом:

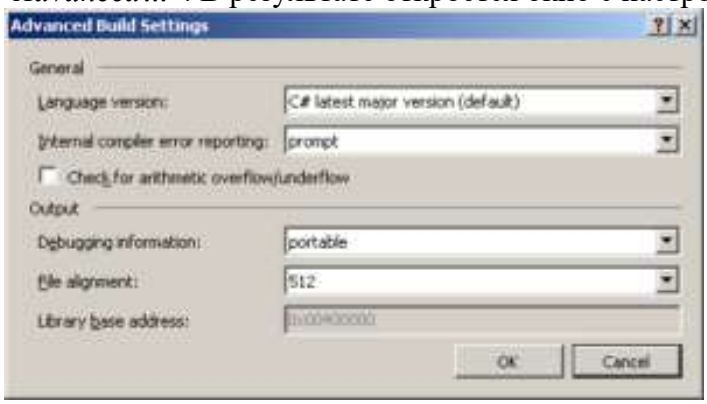
```
try
{
checked
```

```
{
short s5 = 150;
short s6 = 150;
byte b4 = checked((byte)(s5 + s6));
byte b5 = checked((byte)(s5 * 100));
}
}
catch(OverflowException)
{
Console.WriteLine("Overflow is detected!");
}
```

В результате выполнения любого из примеров, на консоль будет выведено сообщение:

Overflowisdetected!

Для того, чтобы выполнять такого типа проверку для всех вычислений в вашем приложении необходимо активировать эту опцию на этапе компиляции. Если вы работаете в *Visual Studio*, то зайдите в свойства проекта, перейти на вкладку *Build* и нажмите на кнопку "*Advanced…*". В результате откроется окно с настройками:



В нем можно поставить (или убрать) галочку в поле "*Checkforarithmeticoverflow/underflow*". Если установить галочку, то все вычисления будут проверяться на переполнение. В таком случае можно отдельно создавать блоки кода, в которых данная проверка производиться не будет, для этого используйте оператор *unchecked*:

```
unchecked
{
short s5 = 150;
short s6 = 150;
byte b4 = checked((byte)(s5 + s6));
}
```

***Класс* System.Convert**

Для приведения типов можно воспользоваться классом *System.Convert*, который содержит методы для приведения одного типа к другому. Приведем несколько примеров его использования. Вариант, когда приведение типа не приводит к переполнению:

```
short s7 = 100;
byte b6 = System.Convert.ToByte(s7);
Console.WriteLine($"byte value: {b6}, short value: {s7}");
```

Вариант, когда приведение типа приводит к переполнению, в этом случае будет выброшено исключение:

```
short s8 = 500;
byte b7 = System.Convert.ToByte(s8);
Console.WriteLine($"byte value: {b7}, short value: {s8}");
```

Арифметические операции

В таблице ниже перечислены арифметические операции, которые можно выполнять над числовыми типами данных в *C#.*

Операция	**Вид**	**Пример**
Сложение	a + b	Console.WriteLine($"3 + 4={(3 + 4)}")
Вычитание	a – b	Console.WriteLine($"7 – 5={(7 – 5)}");
Умножение	a * b	Console.WriteLine($"2 * 9={(2 * 9)}");
Деление	a / b	Console.WriteLine($"8 / 4={(8 / 4)}"); Console.WriteLine($"4.5 / 2={(4.5 / 2)}");
Остаток от деления	a % b	Console.WriteLine($"8 % 4={(8 % 4)}"); Console.WriteLine($"7 % 3={(7 % 3)}");
Постфиксный инкремент	a++	int n1 = 5; Console.WriteLine($"n1++: {n1++}"); Console.WriteLine($"n1 = {n1}");
Префиксный инкремент	++a	int n2 = 5; Console.WriteLine($"++n2: {++n2}"); Console.WriteLine($"n2 = {n2}");
Постфиксный декремент	a–	int n3 = 9; Console.WriteLine($"n3–: {n3–}"); Console.WriteLine($"n3 = {n3}");
Префиксный инкремент	–a	int n4 = 9; Console.WriteLine($"–n4: {–n4}"); Console.WriteLine($"n4 = {n4}");

Составное присваивание	a += b
	a -= b
	a *= b
	a /= b
	a % b
		int d1 = 10;
		int d2 = 3;
		d1 += d2;
		Console.WriteLine($"d1 += d2: {d1}");
		d1 -= d2;
		Console.WriteLine($"d1 -= d2: {d1}");
		d1 *= d2;
		Console.WriteLine($"d1 *= d2: {d1}");
		d1 /= d2;
		Console.WriteLine($"d1 /= d2: {d1}");
		d1 %= d2;
		Console.WriteLine($"d1 %= d2: {d1}");

Из перечисленных выше арифметических операций, обратите внимание на операции инкремента и декремента. При использовании постфиксного варианта, вначале происходит возврат значения переменной, а потом выполнение операции, для префиксного наоборот.

Практическая часть.

Используя ТОЛЬКО операции целочисленной арифметики составьте код, проводящий вычисления.

1. Целой переменной s присвоить сумму цифр трехзначного целого числа k.
2. Идет k-я секунда суток. Определить, сколько полных часов (h) и полных минут (m) прошло к этому моменту (например, h=3 и m=40, если k=13257 =3*3600+40*60+57).
3. Присвоить целой переменной h третью от конца цифру в записи положительного целого числа k (например, если k=130985, то h=9).
4. Дано натуральное число n (n>99). Определить число сотен в нем.
5. Дано натуральное число n (n≤99). Выяснить куб суммы цифр числа n.
6. Данонатуральноечислоn(n≤100).Чемуравнасумма цифр числа?
7. Данонатуральноечислоn(n≤100).Найтипоследнююцифру числа n.
8. Дано натуральное число n (n≤100). Найти первую цифру числа n.
9. Впредположении,чтоn>10,найтипредпоследнююцифру числа n.
10. Определите число, полученное выписыванием в обратном порядке цифр заданного целого трёхзначного числа
11. Напишите программу, которая считывает два целых числа a и b и выводит наибольшее значение из них. Числа — целые от 1 до 1000.При решении задачи можно пользоваться только целочисленными арифметическими операциями +, -, *, /, %, =. Нельзя пользоваться нелинейными конструкциями: ветвлениями, циклами, функциями.
12. Дано целое десятичное число. Найдите число десятков в его десятичной записи.
13. Пирожок в столовой стоит a рублей и b копеек. Определите, сколько рублей и копеек нужно заплатить за n пирожков.
14. В школе решили набрать три новых математических класса. Так как занятия по математике у них проходят в одно и то же время, было решено выделить кабинет для каждого класса и купить в них новые парты. За каждой партой может сидеть не больше двух учеников. Известно количество учащихся в каждом из трёх классов. Сколько всего нужно закупить парт чтобы их хватило на всех учеников? Программа получает на вход три целых десятичных числа: количество учащихся в каждом из трех классов.
15. Занятия в школе начинаются в 9:00. Продолжительность урока — 45 минут, перемены - 10 минут. На вход принимается номер урока, а выводится время, в которое он заканчивается (часы и минуты отдельно). Пример вывода:

```
Урок №2 заканчивается в 10 часов 40 минут
```

16. Доработайте код задачи № 15 таким образом, чтобы он запрашивал время начала занятий (минуты и часы отдельно) и номер урока, а далее также рассчитывал время окончания уроков.
17. Пользователь вводит число и систему счисления этого числа. Программа переводит число в десятичную, двоичную, восьмеричную и шестнадцетеричную системы счисления с использованием стандартных функций. Пример вывода:

```
Введитечисло:01100010
Укажитесистемусчисления:2
Числовдесятичнойсистемесчисления:98
Числовдвоичнойсистемесчисления:0b1100010
Числовввосьмеричнойсистемесчисления:0o142
Числовшестнадцатеричнойсистемесчисления:0x62
```

18. Даны значения двух моментов времени, принадлежащих одним и тем же суткам: часы, минуты и секунды для каждого из моментов времени. Известно, что второй

момент времени наступил не раньше первого. Определите, сколько секунд прошло между двумя моментами времени.

19. Дано целое число n. Выведите следующее за ним четное число.
20. Дано натуральное число n. Найти сумму первой и последней цифры этого числа.
21. 2. Дано натуральное число n. Переставить местами первую и последнюю цифры этого числа.
22. В двухзначное число вписать ноль в середину и получить трехзначное число.

Контрольные вопросы

1. Какие целочисленные типы данных вам известны?

2.Перечислите основные операции в языке программирования C#.

3.Перечислите операции ввода.

4.Перечислите операции вывода