Основы языка C++

Типы данных языка C++
Переменные
Константы

Алфавит и идентификаторы

В языке C++ используются:

- латинские заглавные буквы (A … Z);
- латинские строчные буквы (a … z);
- арабские цифры (0, 1, …, 9);
- специальные знаки (+, - , *, / и др.).

Идентификатором называется любая последовательность букв, символа подчеркивания и цифр, начинающаяся с буквы или символа подчеркивания (т.е. имя переменной не может начинаться с цифры).

Правила именования переменных

- язык C++ является регистрозависимым, т.е. Var и var – это две разные переменные;
- большинство программистов следуют негласному правилу использования в названиях переменных только прописных букв (или букв нижнего регистра);
- некоторые программисты употребляют в именах как строчные, так и прописные буквы: IntVar или dataCount;
- имена, состоящие только из строчных букв (или букв верхнего регистра), иногда применяются для обозначения констант (например, const double PI);
- однобуквенные имена (i или j) рекомендуется использовать только для временных переменных, таких как счетчики циклов;
- желательно, чтобы имя переменной отражало ее смысл; например, имя farengeit_temp является более предпочтительным, чем ft или t;
- в качестве имен переменных нельзя использовать ключевые слова языка C++ (например, int, class, if, while и др.).

Примеры объявления переменных

```
int a = 10, b;
float fvar = 5.14578;
char ch = 'A';
bool bvar = true;
const char* str = "Строка";
```

Константы

В языке C++ предусмотрены два типа констант: литеральные и символьные.

Литеральная константа – это значение, непосредственно вводимое в программе.

Например,

```
delta = gamma * 100;
```

здесь целое число 100 – это литеральная константа.

Символьная константа – это именованная константа (т.е. константа, представленная именем).

Например,

```
const int BITS_IN_WORD = 32;
```

Очевидно, что второй вариант задания константы более предпочтителен, т.к. если по каким-либо причинам потребуется изменить ее значение, то для этого будет достаточно изменить только одно место в программе.

Стандартные типы данных языка C++

Название типа	Нижняя граница диапазона	Верхняя граница диапазона	Точность	Размер в байтах
bool	False	True	Нет	1
char	–128	127	Нет	1
short	–32 768	32 767	Нет	2
int	–2 147 483 648	2 147 483 647	Нет	4
long	–2 147 483 648	2 147 483 647	Нет	4
float	$3.4*10^{-38}$	$3.4*10^{38}$	7	4
double	$1.7*10^{-308}$	$1.7*10^{308}$	15	8

Беззнаковые целые типы языка C++

Названив	Нижняя граница диапазона	Вврхняя граница диапвзона	Размвр в байтах
unsigned char	0	255	1
unsigned short	0	65 535	2
unsigned int	0	4 294 967 295	4
unsigned long	0	4 294 967 295	4

Константы

В языке C++ предусмотрены два типа констант: литеральные и символьные.

Литеральная константа – это значение, непосредственно вводимое в программе.

Например,

```
delta = gamma * 100;
```

здесь целое число 100 – это литеральная константа.

Символьная константа – это именованная константа (т.е. константа, представленная именем).

Например,

```
const int BITS_IN_WORD = 32;
```

Очевидно, что второй вариант задания константы более предпочтителен, т.к. если по каким-либо причинам потребуется изменить ее значение, то для этого будет достаточно изменить только одно место в программе.

Иерархия типов языка C++

Тип данных	Старшинство
long double	Высший
double	
float	
long	
int	
short	
char	Низший

Основные операции для целых чисел

Знак операции	Выполнение действия
=	Присваивание
+	Сложение
–	Вычитание
*	Умножение
/	Деление
%	Деление по модулю

Знак операции	Выполнение действия
==	Равенство
!=	Неравенство
>	Больше
<	Меньше
>=	Больше или равно
<=	Меньше или равно

Особенности проведения операций с целыми числами

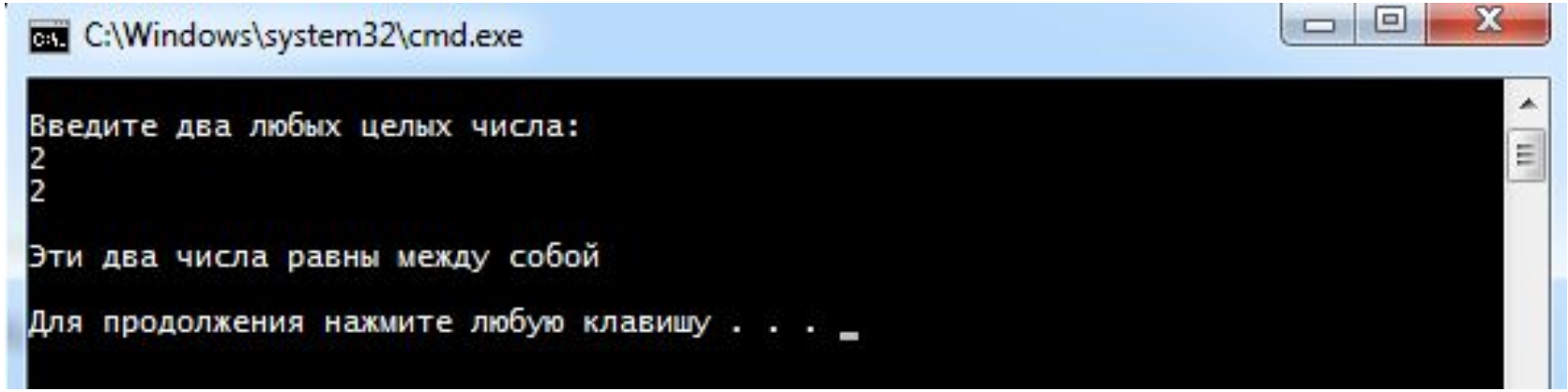
- допустимо смешивать в выражении различные целые типы. Например, операция $x + y$, где x – переменная типа *short*, а y – переменная типа *long*, будет корректной. При выполнении такой операции значение переменной x преобразуется к типу *long*. Такое преобразование можно произвести всегда без потери в точности, т.е. мы не теряем при этом никаких значащих цифр.
- общее правило преобразования целых типов состоит в том, что более короткий тип при вычислениях преобразуется в более длинный;
- только при выполнении операции присваивания длинный тип может преобразовываться в более короткий.

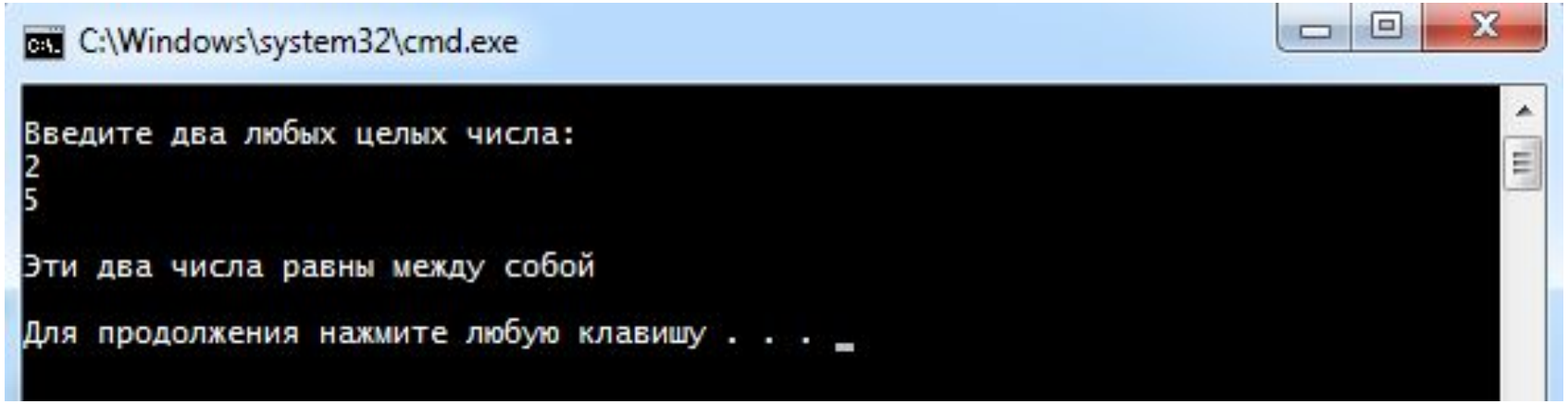
Замечание. Не путайте операцию присваивания (=) и операцию равенства (==).

Пример неправильного использования операции равенства

```
(Global Scope)
//Пример неправильного использования операции равенства
#include <iostream>
#include <windows.h>

using namespace std;

int main()
{
    SetConsoleOutputCP(1251);

    int var1, var2; //Объявление целочисленных переменных

    cout << "\nВведите два любых целых числа: " << endl; //Ввод целых чисел с экрана
    cin >> var1;
    cin >> var2;

    if (var1 = var2) //Проверка на равенство введенных чисел
    cout << "\nЭти два числа равны между собой " << endl << endl;  //Вывод результатов сравнения введенных чисел
    else
    cout << "\nЭти два числа не равны между собой " << endl << endl;
}
```

Пример неправильного использования операции равенства

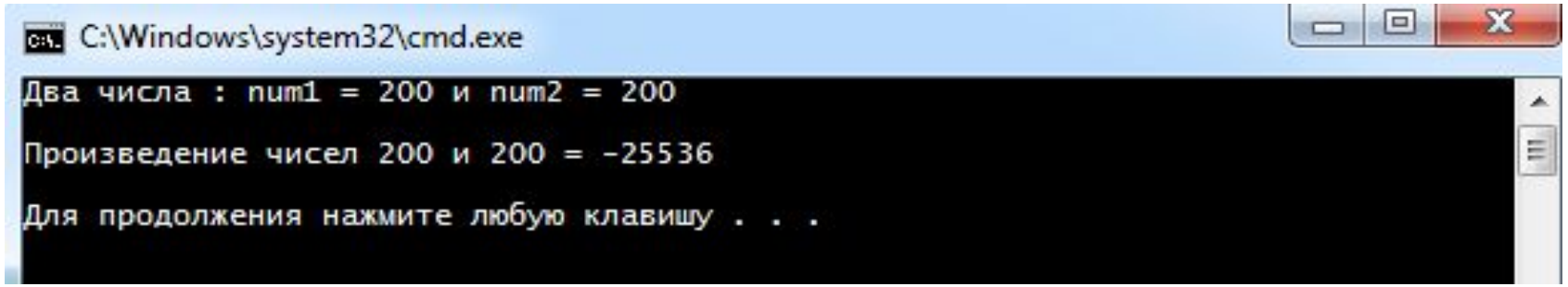




Пример переполнения

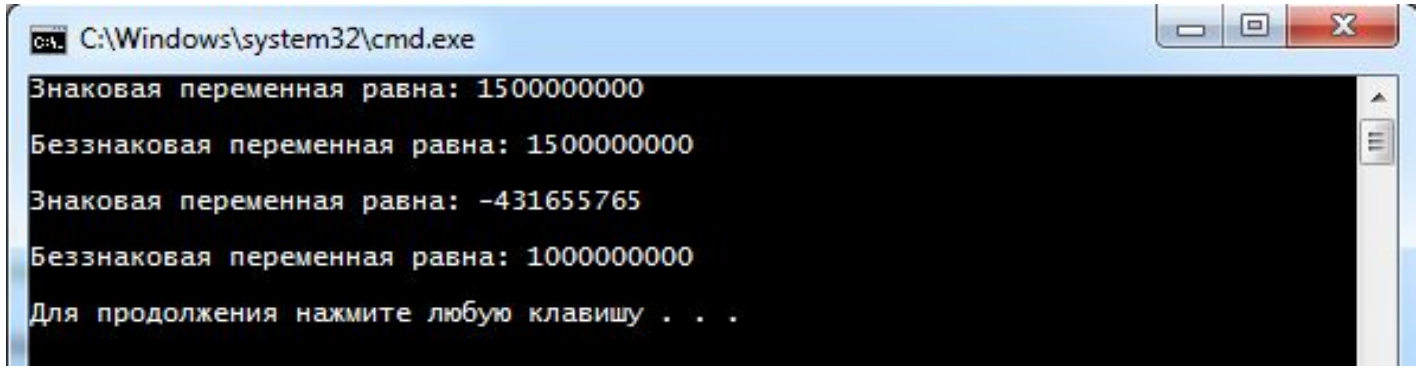
(Global Scope) main()

```
//Пример переполнения
#include <iostream>
#include <windows.h>

using namespace std;

int main()
{
    SetConsoleOutputCP(1251);

    short res;                    //Переменная типа short
    short num1 = 200, num2 = 200 ; //Две переменные типа short

    res = num1 * num2;

    cout << "Два числа : num1 = " << num1 << " и num2 = " << num2 << endl << endl;  //Вывод двух чисел
    cout << "Произведение чисел " << num1 << " и " << num2 << " = " << res << endl << endl;  //Вывод результата
}
```



Пример использования беззнаковой переменной

```
//Пример использования знаковых/беззнаковых переменных
#include <iostream>
#include <windows.h>

using namespace std;

int main()
{
    SetConsoleOutputCP(1251);

    int signedVar = 1500000000;           //Знаковая переменная
    unsigned int unsignedVar = 1500000000;//Беззнаковая переменная

    cout << "Знаковая переменная равна: " << signedVar << endl << endl;     //Вывод начального значения
    cout << "Беззнаковая переменная равна: " << unsignedVar << endl << endl;//Вывод начального значения

    signedVar = signedVar * 2 / 3;      //Выход за границы диапазона
    unsignedVar = unsignedVar * 2 / 3;  //Вычисление внутри диапазона

    cout << "Знаковая переменная равна: " << signedVar << endl << endl;     //Вывод некорректного результата
    cout << "Беззнаковая переменная равна: " << unsignedVar << endl << endl;//Вывод правильного результата
}
```

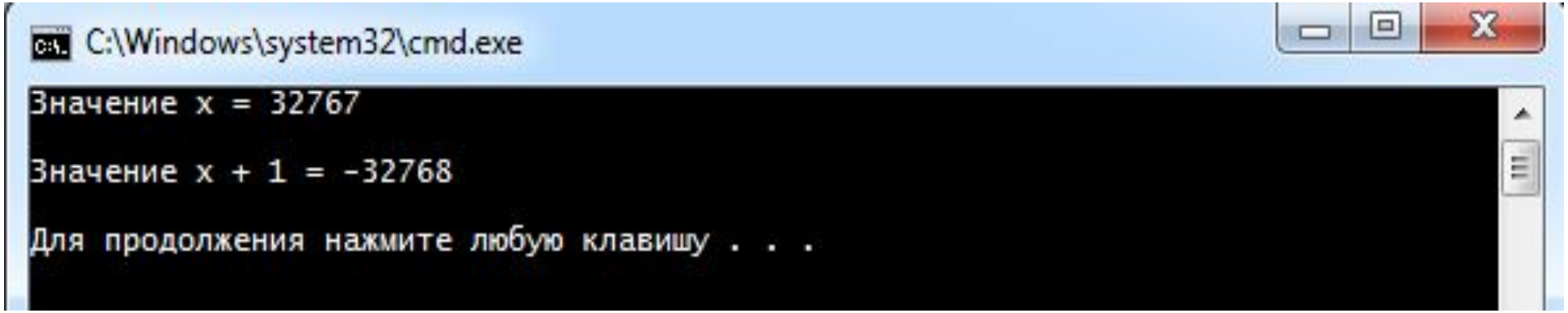


Пример эффекта «заворачивания» (wrapping)

```
//Пример заворачивания (wrapping)
#include <iostream>
#include <windows.h>

using namespace std;

int main()
{
    SetConsoleOutputCP(1251);

    short x = 32767;                    //Максимальное значение для переменной типа short
    cout << "Значение x = " << x << endl << endl;     //Вывод значения x
    x = x + 1;                          //Увеличение значения переменной x на 1
    cout << "Значение x + 1 = " << x << endl << endl;     //Вывод нового значения переменной x
}
```

Вопрос: чему равно значение x + 1 ?

Пример эффекта «заворачивания» (wrapping)



Особенности проведения операций с вещественными числами

Для представления ***вещественных чисел*** в языке C++ существует три типа: float – с одинарной точностью, double – с двойной точностью и long double – с расширенной точностью.

Точность представления чисел составляет 7 десятичных значащих цифр для типа float, 15 значащих цифр для типа double и 19 значащих цифр для типа long double.

Вещественные числа записываются либо в виде десятичных дробей, например, 1.3, 3.14159, 0.0005, либо в виде мантиссы и экспоненты: 1.2E0, 0.12e1.

Замечание. По умолчанию вещественная константа принадлежит к типу double.

Особенности проведения операций с вещественными числами

- Для вещественных чисел определены все *стандартные арифметические операции* за исключением операции нахождения остатка от деления.
- Если арифметическая операция применяется к двум вещественным числам разных типов, то менее точное число преобразуется к более точному, т.е. float преобразуется к double и double преобразуется к long double. Очевидно, что такое преобразование всегда можно выполнить без потери точности.
- Если вторым операндом в операции с вещественным числом является целое число, то целое число преобразуется в вещественное представление.

Правила преобразования типов

- при использовании в операциях данных разного типа для корректности проведения вычислений происходит неявное преобразование типов по следующему правилу: менее точное число преобразуется к более точному, например

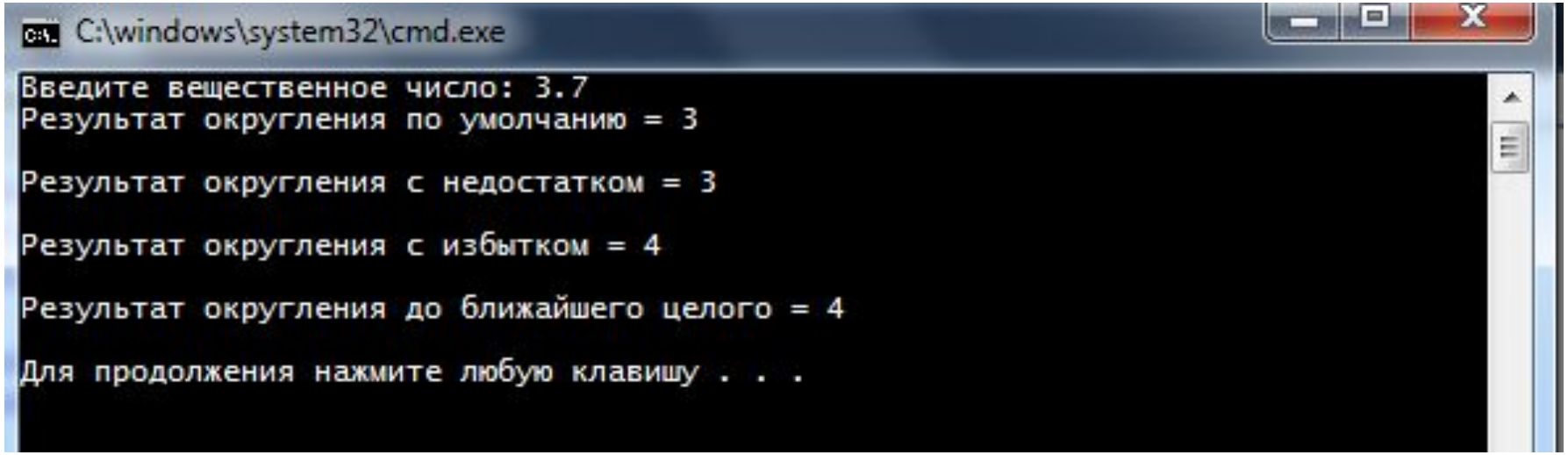
int → float, float → double

- при автоматическом преобразовании (преобразование по умолчанию) вещественного числа в целое число, у вещественного числа просто отбрасывается дробная часть: 15,745 → 15;
- для корректного округления вещественных чисел в языке C++ используются функции floor() (для округления с недостатком: floor(15,745) = 15) и ceil() (для округления с избытком: ceil(15,745) = 16), подключаемые с помощью заголовочного файла **math.h**.

Примеры преобразования типов

```
//Округление вещественных чисел
#include <iostream>
#include <windows.h>
#include <math.h>

using namespace std;
int main()
{
    SetConsoleOutputCP(1251);

    float a;                                    // вещественная переменная
    int b, c, d;                                // целая переменная
    cout << "Введите вещественное число: ";     //Ввод числа
    cin >> a;
    b = a;                                                  //Округление по умолчанию
    cout << "Результат округления по умолчанию = " << b << endl << endl;
    b = floor(a);                                           //Округление с недостатком
    cout << "Результат округления с недостатком = " << b << endl << endl;
    b = ceil(a);                                            //Округление с избытком
    cout << "Результат округления с избытком = " << b << endl << endl;
    b = floor(a + 0.5);                                     //Округление до ближайшего целого
    cout << "Результат округления до ближайшего целого = " << b << endl << endl;
}
```

Примеры преобразования типов



Особенности типа char

В языке C++тип данных char относится к целочисленному, т.е. любому символу в языке C++ соответствует некоторое целое число.

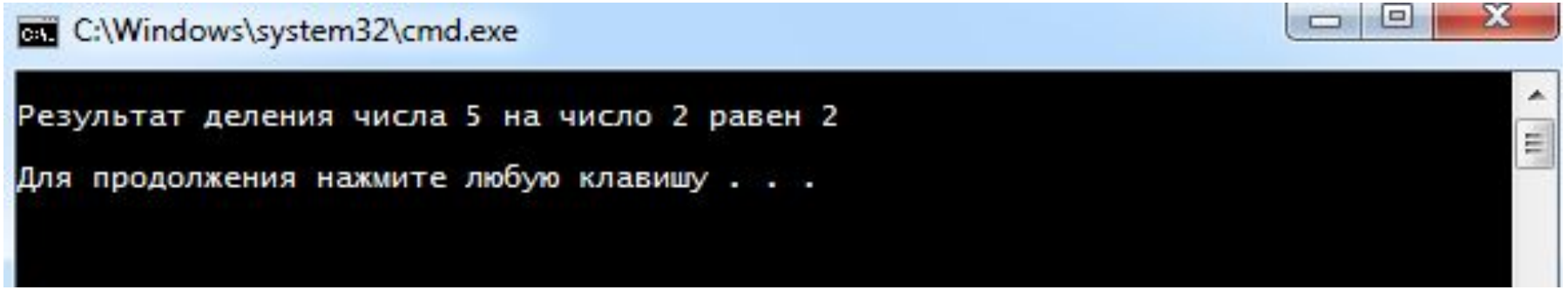
```
(Global Scope)                                                    main()
//Пример соответствия символов и целых чисел
#include <iostream>
#include <windows.h>

using namespace std;

int main()
{
    SetConsoleOutputCP(1251);
    char char_var1, char_var2;     //Объявление символьных переменных
    int int_char_var1, int_char_var2; //Объявление целочисленных переменных

    cout << "\nВведите какой-нибудь символ: "; //Ввод символа с экрана
    cin >> char_var1;

    int_char_var1 = char_var1; //Получение его целочисленного значения

    cout << "\nВы ввели символ: " << char_var1 << endl << endl  //Вывод введенного символа
         << "Его целочисленное значение = " << int_char_var1 << endl << endl;   //Вывод его целочисленного значения

    cout << "\nВведите какое-нибудь целое число: "; //Ввод целого числа с экрана
    cin >> int_char_var2;

    char_var2 = int_char_var2; //Получение соответствующего символа

    cout << "\nВы ввели целое число: " << int_char_var2 << endl << endl  //Вывод введенного целого числа
         << "Этому числу соответствует символ " << char_var2 << endl << endl;   //Вывод соответствующего символа
}
```

```
C:\Windows\system32\cmd.exe

Введите какой-нибудь символ: A

Вы ввели символ: A

Его целочисленное значение = 65


Введите какое-нибудь целое число: 245

Вы ввели целое число: 245

Этому числу соответствует символ x
```

Примеры потери точности при неявном преобразовании

(Global Scope) main()

```
//Потеря точности при делении целых чисел
#include <iostream>
#include <windows.h>

using namespace std;

int main()
{
    SetConsoleOutputCP(1251);

        int a, b, res;
        a = 5;
        b = 2;
        res = a/b;
        cout << "\nРезультат деления числа " << a << " на число " << b << " равен " << res << endl <<endl;

}
```



Правила явного преобразования типов

Для реализации явного преобразования типа переменной к другому типу перед именем переменной необходимо указать в круглых скобках тот тип данных, к которому необходимо преобразовать эту переменную:

Prim_1_4_2.cpp*

(Глобальная область) main()

```
#include <iostream>
using namespace std;
void main()
{
int a = 5;
float z1, z2;
z1 = a / 2;          //  z1 = 5/2 = 2 - результат неверный
z2 = (float) a / 2; //  z2 = 5.0/2 = 2.5 - явное преобразование - результат верный
cout << "\t z1 = " << z1 << " size = "<< sizeof(z1) << endl;
cout << "\t z2 = " << z2 << " size = "<< sizeof(z2) << endl;
}
```