Занятие 2

Работа с данными

Встроенные целочисленные типы (`int`, `short int`, `long int`, `long long int`, `unsigned`, `char`*). Символьные константы из файла* `climits`*. Числовые литералы целочисленных типов (десятичные, восьмеричные, шестнадцатеричные). Использование квалификатора* `const`*. Встроенные типы с плавающей точкой (*`float`, `double`*). Арифметические операции. Автоматические и принудительные преобразования типов.*

Пример 1. *Ограничения целых чисел.*
Помимо основного типа `int` существует и несколько разновидностей, которые обозначаются как `short int`, `long int`, `long long int`. При декларировании таких типов слово `int` можно не писать. Все эти типы отличаются количеством памяти, выделяемой для хранения переменных. Меньше всего выделяется для хранения переменных типа `short`, а больше всего для типа `long long`. Если целочисленная переменная в программе принимает только неотрицательные значения, то можно ее определить с использованием ключевого слова `unsigned`. Комманда `sizeof()` возвращает количество байт, выделяемых для хранения данного типа. Аргументом команды `sizeof()`может быть как название типа, так и имя конкретной переменной.

Для каждого типа данных существуют минимальные и максимальные значения, которые может принимать переменная этого типа. Эти значения определяются не только типом, но и операционной системой. Например, для хранение переменных типа `int` может выделяться различный размер памяти на разных операционных системах, а поэтому предельные значения переменных этого типа могут отличаться. Этот может приводить к проблемам при переносе программ между различными операционными системами.

Максимальные и минимальные значения переменных прописаны в файле `climits` в виде значений именнованных постоянных. Например, значение постоянной `INT_MAX` – максимальное значение переменной типа `int`, `INT_MIN` – минимальное значение переменной типа `int`, `SHRT_MIN` – минимальное значение переменной типа `short int`, `UINT_MAX` – максимальное значение переменной типа `unsigned int`. Минимальные значения переменных типа `unsigned` всегда равны 0 и не хранятся в виде постоянных.

Если значение целочисленной переменной равно максимальному значению для данного типа, то добавление 1 приводит к тому, что ее значение становится минимальным.

```
#include <iostream>
#include <climits>
int main(void){
  using namespace std;
  // Операция sizeof выдает размер типа или переменной
  cout << "sizeof(int)="               << sizeof(int)               << endl;
  cout << "sizeof(signed int)="        << sizeof(signed int)        << endl;
  cout << "sizeof(short)="             << sizeof(short)             << endl;
  cout << "sizeof(short int)="         << sizeof(short int)         << endl;
  cout << "sizeof(signed short int)=" << sizeof(signed short int) << endl;

  // Определение пределов
  int n = INT_MAX;
  cout << "sizeof(n) = " << sizeof(n) << endl;
```

```
  cout << "n= "           << n          << endl;
  cout << "INT_MAX = "    << INT_MAX    << endl;
  cout << "INT_MIN = "    << INT_MIN    << endl;
  cout << "UINT_MAX = "   << UINT_MAX   << endl;

  // Переход через предел
  short s = SHRT_MAX;
  cout << "s= " << s << endl;
  s++;
  cout << "s= " << s << endl;
  return 0;
}
```

Пример 2. *Шестнадцатеричные и восьмеричные литералы. Манипуляторы* `dec`, `hex`, `oct`.
Помимо десятичной записи целые числа можно задавать в восьмеричном или шестнадцатеричном виде. Для обозначения восьмеричной записи ставится в начале `0`, для шестнадцатеричной записи ставится в начале `0x`. Например, запись

```
int i2=0x42;
```

означает, что целочисленной переменной `i2` присваивается значение $4 \times 16^1 + 2 \times 16^0 = 66$. При выводе на экран можно также использовать шестнадцатеричную и восьмеричную записи. Для изменения формата вывода на шестнадцатеричный используются команда `hex`, на восьмеричный – команда `oct`. Для возвращения формата к десятичной записи надо использовать `dec`.

```
#include <iostream>
int main(void){
  using namespace std;
  int i1 = 42;   // десятичный целочисленный литерал
  int i2 = 0x42; // шестнадцатеричный целочисленный литерал
  int i3 = 042;  // восьмеричный целочисленный литерал
  cout << "i1 = " << i1 << " (= 42 in decimal)\n";
  cout << "i2 = " << i2 << " (= 0x42 in hex)\n";
  cout << "i3 = " << i3 << " (= 042 in octal)\n";
  cout << "i2 = " << hex << i2 << " (hexadecimal 0x42)\n";
  cout << "i3 = " << oct << i3 << " (octal 042)\n";
  cout << dec;  // вернуться к десятичной системе
  cout << "i3 = " << i3 << "\n";
  return 0;
}
```

Пример 3. *Сравнение типов char и int.*
Тип `char` является тоже целочисленным, но при вводе и выводе на экран целочисленные значения переменных преобразуются в символы с использованием таблицы ASCII. Переменная типа `char` может хранить только один символ. При присвоении значения переменной можно использовать либо символ в одинарных кавычках, либо целое число. При выводе на экран значения переменной типа `char` будет печататься символ. Для вывода соответствующего целочисленного значения можно конвертировать тип переменной в `int`.

```cpp
#include <iostream>
#include <climits>
int main(void){
  using namespace std;
  char ch = 'M'; // присваивает ch код ASCII символа M
  int i = ch;    // сохраняет этот же код в int
  cout << "The ASCII code for " << ch << " is " << i << endl;
  char ch2 = 77; // можно использовать код
  cout << "ch2 = " << ch2 << ", code = " << (int) ch2 << "\n\n";

  /* Пределы для char */
  cout << "Maximum char value ="        << CHAR_MAX << endl;
  cout << "Minimum char value ="        << CHAR_MIN << endl;
  cout << "Number of bits in a char =" << CHAR_BIT << endl;
  cout << "sizeof(char) = " << sizeof(char) << endl;
  return 0;
}
```

Пример 4. *Использование операции* `%` *для нахождения остатка. Использование* `const`.
Операция `%` возвращает остаток от деления двух целых чисел. Ключевое слово `const` перед именем переменой означает, что значение переменной не будет изменяться. Все переменные со словом `const` должны инициализироваться.

```cpp
#include <iostream>
int main(void){
  using namespace std;
  const int inch_per_foot = 12; // дюймов в футе
  int length_inches = 67;       // некоторая длина в дюймах
  int feet = length_inches/inch_per_foot; // целое число футов
  int inches = length_inches % inch_per_foot; // остаток в дюймах

  cout << length_inches << "\" = " << feet << "\'" << inches << "\"\n";
  return 0;
}
```

Пример 5. *Вещественные типы (типы с плавающей точкой).*
Для работы с вещественными числами используются типы `float` и `double`. Переменные типа `float` имееют точность примерно 6 знаков, а типа `double` - около 15. Точность представления чисел влияет на точность математических операций с ними, как показано в примере ниже.

```cpp
#include <iostream>
int main(void){
  using namespace std;

  cout << fixed; // для использования записи с фиксированной точкой

  // выведем количество знаков после точки
  cout << "precision = " << cout.precision() << " digits" << endl;

  float    one = 10.0/3.0;  // about 6  digit precision
```

```
11    double  two = 10.0/3.0;  // about 15 digit precision
12
13    const float million = 1.0e6;
14    cout << "one = " << one << endl;
15    cout << "a million ones = " << million * one << endl;
16    cout << "and ten million ones = " << 10 * million * one << endl;
17    cout << "two = " << two << endl;
18    cout << "a million twos = " << million * two << endl;
19    return 0;
20 }
```

Результат выполнения примера 5:

```
precision = 6 digits
one = 3.333333
a million ones = 3333333.250000
and ten million ones = 33333332.000000
two = 3.333333
a million twos = 3333333.333333
```

Обратите внимание, что перемножение `million * one` содержит только 7 правильных цифр, так как обе переменные заданы как `float`. Перемножение `million * two` дает более точный результат, так как переменная `million` будет сначала конвертироваться в тип `double`, а затем перемножаться с переменной `two`, которая имеет тип `double`.

Задания

Задание 2.1. Определите размер памяти, максимальные и минимальные значения для типов `short int`, `unsigned short int`, `long int`, `unsigned long int`, `long long`. Определите, что получается с переменными типа `unsigned`, если их значения выходят за максимальные и минимальные пределы для данного типа.

Задание 2.2. Напишите программу, которая запрашивает десятичное число и выводит его на экран в восьмеричном и шестнадцатиричном виде.

Задание 2.3. Напишите программу, которая распечатывает на экран все возможные символьные значения переменной типа `char`. Используйте цикл.

Задание 2.4 Напишите программу, которая сначала запрашивает время, оставшееся до сессии, в неделях, днях и часах, затем преобразует его в недели и выводит в десятичном формате. После чего программа запрашивает продолжительность сессии в часах (используйте `unsigned`) и отображает в неделях, днях и часах. Для представления коэффициентов преобразования используйте символьные константы `const`. Каким будет результат, если ввести максимальное значение часов, которое допустимо для используемого типа переменной? Результат выполнения программы должен выглядеть следующим образом:

```
Enter a first period in weeks, days, and hours:
  number of weeks: 12
  number of days: 3
  number of hours: 21
```

```
12 weeks, 3 days, 21 hours = 12.5536 weeks
Enter a second period in hours: 750
750 hours = 4 weeks 3 days 6 hours
```