Лабораторная работа N 1

Использование потокового ввода-вывода в C++

1. Введение

В отличие от таких языков программирования как Pascal и Basic, в C++, как и в C, нет встроенных в язык средств ввода-вывода. Их и не нужно, поскольку такие средства можно просто создать на самом языке. Библиотека потокового ввода-вывода C++ предоставляет строгий и вместе с тем гибкий и эффективный способ символьного ввода и вывода целых, вещественных чисел и символьных строк.

Традиционно средства ввода-вывода были рассчитаны исключительно на небольшое число встроенных типов данных. Однако, в нетривиальных программах на C++ есть много пользовательских типов данных, для вывода которые недостаточно стандартных средств ввода-вывода. Следовательно, должна быть возможность как расширять стандартные средства ввода-вывода, так и создавать свои собственные.

Основная задача потоковых средств ввода-вывода - это процесс преобразования объектов определенного типа в последовательность символов и наоборот. Поэтому задача программиста сводится к описанию связи между объектом и строкой.

Целью лабораторной работы является получение практических навыков при работе с библиотекой потокового ввода-вывода языка C++. В работе рассматриваются базовые средства ввода-вывода, использование флагов форматирования, использование и создание собственных манипуляторов ввода-вывода.

2. Общие сведения

2.1 Потоки языка C++

В языке C для операций ввода-вывода используется стандартная библиотека stdio.h, но она имеет несколько недостатков:

-функции ввода-вывода сложны в употреблении, что приводит к частым ошибкам;

- затруднителен ввод-вывод абстрактных типов данных (например, структур – struct).

В C++ разработана новая библиотека ввода-вывода (iostream.h), написанная на C++, и использующая технологию объектно-ориентированного программирования. Библиотека iostream.h определяет 4 стандартных потока:

cin стандартный входной поток (stdin в C)

cout стандартный выходной поток (stdout в C)

cerr небуферизированный стандартный поток вывода сообщений об ошибках (немедленный вывод)

clog буферизованный поток вывода сообщений об ошибках (вывод происходит только после того, как наполнится буфер)

По умолчанию поток *cin* связан с клавиатурой , *cout* - с дисплеем, но они могут быть перенаправлены на другие устройства или на файловую систему.

Для выполнения операций ввода-вывода переопределены две операции поразрядного сдвига:

>> получить из входного потока

<< поместить в выходной поток

Ввод значений переменной может быть осуществлен следующим способом:

cin >> идентификатор;

По этому выражению из входного потока читается последовательность символов до пробельного символа, затем эта последовательность преобразуется к типу

«идентификатора» и получаемое значение помещается в «идентификатор». Вывод информации осуществляется с использованием потока cout:

cout << значение;

Здесь «значение» преобразуется в последовательность символов и выводится в выходной поток.

Пример 1 Программа выводящая сообщение Hello, world.

```
#include <iostream.h>
main()
 {
     cout << "Hello, world\n";
 }
```

Для того чтобы использовать операторы ввода-вывода, необходимо подключить библиотеку iostream.h, в которой определены операции, объекты и перегружены операции сдвига. Без этих описаний выражение `cout << "Hello, world\n"` не имело бы смысла. Операция `<<` ("поместить в") пишет свой первый аргумент во второй (в данном случае, строку `"Hello, world\n"` в стандартный поток вывода `cout`.

Возможно многократное назначение потоков:

cin >> ‘переменная1’ >> ‘переменная2’ >>...>> ‘переменная n’;

cout << ‘значение1’ << ‘значение2’ << ... << ‘значение n’;

При наборе данных на клавиатуре значения для такого оператора должны быть разделены пробельными символами (␣ , \n, \t).

Пример 2. Сравнение ввода-вывода в С и С++

Программа на С

```
# include <stdio.h>
main()
{ int n;
  char str[80];
  printf (“Ввести число\n”);
  scanf(“%d”,&n);
  printf (“Ввести строку\n”);
  gets(str);
  printf(“n=%d\n”,n);
  printf(“str=%s\n”,str);
}
```

Программа на С++

```
# include <iostream.h>
main()
{ int n;
  char str[80];
  cout << “Ввести число\n”;
  cin >> n;
  cout << “Ввести строку\n”;
  cin >> str;
  cout << “n=”<<n<<endl;
  cout << “str=”<<str <<endl;
}
```

2.3 Форматированный ввод/вывод

Для того чтобы организовать форматированый ввод и вывод , анологичный тому , что предоставляют пользователю функции printf() и scanf() в языке С++ используются два способа:

1) использование флагов форматирования (функций членов - класса *ios)*;

2) использование специальный функций, называемых манипуляторами.

2.3.1 Использование флагов форматирования.

Состояние правил форматного вывода в поток определяется состоянием флагов форматированого потока. В заголовочном файле iostream.h определено следущее перечисление , задающее флаги форматирования:

```
enum    {
        skipws    = 0x0001, // отбрасывание пробелов
        left      = 0x0002, // выравнивание по левому краю
        right     = 0x0004, // выравнивание по правому краю
        internal  = 0x0008, // заполнение пустых позиций
        dec       = 0x0010, // выдача в десятеричном формате
        oct       = 0x0020, // выдача в восьмеричном формате
        hex       = 0x0040, // выдача в шестнадцетиричном формате
        showbase  = 0x0080, // выдача основания системы счичления
        showpoint = 0x0100, // выдача позиции точки
        uppercase = 0x0200, // выдача в формате xx.xxxx Exx
        showpos   = 0x0400, // выдача знака у положительного числа
        scientific= 0x0800, // выдача в форме с фиксированной точкой (научный
формат)
        fixed     = 0x1000, // выдача в форме с плавающей точкой
        unitbuf   = 0x2000, // улучшенная выдача
        stdio     = 0x4000  // освобождение потока
};
```

Таблица 1. - Функции работы с флагами форматирования

Прототип функции	**Описание функции**
long setf(long flags)	Установка флагов **flags** в состояние включено (on) **Пример 1.** Чтобы установить флаг showbase в режим включено (on), используем оператор **stream.setf(ios::showbase);** здесь stream - тот конкретный поток (например cout) **Пример 2.** Для установки флагов можно использовать побитовые операции: **cout.setf(ios::left | ios::hex);**
unsetf(long flagsl)	Сброс флагов **flags** в состояние выключено(off). **Пример 1.** Чтобы сбросить флаг showbase в режим выключено(off), используем оператор **cout.unsetf(ios::showpos);**

long flags(void)	Возвращает текущее состояние флагов
	
	Пример 1. Запрос текущего состояния флагов
	`long fl;`
	`fl= cout.flags();`
	`cout << "Состояние флагов: "<< fl<<"\n";`

int width(int len)	Возвращает текущую ширину поля выдачи и устанавливает её ширину **len**.
char fill(char ch)	возвращает текущий символ заполнения и устанавливает новый символ заполнения **ch**
int precision(int num)	возвращает текущее значение десятичных знаков после точки и устанавливает новое значение **num**

Пример 3. Использование флагов форматирования.

```
#include <iostream.h>
#include <iomanip.h>
main(void)
{
  long fl;
```

```
  fl= cout.flags();
  cout << "Исходное состояние флагов: "<< fl<<"\n";

  cout.setf(ios::showpos);
  cout.setf(ios::scientific);
  cout << 123 << " " << 1.2345678 <<"\n";

  cout.setf(ios::hex | ios::showbase);
  cout.unsetf(ios::showpos);
  cout.width(20);
  cout.precision(10);
  cout << 123 << " " << 123.456 << " " << 1.2345678 << "\n";
  cout << "Новое состояние флагов: " << cout.flags() << "\n";
  cout.flags(fl);
  cout << "После востановления исходного состояния флагов: \n";
  cout << 123 << " " << 123.456 << 1.2345678 << "\n";
  return 0;
}
```

2.3.2 Использование манипуляторов

Для управления форматом выдачи из потока можно использовать специальные функции, называемые манипуляторами. Стандартными манипуляторами, доступ к которым можно получить, подключив файл iomanip.h приведены в таблице.

Таблица 2 – Виды манипуляторов.

Манипулятор	**Действие манипулятора**
`dec`	Десятичный формат
`endl`	Вывод символа '\n' и освобождение буфера
`ends`	Вывод NULL
`flush`	Освободить поток
`hex`	Шестнадцатеричный формат числа
`oct`	Установка основания 8-ой системы счисления
`resetioflags(long f)`	Отключить флаги , определённые в f
`setbase(int base)`	Установить основание системы счисления
`setfill(char ch)`	Установить символ заполнения
`setiosflags(long f)`	Установить режим on флагам , указанным в f
`setprecision(int p)`	Установить число цифр после десятичной точки
`setw(int w)`	Установить ширину поля выдачи
`ws`	Режим пропуска символов пробела

Пример 3 Использование стандартных манипуляторов

```
#include <iostream.h>
 #include <iomanip.h>
main(void)
{
  cout << setprecision(2) << 100.5375 << endl;
  cout << setw(20) << "MANIPULATORS \n";
  return 0;
}
```

2.3.3 Создание манипуляторов

Можно создать свою собственную функцию - манипулятор. Покажем, как сделать функцию манипулятор без параметров. Формат объявления функции – манипулятора вывода:

```
ostream &manip_name(ostream &stream)
{
  //Коды программы
  return stream;
}
```
Для создания функции - манипулятора ввода надо заметить поток **ostream** на **istream**.

Пример 4. Создание манипулятора с именем left10, который устанавливает выравнивание по левому краю и ширину поля выдачи 10 символов.

```
#include <iostream.h>
#include <iomanip.h>

ostream& left10(ostream &stream)
{
  stream.setf(ios::left);
  stream << setw(10);
  return stream;
}

main (void)
{
  cout << 12 << left10 <<15 << 17 <<"\n";
  return 0;
}
```

2.4 Контроль ошибок

Каждый поток (istream или ostream) имеет ассоциированное с ним состояние, и обработка ошибок и нестандартных условий осуществляется с помощью соответствующей установки и проверки этого состояния.

Поток может находиться в одном из следующих состояний:

```
enum stream_state { _good, _eof, _fail, _bad };
```

Если состояние **_good** или **_eof**, значит последняя операция ввода прошла успешно. Если состояние **_good**, то следующая операция ввода может пройти успешно, в противном случае она закончится неудачей. Другими словами, применение операции ввода к потоку, который не находится в состоянии **_good**, является пустой операцией. Если делается попытка читать в переменную v, и операция оканчивается неудачей, значение v должно остаться неизменным (оно будет неизменным, если v имеет один из тех типов, которые обрабатываются функциями членами istream или ostream). Отличие между состояниями **_fail** и **_bad** очень незначительно и представляет интерес только для разработчиков операций ввода. В состоянии **_fail** предполагается, что поток не испорчен и никакие символы не потеряны. В состоянии **_bad** может быть все что угодно.

Состояние потока можно проверять например так:

```
switch (cin.rdstate()) {
  case _good: // последняя операция над cin прошла успешно
        break;
  case _eof: // конец файла
        break;
  case _fail: // некоего рода ошибка форматирования, возможно, не слишком
плохая
        break;
  case _bad: // возможно, символы cin потеряны
        break;
  }
```

Для любой переменной z типа, для которого определены операции << и >>, копирующий цикл можно написать так:

```
while (cin>>z) cout << z << "\n";
```

Например, если z - вектор символов, этот цикл будет брать стандартный ввод и помещать его в стандартный вывод по одному слову (то есть, последовательности символов без пробела) на строку.

Когда в качестве условия используется поток, происходит проверка состояния потока, и эта проверка проходит успешно (то есть, значение условия не ноль) только если состояние **_good**. В частности, в предыдущем цикле проверялось состояние **istream**, которое возвращает **cin>>z.** Чтобы обнаружить, почему цикл или проверка закончились неудачно, можно исследовать состояние.

Делать проверку на наличие ошибок после каждого ввода или вывода действительно не очень удобно, и обычно источником ошибок служит программист, не сделавший этого в том месте, где это существенно. Например, операции вывода обычно не проверяются, но они могут случайно не сработать. Парадигма потока ввода/вывода построена так, чтобы когда в С++ появится (если это произойдет) механизм обработки исключительных ситуаций (как средство языка или как стандартная библиотека), его будет легко применить для упрощения и стандартизации обработки ошибок в потоках ввода/вывода.

3. Дополнительная литература

1. Бьярн Страуструп. Введение в язык Си++. (глава 8)

2. Герберт Шилдт. Самоучитель С++: Пер. с англ. – 3-е изд. – СПб.: БХВ-Петербург, 2003 (глава 9)

2. Харви Дейтел, Пол Дейтел Как программировать на С++: пер. с англ. -М: ЗАО "Издательство Бином", 1999. (глава 11)

3. Бьярн Страуструп. Язык программирование Си++. – М.: Бином, 2005.(глава 10)

4. Порядок выполнения работы

1. Создайте новый проект в среде MS Visual C++.

2. Наберите текст программы на С++ из **примера 2.** Откомпилируйте и выполните программу. Проанализируйте результаты программы.

3. Использование отладчика. Запуск программу в режиме отладки. Проверка текущего значения переменных, трассировка по шагам, установка контрольных точек, слежение за состоянием переменной.

4. Для полученного варианта задания, руководствуясь описанием настоящей лабораторной работы, напишите программы на языке С++. Откомпилируйте, отладьте и выполните свою программу.

5. Подготовьте отчет о выполнении лабораторной работы

Для успешной сдачи лабораторной работы необходимо:

- представить преподавателю набранный текст программы из примера 2;
- продемострировать уверенную работу с отладчиком **в** среде MS Visual C++;
- представить преподавателю работающую программу для указанного варианта задания;
- ответить на вопросы по тексту программы.

5. Порядок оформления отчета

Отчет о выполнении лабораторной работы должен содержать:

1) титульный лист;
2) вариант задания;
3) текст программы;
4) результаты работы программы.

6. Варианты заданий

Для допуска к экзамену достаточно выполнить любое одно задание. Для студентов, претендующих на освобождение от экзаменов необходимо выполнить оба задания.

Вариант 1

1. Написать программу вычисления ближайшего сверху числа степени 2; Программа должна использовать цикл while. Входные данные поступают с клавиатуры. Результат выводится на экран в десятичной и восьмеричной системе. Установить ширину поля 10 символов, установить точность 9 цифры, заполнить вместо пробелов символом *: с помощью функций и манипуляторов.
2. Проверить какие флаги потока вывода установлены и заменить попарно на левое выравнивание на правое выравнивание (и наоборот) убрать вывод основания системы, если установлен и установить, если сброшен Проверить результат. Добавить в программу два своих манипулятора. Один выводит сообщение, другой устанавливает шестнадцатеричный вывод и ширину поля 10.

Вариант 2

1. Написать программу преобразования шкалы Цельсия в шкалу Фаренгейта. 0 по Цельсию равен 32 по Фаренгейту. 1 градус по Цельсия равен 1.8 по Фаренгейту. Входные данные поступают с клавиатуры. Предусмотреть обработку ошибок. Установить ширину поля 10 символов, установить точность 9 цифры, заполнить вместо пробелов символом +: с помощью функций и манипуляторов. Результат выводится на экран в десятичной и шестнадцатеричной системе.
2. Проверить какие флаги потока вывода установлены и заменить попарно на десятичные на шестнадцатеричные (и наоборот) левое выравнивание на правое выравнивание (и наоборот) Проверить результат. Добавить в программу два своих манипулятора. Один выводит сообщение, другой устанавливает восьмеричный вывод и точность 15.

Вариант 3

1. Написать программу преобразования шкалы Фаренгейта в шкалу Цельсия. 0 по Цельсию равен 32 по Фаренгейту. 1 градус по Цельсия равен 1.8 по Фаренгейту. Установить ширину поля 11 символов, установить точность 8 цифры, заполнить вместо пробелов символом # с помощью функций и манипуляторов. Результат выводится на экран в десятичной и восьмеричной системе.
2. Проверить какие флаги потока вывода установлены и заменить попарно на десятичные на шестнадцатеричные (и наоборот) левое выравнивание на правое выравнивание (и наоборот) Проверить результат. Добавить в программу два своих манипулятора. Один выводит сообщение, другой устанавливает шестнадцатеричный вывод и символ заполнения _.

Вариант 4

1. Написать программу решения квадратного уравнения. Корни только вещественные. Ввод и вывод через стандартные потоки ввода-вывода. Вывод результата в "научном" формате. Установить ширину поля 12 символов, установить точность 4 цифры, заполнить вместо пробелов символом _. с помощью

функций и манипуляторов. Результат выводится на экран в десятичной и шестнадцатеричной системе.
2. Проверить какие флаги потока вывода установлены и заменить попарно на десятичные на шестнадцатеричные (и наоборот) обычную на научную нотацию (и наоборот) Проверить результат. Добавить в программу два своих манипулятора. Один выводит сообщение, другой устанавливает шестнадцатеричный вывод и переход на новую строку.

Вариант 5

1. Написать программу, вычисляющую $y(n) = \sum_{k=0}^{n} x(n-k), n = 0,...N-1$ Программа должна использовать цикл while. Входные данные поступают с клавиатуры. Установить точность 4 цифры. Предусмотреть обработку ошибок. установить ширину поля 10 символов, заполнить вместо пробелов символом ^ с помощью функций и манипуляторов. Результат выводится на экран в десятичной, шестнадцатеричной и восьмеричной системе.
2. Проверить какие флаги потока вывода установлены и заменить попарно на левое выравнивание на правое выравнивание (и наоборот) убрать + перед числом, если установлен и установить, если сброшен Проверить результат. Добавить в программу два своих манипулятора. Один выводит сообщение, другой устанавливает десятичный вывод и ширину поля 10.

Вариант 6

1. Написать программу, печатающую символы от a до z. Использовать цикл while. В строку выводится номер, символ, шестнадцатеричный и восьмеричный код. Установить ширину поля 10 символов, установить точность 4 цифры, заполнить вместо пробелов символом $ с помощью функций и манипуляторов.
2. Проверить какие флаги потока вывода установлены и заменить попарно на убрать вывод основания системы, если установлен и установить, если сброшен убрать + перед числом, если установлен и установить, если сброшен Проверить результат. Добавить в программу два своих манипулятора. Один выводит сообщение, другой устанавливает шестнадцатеричный вывод и точность 10.

Вариант 7

1. Написать программу - простой калькулятор с операциями +, -, *, /. Входные данные, включая операции, поступают с клавиатуры. Предусмотреть обработку ошибок. Результат выводится на экран в десятичной, шестнадцатеричной и восьмеричной системе. Установить ширину поля 10 символов, установить точность 4 цифры, заполнить вместо пробелов символом ~ с помощью функций и манипуляторов.
2. Проверить какие флаги потока вывода установлены и заменить попарно на обычную на научную нотацию (и наоборот) убрать вывод основания системы, если установлен и установить, если сброшен Проверить результат. Добавить в программу два своих манипулятора. Один выводит сообщение, другой устанавливает шестнадцатеричный вывод и символ заполнения ?.

Вариант 8

1. Написать программу - возведение числа n в m-ую степень. Входные данные поступают с клавиатуры. Предусмотреть обработку ошибок. Результат выводится на экран в десятичной, шестнадцатеричной и восьмеричной системе. Установить ширину поля 10 символов, установить точность 4 цифры, заполнить вместо пробелов символом & с помощью функций и манипуляторов.
2. Проверить какие флаги потока вывода установлены и заменить попарно на левое выравнивание на правое выравнивание (и наоборот) убрать + перед числом, если установлен и установить, если сброшен Проверить результат. Добавить в программу два своих манипулятора. Один выводит сообщение, другой устанавливает восьмеричный вывод и ширину поля 20.

Вариант 9

1. Написать программу - посчитать длину окружности. Входные данные поступают с клавиатуры. Установить ширину поля 10 символов, установить точность 4 цифры, заполнить вместо пробелов символом / с помощью функций и манипуляторов.Предусмотреть обработку ошибок. Результат выводится на экран в десятичной, шестнадцатеричной и восьмеричной системе.
2. Проверить какие флаги потока вывода установлены и заменить попарно на обычную на научную нотацию (и наоборот) убрать вывод основания системы, если установлен и установить, если сброшен Проверить результат. Добавить в программу два своих манипулятора.Один выводит сообщение, другой устанавливает десятичный вывод и точность 6.

Вариант 10

1. Написать программу - посчитать площадь окружности. Входные данные поступают с клавиатуры. Установить ширину поля 10 символов, установить точность 4 цифры, заполнить вместо пробелов символом - с помощью функций и манипуляторов. Предусмотреть обработку ошибок. Результат выводится на экран в десятичной, шестнадцатеричной и восьмеричной системе.
2. Проверить какие флаги потока вывода установлены и заменить попарно на левое выравнивание на правое выравнивание (и наоборот) убрать + перед числом, если установлен и установить, если сброшен Проверить результат. Добавить в программу два своих манипулятора. Один выводит сообщение, другой устанавливает шестнадцатеричный вывод и ширину поля 10.

Вариант 11

1. Написать программу решения линейного уравнения. Входные данные поступают с клавиатуры. Предусмотреть обработку ошибок. Установить ширину поля 10 символов. Установить точность 4 цифры. Заполнить вместо пробелов символом %. Результат выводится на экран в десятичной, шестнадцатеричной и восьмеричной системе.
2. Проверить какие флаги потока вывода установлены и заменить попарно на десятичные на шестнадцатиричные (и наоборот) левое выравнивание на правое выравнивание (и наоборот) Проверить результат. Добавить в программу два своих манипулятора. Один выводит сообщение, другой устанавливает шестнадцатеричный вывод и символ заполнения $.

Вариант 12

1. Написать программу, которая получает данные по Фаренгейту в виде 59F и преобразует их в данные по Цельсию 15C. 0 по Цельсию равен 32 по Фаренгейту. 1 градус по Цельсия равен 1.8 по Фаренгейту. Входные данные поступают с клавиатуры. . Предусмотреть обработку ошибок.Установить ширину поля 10 символов, установить точность 4 цифры, заполнить вместо пробелов символом * с помощью функций и манипуляторов. Результат выводится на экран в десятичной, шестнадцатеричной и восьмеричной системе
2. Проверить какие флаги потока вывода установлены и заменить попарно на обычную на научную нотацию (и наоборот) убрать вывод основания системы, если установлен и установить, если сброшен Проверить результат. Добавить в программу два своих манипулятора. Один выводит сообщение, другой устанавливает шестнадцатеричный вывод и ширину поля 10.

Вариант 13

1. Написать программу вычисления ближайшего сверху числа степени 2; Программа должна использовать цикл for. Входные данные поступают с клавиатуры. Предусмотреть обработку ошибок. Результат выводится на экран в десятичной и восьмеричной системе. Установить ширину поля 10 символов, установить точность 9 цифры, заполнить вместо пробелов символом *: с помощью функций и манипуляторов.
2. Проверить какие флаги потока вывода установлены и заменить попарно на десятичные на шестнадцатеричные (и наоборот) левое выравнивание на правое выравнивание (и наоборот) Проверить результат. Добавить в программу два своих манипулятора. Один выводит сообщение, другой устанавливает восьмеричный вывод и точность 15.

Вариант 14

1. Написать программу вычисления частного и остатка от деления двух целых чисел. Входные данные поступают с клавиатуры. Предусмотреть обработку ошибок. Установить ширину поля 10 символов, заполнить вместо пробелов символом $ с помощью функций и манипуляторов. Результат выводится на экран в десятичной и восьмеричной системе.
2. Проверить какие флаги потока вывода установлены и заменить попарно на десятичные на шестнадцатеричные (и наоборот) обычную на научную нотацию (и наоборот) Проверить результат. Добавить в программу два своих манипулятора. Один выводит сообщение, другой устанавливает шестнадцатеричный вывод и переход на новую строку.

Вариант 15

1. Написать программу вычисления ближайшего снизу числа степени 2; Программа должна использовать цикл while. Входные данные поступают с клавиатуры. Предусмотреть обработку ошибок. Установить ширину поля 10 символов, заполнить вместо пробелов символом $ с помощью функций и манипуляторов. Результат выводится на экран в десятичной и шестнадцатеричной системе.
2. Проверить какие флаги потока вывода установлены и заменить попарно на десятичные на шестнадцатеричные (и наоборот) левое выравнивание на правое выравнивание (и наоборот) Проверить результат. Добавить в программу два своих манипулятора. Один выводит сообщение, другой устанавливает шестнадцатеричный вывод и символ заполнения _.

Вариант 16

1. Написать программу решения квадратного уравнения. Корни могут быть комплексными. Входные данные поступают с клавиатуры. Предусмотреть обработку ошибок. Установить ширину поля 10 символов, установить точность 4 цифры, заполнить вместо пробелов символом & с помощью функций и манипуляторов. Результат выводится на экран в десятичной и шестнадцатеричной системе.
2. Проверить какие флаги потока вывода установлены и заменить попарно на левое выравнивание на правое выравнивание (и наоборот) убрать + перед числом, если установлен и установить, если сброшен Проверить результат. Добавить в программу два своих манипулятора. Один выводит сообщение, другой устанавливает восьмеричный вывод и ширину поля 20.

Вариант 17

1. Написать программу, которая получает данные по Цельсию в виде 15C и преобразует их в данные по Фаренгейту 59F. 0 по Цельсию равен 32 по Фаренгейту. 1 градус по Цельсия равен 1.8 по Фаренгейту. Установить ширину поля 10 символов, заполнить вместо пробелов символом $ с помощью функций и манипуляторов. Результат выводится на экран в десятичной и шестнадцатеричной системе.
2. Проверить какие флаги потока вывода установлены и заменить попарно на обычную на научную нотацию (и наоборот) убрать вывод основания системы, если установлен и установить, если сброшен Проверить результат. Добавить в программу два своих манипулятора.Один выводит сообщение, другой устанавливает десятичный вывод и точность 6.

Вариант 18

1. Написать программу, которая получает данные либо по Фаренгейту в виде 59F и преобразует их в данные по Цельсию 15C, либо наоборот. 0 по Цельсию равен 32 по Фаренгейту. 1 градус по Цельсия равен 1.8 по Фаренгейту. Входные данные поступают с клавиатуры. Предусмотреть обработку ошибок. Установить ширину поля 10 символов, установить точность 4 цифры, заполнить вместо пробелов символом \ с помощью манипуляторов. Результат выводится на экран в десятичной, шестнадцатеричной и восьмеричной системе.
2. Проверить какие флаги потока вывода установлены и заменить попарно на десятичные на шестнадцатиричные (и наоборот) левое выравнивание на правое выравнивание (и наоборот) Проверить результат. Добавить в программу два своих манипулятора. Один выводит сообщение, другой устанавливает шестнадцатеричный вывод и символ заполнения $.

Вариант 19

1. Написать программу, печатающую все вводимые символы. В строку выводится символ, шестнадцатеричный и восьмеричный код. Установить ширину поля 10 символов, установить точность 4 цифры, заполнить вместо пробелов символом / с помощью функций и манипуляторов.
2. Проверить какие флаги потока вывода установлены и заменить попарно на убрать вывод основания системы, если установлен и установить, если сброшен убрать + перед числом, если установлен и установить, если сброшен Проверить результат.

Добавить в программу два своих манипулятора. Один выводит сообщение, другой устанавливает шестнадцатеричный вывод и точность 10.

Вариант 20

1. Написать программу решения квадратного уравнения. Корни только вещественные. Ввод и вывод через стандартные потоки ввода-вывода. Вывод результата в "научном" формате. Установить ширину поля 12 символов, установить точность 4 цифры, заполнить вместо пробелов символом _. с помощью функций и манипуляторов. Результат выводится на экран в десятичной и шестнадцатеричной системе.
2. Проверить какие флаги потока вывода установлены и заменить попарно на десятичные на шестнадцатеричные (и наоборот) левое выравнивание на правое выравнивание (и наоборот) Проверить результат. Добавить в программу два своих манипулятора. Один выводит сообщение, другой устанавливает шестнадцатеричный вывод и символ заполнения _.

Вариант 21

1. Написать программу - простой калькулятор с операциями +, -, *, /. Входные данные, включая операции, поступают с клавиатуры. Предусмотреть обработку ошибок. Результат выводится на экран в десятичной, шестнадцатеричной и восьмеричной системе. Установить ширину поля 10 символов, установить точность 4 цифры, заполнить вместо пробелов символом ~ с помощью функций и манипуляторов.
2. Проверить какие флаги потока вывода установлены и заменить попарно на обычную на научную нотацию (и наоборот) убрать вывод основания системы, если установлен и установить, если сброшен Проверить результат. Добавить в программу два своих манипулятора. Один выводит сообщение, другой устанавливает шестнадцатеричный вывод и символ заполнения ?.

Вариант 22

1. Написать программу - возведение числа n в m-ую степень. Входные данные поступают с клавиатуры. Предусмотреть обработку ошибок. Результат выводится на экран в десятичной, шестнадцатеричной и восьмеричной системе. Установить ширину поля 10 символов, установить точность 4 цифры, заполнить вместо пробелов символом & с помощью функций и манипуляторов.
2. Проверить какие флаги потока вывода установлены и заменить попарно на десятичные на шестнадцатеричные (и наоборот) левое выравнивание на правое выравнивание (и наоборот) Проверить результат. Добавить в программу два своих манипулятора. Один выводит сообщение, другой устанавливает восьмеричный вывод и точность 15.

Вариант 23

1. Написать программу, печатающую все вводимые символы в нижнем регистре. Программа должна использовать цикл while. Входные данные поступают с клавиатуры. Предусмотреть обработку ошибок. Результат выводится на экран в десятичной, шестнадцатеричной и восьмеричной системе. Установить ширину поля 10 символов, установить точность 4 цифры, заполнить вместо пробелов символом & с помощью функций и манипуляторов.

2. Проверить какие флаги потока вывода установлены и заменить попарно на убрать вывод основания системы, если установлен и установить, если сброшен убрать + перед числом, если установлен и установить, если сброшен Проверить результат. Добавить в программу два своих манипулятора. Один выводит сообщение, другой устанавливает шестнадцатеричный вывод и точность 10.

Вариант 24

1. Написать программу - посчитать длину окружности. Входные данные поступают с клавиатуры. Установить ширину поля 10 символов, установить точность 4 цифры, заполнить вместо пробелов символом / с помощью функций и манипуляторов.Предусмотреть обработку ошибок. Результат выводится на экран в десятичной, шестнадцатеричной и восьмеричной системе.
2. Проверить какие флаги потока вывода установлены и заменить попарно на десятичные на шестнадцатиричные (и наоборот) левое выравнивание на правое выравнивание (и наоборот) Проверить результат. Добавить в программу два своих манипулятора. Один выводит сообщение, другой устанавливает шестнадцатеричный вывод и символ заполнения $.

Вариант 25

1. Написать программу решения линейного уравнения. Входные данные поступают с клавиатуры. Предусмотреть обработку ошибок. Установить ширину поля 10 символов. Установить точность 4 цифры. Заполнить вместо пробелов символом %. Результат выводится на экран в десятичной, шестнадцатеричной и восьмеричной системе.
2. Проверить какие флаги потока вывода установлены и заменить попарно на левое выравнивание на правое выравнивание (и наоборот) убрать + перед числом, если установлен и установить, если сброшен Проверить результат. Добавить в программу два своих манипулятора. Один выводит сообщение, другой устанавливает шестнадцатеричный вывод и ширину поля 10.