Программирование на языке C++

Зариковская Наталья Вячеславовна

Лекция 2

Начальные сведения о вводе - выводе

- Ввод и вывод как в С, так и в С++ не относятся непосредственно языку. Они обеспечиваются стандартными библиотеками. Для С++ такая библиотека называется iostream. Сейчас мы разберем ее на том уровне, который достаточен для начала работы, а в свое время вернемся к ней для более детального изучения и тогда же разберем средства ввода-вывода, которые включены в стандарт языка С и тоже могут быть доступны в программах на С++.
- Ввод, идущий с клавиатуры пользователя, называется стандартным входным потоком или стандартным вводом. Он связывается с предопределенным в iostream.h потоком cin. Вывод, направляемый на экран пользователя, называется стандартным выходным потоком или стандартным выводом. Он связывается с предопределенным в iostream.h потоком cout.

Начальные сведения о вводе - выводе

- Операция вывода << направляет значение в стандартный выходной поток.
- cout << index;
- Для перехода на новую строку существуют два способа. Первый - это использовать определенный в iostream.h манипулятор endl. Манипулятор можно выводить в поток и при этом он меняет параметры вывода. Здесь endl вызовет переход на новую строку
- cout << endl;
- Второй способ - это явно вывести в поток символ новой строки. В C++ он записывается двумя символами : '\n'.
- cout << '\n';
- Одиночные кавычки ограничивают символ. Такой символ может быть внутри строки символов, например оператор
- cout << "Программа на C++\n";
- вызовет переход на новую строку после вывода данного сообщения.
- В одном операторе вывода можно соединять несколько операций. Например :
- cout << "Значение index равно : " << index << endl;
- Вывод осуществляется по порядку, считая слева направо.
-

Начальные сведения о вводе - выводе

- Аналогично операция ввода >> читает значение из стандартного входного потока, например
- cin >> index;
- Такие операции тоже можно соединять в одном операторе. Например,
- если в программе встретится следующий оператор :
- cin >> i1 >> i2;
- то программа будет ждать ввода с клавиатуры двух величин и первую из них поместит в переменную i1, а вторую - в переменную i2. Эти две вводимых величины можно разделять пробелом или табуляцией, а можно каждую из них вводить с новой строки - операция ввода сработает правильно.
-

Начальные сведения о вводе - выводе

- Если программист забудет включить в программу файл iostream.h, то о каждом появлении в программе cin или cout транслятор будет сообщать как об ошибке, поскольку cin и cout описаны в iostream.h.
- Третий предопределенный поток из iostream.h называется cerr и является стандартным потоком сообщений об ошибках. Он тоже связан с экраном пользователя и нужен, чтобы сообщать пользователю об особых ситуациях и ошибках при выполнении программы.

- Например
- cerr << "Ошибка чтения диска" << endl;
- Для облегчения понимания операций << и >> можно представлять их как воронки, через которые информация из программы (например, из переменной
- i1) выводится на экран (cout)
- cout << i1;
- или с клавиатуры (cin) попадает в программу (например в переменную i2)
- cin >> i2;
-

Элементарный Ввод/Вывод информации

- В C++ отсутствуют специальные операторы ввода/вывода. Для организации ввода в языке С используется библиотечные функции. Библиотеки BORLAND C++ в настоящее время содержат: stdio.h -ANSI C(turbo c); stream.h - C++; iosistem.h - C++. Каж¬дая из этих библиотек содержат значительное число функций. Рассмотрим простейшие.
- Язык С поддерживает форматированный и неформатированный консольный ввод / вывод информации.

Вывод информации на экран дисплея

- Функции неформатного вывода в stdio имеют описание
- int_cdecl putchar(const int_c);
- int_cdecl puts(const char*_s);
- Функция putchar() - предназначена для вывода единичного символа в поток stdout. Аргументом функции может быть: символьный литерал; символьная переменная или символьная константа.
- Функция puts() копирует строку символов с нулевым окончанием в стандартный выходной поток stdout, причём добавляет в конец символ перехода на новую строку. При успешном завершении функция возвращает ненулевое значение, а в противном случае EOF.
- Аргументом функции может быть: строковый литерал; строковая переменная или константа. Примеры:
- #define G "хорошо"
- char g[ ]="хорошо"
- puts("данная информация отобразится на экране дисплея");
- puts('f');- отобразится символьный литерал- f
- puts(g);-отобразится символьная переменная- слово "хорошо"
- puts(G);- отобразится символьная константа- слово "хорошо"
-

Простейший неформатный ввод данных

- Функции неформатного ввода в stdio имеют описание
- int_cdecl qetchar(void);
- int_cdecl qets(char*_s);
- где для ввода символа используется qetchar() без аргумента.

- Функция gets() читает строку символов, оканчивающуюся символом перевода строки в переменную *s из стандартного входного потока stdin. Данная символьная строка оканчивается символом перехода на новую строку, которая при записи в *s заменяется на нулевое окончание ‘\0’.
- Функция qets(string) - аналогична scanf(" %s",string), но в отличие от нее передает все символы до '\n’. Пример:
- #include <stdio.h>
- int main(void)
- {char string[80];
- printf("введите строку");
- gets(string);
- printf("Была введена строка: %s\n", string);
- return 0;
- }

Форматированный ввод/ вывод

- Форматированный ввод/ вывод может быть выполнен благодаря использованию 2-х функций: scanf и printf, соответственно. В заголовочном файле stdio.h эти функции описаны как
- int_cdecl printf(const char *__format,...);
- int_cdecl scanf(const char *__format,...);
- Запишем эти функции в следующем виде:
- printf(упр.текст.(форматная)строка[,список аргументов]);
- scanf(упр.текст.(форматная) строка[,список аргументов]);

- Управляющая строка символов задает способ преобразования данных и состоит из произ¬вольной последовательности символов, в которой знак "%xxx" заменяется символами выводи¬мой информации. Символы после %xxx до первого разделителя рассматриваются как специфи¬кация преобразования значения выводимой переменной из списка аргументов. Спецификация преобразования задается в виде последовательности:

- %[флаг][ширина][.точность][f/n/h/l]<тип>
-

Форматированный ввод/ вывод

- **[флаг]**

-	выравнивание влево в пределах выводимого поля. Правая строка дополняется пробелами (по умалчиванию - выравнивается вправо).
+	выводится знак
пробел	печатается пробел, если число положительное и знак ‘-‘ для отрицательного.
#	выводится идентификатор систем исчисления(0-8ми ричная,0x-16ти ричная ничего-10ти ричная, с точкой - float)

Форматированный ввод/ вывод

- **[ширина]- воздействует только на вывод**

n	ширина поля. Если символов меньше чем ширина, то лишние символы заполняются пробелами. Если символов больше, то выводится сколько надо.
0x	тоже что прежде, но для целого числа позиции слева заполняются символами 0
*	следующий аргумент задает ширину

Форматированный ввод/ вывод

- **[.точность]- воздействует только на вывод**

Ничего	по умолчанию
.O	для d,i,o,u,x -по умолчанию. Для «e,E,f» -десятичная точка отсутствует.
n	не более n знаков после точки, для «e,E,f».
*	следующий аргумент из списка аргументов - точность

Форматированный ввод/ вывод

- **--->[F | N | h | l] -модификатор**

F	F - рассматривается как FAR - указатель
n	рассматривается как near - указатель
h	для d,i,o,u,x,X - аргумент является short int.
L	для (d,i,o,u,x,E) - аргумент longint , для (e,E,f,g,G) аргумент для scanf - double

Форматированный ввод/ вывод

- **<тип>**

C	при вводе читается один байт - ссылка на char (один символ)
C	при выводе переменная преобразуется к типу char (1 байт)

Переменная int.

d	десятичное int со знаком
i	десятичное int со знаком
o	восьмеричное int. unsigned
u	десятичное int unsigned
x	шестнадцатеричное int unsigned; при выводе используются символы o-f
X	шестнадцатеричное int unsigned; при выводе используются символы O-F

Переменная float

f	значение со знаком в форме с фиксированной точкой [-] dddd.dddd
e	значение со знаком в форме [-] d.ddddе[+|-]ddd , при вводе они e|E
E	значение со знаком в форме [-] d.ddddE[+|-]ddd , при вводе они e|E
g	значение со знаком, в форме «f» или «e», в зависимости от значения
G	значение со знаком, в форме «F» или «E» , в зависимости от значения

Переменная char

S	ссылка на массив char - при вводе принимает символы без преобразования до тех пор, пока не достигнута специфицируемая точность или не достигнут символ \n или пробела.
S	При выводе в поток передаются символы до «\0»или пока не достигнута специфицируемая точность.

Форматированный ввод/ вывод

Примеры:
printf ("Печатается данный текст");
курсор остается на этой же строке после символа «Т».
printf ("печатается данный текст \n ");//курсор переходит на следующую строку в первую позицию.

- Вывод строк puts(string) - аналогично printf(“%s\n”,string).
- Необходимо отметить, что функция scanf прекращает работу, если:
- - закончилась управляющая последовательность (форматная строка).
- - очередной элемент ввода не удовлетворяет текущей спецификации преобразования или не совпадает с символом образца.
- - достигнут конец файла ввода.
-

Для 1,2 случая функция возвращает число введенных переменных (т.е. в ячейке scanf число введенных переменных). Для 3 случая функция возвращает значение EOF. Следует заметить, что функция scanf по формату S вводит символы вводимого потока до первого разделителя, поэтому часто приходится очищать буфер stdin с помощью функции fflush(stdin).

Форматированный ввод/ вывод (примеры)

int a,b,c,d;
a=20;
b=-252;
c=01777;
d=0xa7c;
printf(‘’a=%d\t b=%d\t c=%d\t
d=%d\t’’,a,b,c,d);
a=20 b=-252 c=1023 d=2684
Вывод переменных типа int как десятичных целых
	long l,m,n;
	e=-15l;
	m=-455;
	n=156765341;
	printf(‘’e=%ld\t m=%ld\t n=%ld\n’’,l,m,n);
	l=-15 m=-455 n=156765341
	Вывод переменных типа long

short a,b,c
a=99;
b=-32767;
c=-9;
printf(‘’a=%d\t b=%d\t c=%d\t’’,a,b,c);
a=99 b=-32767 c=-9
Вывод переменных типа short
	unsigned int ui,uj;
	uj=0xff;
	printf(‘’восьмер.uj=%0\t
	шестнадц.uj=%x\h’’,uj,uj);
	восьмер.uj=377 шестнадц.uj=ff

Простейший ввод /вывод С++

Рассмотренные выше функции ввода вывода относятся к простейшим и реализованы в ранних версиях языка С++.
Однако эти функции используются и в последних версиях языка, поэтому возникла необходимость их приведения.
В последних версиях операции ввода /вывода реализованы на основе использования методов объектно-ориентированного программирования и поэтому будет приведено их краткое их описание, а подробное их рассмотрение будет произведено в последующих главах.
При рассмотрении вопросов связанных с вводом выводом информации пользуются понятием поток. Под потоком понимается абстрактное понятие, относящееся к любому переносу данных от источника к приемнику данных.
Базовые операции ввода /вывода последних версий поддерживаются тремя классами: istream- ввод; ostream- вывода; iostream- ввода /вывода. Использование базовых функций требует подключения заголовочного файла <iostream.h>- где приведены описания перечисленных классов.

Простейший ввод /вывод C++

В качестве начальных сведений необходимо знать, что пользователю доступны четыре объекта: cin - объект класса istream , обеспечивающий ввод; cout- объект класса ostream , обеспечивающий вывод; cerr- объект класса ostream, выполняющий не буферизованный вывод в стандартный поток ошибок -stderr; clog- объект класса ostream, выполняющий буферизованный вывод в стандартный поток ошибок - stderr.

Библиотека iostream.h перегружает два оператора поразрядного сдвига: << - поместить в выходной поток; >> - получить из входного потока.

Операция потокового вывода “<<”- в качестве операнда слева служит объект типа ostream (cout), а правый операнд может иметь любой встроенный тип (char, short, int, long, char* (строка), float, double, long double, void*) или любой переопределённый из встроенных типов тип. Операция “<<” обладает ассоциативностью слева и возвращает ссылку на объект ostream, что позволяет организовать каскадный вывод операндов.

Например:

```
int d=99;
float r=27.7;
cout<<”переменная d=”<<d<<”,r=”<<r;
```

позволит отобразить на экране дисплея переменную d=99,r=27.7

```
cout<<&d;// выводится адрес переменной  d в шестнадцатеричном формате
```

Простейший ввод /вывод C++

Операция потокового ввода “>>”- в качестве операнда слева служит объект типа istream (cin), а правый операнд может иметь любой встроенный тип (char, short, int, long, char* (строка), float, double, long double, void*) или любой переопределённый из встроенных типов тип. Операция “>>” обладает ассоциативностью слева и возвращает ссылку на объект istream, что позволяет организовать каскадный ввод операндов.

Форматирование ввода и вывода определяется различными флагами состояний формата, перечисленными в классе ios. Эти состояния определяются битами числа типа long int следующим образом:

Простейший ввод /вывод C++

```
 //  флаги  формата
public:
 enum    {
skipws    = 0x0001, //  пропуск пробельного символа на вводе
left         = 0x0002, //  вывод с левым выравниванием
right       = 0x0004, //  вывод с правым выравниванием
internal    = 0x0008, //  заполнитель после знака или указателя системы счисления
dec          = 0x0010,  //  десятичное преобразование
oct          = 0x0020,  //  восьмеричное преобразование
hex          = 0x0040,  //  шестнадцатеричное преобразование
showbase = 0x0080, //  показать на выходе указатель системы счисления
showpoint = 0x0100, //  показать позицию десятичной точки (на выходе)
uppercase  = 0x0200,//  шестнадцатеричный вывод значений буквами верхнего регистра
showpos   = 0x0400, //  добавляет " + " к положительным целым числам
scientific  = 0x0800,// использовать запись чисел с плавающей
точкой с выводом экспоненты E например,  12345E2
fixed = 0x1000, //  использовать запись чисел с плавающей точкой типа 123.45
unitbuf      = 0x2000, // стирание всех потоков после вставки
stdio         = 0x4000, // стирание stdout и stderr после вставки
boolalpha = 0x8000// вставка/извлечение bools как текстовый или числовой
```

Простейший ввод /вывод C++

Эти флаги читаются и устанавливаются функциями элементами flags, setf и unsetf (их описание приводится ниже).

По умолчанию операция “>>” опускает пробельные символы, и затем считывает символы, соответствующие типу вводимого объекта. Пропуск пробельных символов управляется флагом ios::skipws в перечислимой переменной состояния. Флаг skipws обычно ус¬танавливает пропуск пробельных символов. Очистка этого флага (например, при помощи setf) выключает пропуск пробельных символов.

Например:

```
cout<<”введите переменные d,r”;
cin>>d>>r;
```

В результате выполнения операции вывода на экране дисплея будет отображено введите переменные d,r.

После чего необходимо ввести переменные d и r, разделяя их произвольным количеством пробелов.

Использование манипуляторов при вводе /выводе данных

Форматированный вывод в C++ обеспечивается путём изменения переменных форматирования при использовании специальных операторов (функций), называемых манипуляторами.

Манипуляторы потока	
Манипулятор	Действие
dec	Установка флага форматирования с десятичными преобразованиями
hex	Установка флага форматирования с шестнадцатеричными преобразованиями
oct	Установка флага форматирования с восьмеричными преобразованиями
ws	Извлечение пробельных символов
endl	Вставка символа новой строки и очистка потока
ends	Вставка пустого конечного символа в строку
flush	Сброс на диск и очистка ostream
setbase(int n)	Установка системы счисления при преобразованиях с основанием n (0, 8, 10 или 16). Нуль означает по умолчанию десятичную систему при выводе и правила Си для литералов целых чисел при вводе.
Resetiosflags (long f)	Очистка форматных бит в ins или outs, заданных аргументом f.
Setiosflags (long f)	Установка бит форматирования в ins или outs, заданных аргументом f
setfill(int c)	Установка символа-заполнителя в c.
Setprecision (int n)	Установка точности представления чисел с плавающей точкой равной n разрядам
setw(int n)	Установка ширины поля в значение n

Использование манипуляторов при вводе /выводе данных

Манипуляторы воспринимают в качестве аргументов ссылку на поток и возвращают ссылку на тот же поток. Поэтому манипуляторы могут объединяться в цепочку занесения в поток (или извлечений из потока). Например:

```
#include<iostream.h>
//требуется для параметризованных манипуляторов
#include<iomanip.h>
int main(void) {
int i = 1122, j = 3344, k = 55;
cout << setw(4) << i << setw(6) << j<<setw(2)<< k;
cout << "\n";
cout << setw(6) << i << setw(6) << j << setw(6) << k;
cin>>i; return(0);
     }
```

на экране дисплея будет отображена информация:

```
1122  334455
 1122  3344    55
```

Использование манипуляторов при вводе /выводе данных

Манипулятор setw представляет собой параметризованный манипулятор, объявление которого находится в iomanip.h. Прочие параметризованные манипуляторы, работают аналогичным образом.
Поскольку манипуляторы объединяются в цепочку занесения в поток то для некоторых задач целесообразно писать свои собственные манипуляторы, без параметров. Например, ниже приведена функция префиксирования к выводимой информации знака доллара:

```
#include <iostream.h>
// табулировать и префиксировать вывод знаком доллара
ostream& money (ostream& output) {
return output << "\t$";
    }
int main(void) {
int rashod= 135, prixod = 231;
cout<<"rashod="<<money <<rashod<<"prixod ="<<money<< prixod;  return(0);
    }
```

на экране дисплея будет отображена информация:
rashod= $135 prixod = $231

Использование манипуляторов при вводе /выводе данных

Манипуляторы dec, hex и oct изменяют основание системы счисления при преобразовании и оставляют это изменение в силе:

```
int main(void) {
int rashod= 135, prixod = 231;
cout<<"rashod="<<money<<oct<<rashod<<"prixod="<<hex <<money<<prixod;
cin>>rashod;
cout << dec;//желательно , чтобы использовать основание 10
return(0);     }
```

на экране дисплея будет отображена информация:

```
rashod= $207 prixod =   $e7
```