ФАЙЛОВЫЕ ОПЕРАЦИИ В/В В C++

По мере усложнения ваших программ они будут сохранять и получать информацию, используя файлы. Если вы знакомы с файловыми манипуляциями в языке С, вы сможете использовать подобные методы и в C++. Кроме того, как вы узнаете из этого урока, C++ предоставляет набор классов файловых потоков, с помощью которых можно очень легко выполнять операции ввода и вывода (В/В) с файлами. К концу данного урока вы освоите следующие основные концепции:

- Используя выходной файловый поток, вы можете писать информацию в файл с помощью оператора вставки (<<).
- Используя входной файловый поток, вы можете читать хранимую в файле информацию с помощью оператора извлечения (>>).
- Для открытия и закрытия файла вы используете методы файловых классов.
- Для чтения и записи файловых данных вы можете использовать операторы вставки и извлечения, а также некоторые методы файловых классов.

Многие программы, которые вы создадите в будущем, будут интенсивно использовать файлы. Выберите время для экспериментов с программами, представленными в данном уроке. И вы обнаружите, что в C++ выполнять файловые операции очень просто.

ВЫВОД В ФАЙЛОВЫЙ ПОТОК

Из урока 33 вы узнали, что cout представляет собой объект типа ostream(выходной поток). Используя класс ostream, ваши программы могут выполнять вывод в cout с использованием оператора вставки или различных методов класса, например cout.put. Заголовочный файлiostream.h определяет выходной поток cout. Аналогично, заголовочный файл f stream.h определяет класс выходного файлового потока с именемofstream. Используя объекты класса ofstream, ваши программы могут выполнять вывод в файл. Для начала вы должны объявить объект типаofstream, указав имя требуемого выходного файла как символьную строку, что показано ниже:

```
ofstream file_object(«FILENAME.EXT»);
```

Если вы указываете имя файла при объявлении объекта типа ofstream, C++ создаст новый файл на вашем диске, используя указанное имя, или перезапишет файл с таким же именем, если он уже существует на вашем диске Следующая программа OUT_FILE.CPP создает объект типа ofstreamи затем использует оператор вставки для вывода нескольких строк текста в файл BOOKINFO.DAT:

```
#include <fstream.h>

void main(void)

{
ofstream book_file(«BOOKINFO.DAT»);
book_file << «Учимся программировать на языке C++, » << «Вторая редакция» << endl;
book_file << «Jamsa Press» << endl;
book_file << «22.95” << endl;
}
```

В данном случае программа открывает файл BOOKINFO.DAT и затем записывает три строки в файл, используя оператор вставки. Откомпилируйте и запустите эту программу. Если вы

работаете в среде MS-DOS, можете использовать команду TYPE для вывода содержимого этого файла на экран:

```
C:\> TYPE BOOKINFO.DAT <ENTER>

Учимся программировать на языке C++, Вторая редакция

Jamsa Press

$22.95
```

Как видите, в C++ достаточно просто выполнить операцию вывода в файл.

ЧТЕНИЕ ИЗ ВХОДНОГО ФАЙЛОВОГО ПОТОКА

Только что вы узнали, что, используя класс ofstream, ваши программы могут быстро выполнить операции вывода в файл. Подобным образом ваши программы могут выполнить операции ввода из файла, используя объекты типа ifstream. Опять же, вы просто создаете объект, передавая ему в качестве параметра требуемое имя файла:

```
ifstream input_file(«filename.EXT»);
```

Следующая программа FILE_IN.CPP открывает файл BOOKINFO.DAT, который вы создали с помощью предыдущей программы, и читает, а затем отображает первые три элемента файла:

```
#include <iostream.h>

#include <fstream.h>

void main(void)

{
ifstream input_file(«BOOKINFO.DAT») ;
char one[64], two[64], three[64];
input_file >> one;
input_file >> two;
input_file >> three;
cout << one << endl;
cout << two << endl;
cout << three << endl;
}
```

Если вы откомпилируете и запустите эту программу, то, вероятно, предположите, что она отобразит первые три строки файла. Однако, подобно cin, входные файловые потоки используют пустые символы, чтобы определить, где заканчивается одно значение и начинается другое. В результате при запуске предыдущей программы на дисплее появится следующий вывод:

```
C:\> FILE_IN <ENTER>

учимся

программировать

на
```

Чтение целой строки файлового ввода

Из урока 33 вы узнали, что ваши программы могут использовать cin.getlineдля чтения целой строки с клавиатуры. Подобным образом объекты типаifstream могут использовать getline для чтения строки файлового ввода. Следующая программа FILELINE.CPP использует функцию getline для чтения всех трех строк файла BOOKINFO.DAT:

```
#include <iostream.h>

#include <fstream.h>

void main(void)

{
ifstream input_file(«BOOKINFO.DAT»);
char one[64], two[64], three [64] ;
input_file.getline(one, sizeof(one)) ;
input_file.get line(two, sizeof(two));
input_file.getline(three, sizeof(three)) ;
cout << one << endl;
cout << two << endl;
cout << three << endl;
}
```

В данном случае программа успешно читает содержимое файла, потому что она знает, что файл содержит три строки. Однако во многих случаях ваша программа не будет знать, сколько строк содержится в файле. В таких случаях ваши программы будут просто продолжать чтение содержимого файла пока не встретят конец файла.

ОПРЕДЕЛЕНИЕ КОНЦА ФАЙЛА

Обычной файловой операцией в ваших программах является чтение содержимого файла, пока не встретится конец файла. Чтобы определить конец файла, ваши программы могут использовать функцию eof потокового объекта. Эта функция возвращает значение 0, если конец файла еще не встретился, и 1, если встретился конец файла. Используя цикл while, ваши программы могут непрерывно читать содержимое файла, пока не найдут конец файла, как показано ниже:

```
while (! input_file.eof())

{
// Операторы
}
```

В данном случае программа будет продолжать выполнять цикл, пока функция eof возвращает ложь (0). Следующая программа TEST_EOF.CPP использует функцию eof для чтения содержимого файла BOOKINFO.DAT, пока не достигнет конца файла:

```
#include <iostream.h>

#include <fstream.h>

void main (void)

{
ifstream input_file(«BOOKINFO.DAT»);
```

```
char line[64];
while (! input_file.eof())

{
input_file.getline(line, sizeof(line));
cout << line << endl;
}
}
```

Аналогично, следующая программа WORD_EOF.CPP читает содержимое файла по одному слову за один раз, пока не встретится конец файла:

```
#include <iostream.h>

#include <fstream.h>

void main(void)

{
ifstream input_file(«BOOKINFO.DAT»);
char word[64] ;
while (! input_file.eof())

{
input_file >> word;
cout << word << endl;
}
}
```

И наконец, следующая программа CHAR_EOF.CPP читает содержимое файла по одному символу за один раз, используя функцию get, пока не встретит конец файла:

```
#include <iostream.h>

#include <fstream.h>

void main(void)

{
ifstream input_file(«BOOKINFO.DAT»);
char letter;
while (! input_file.eof())

{
letter = input_file.get();
cout << letter;
}
}
```

ПРОВЕРКА ОШИБОК ПРИ ВЫПОЛНЕНИИ ФАЙЛОВЫХ ОПЕРАЦИЙ

Программы, представленные до настоящего момента, предполагали, что во время файловых операций В/В не происходят ошибки. К сожалению, это сбывается не всегда. Например, если вы открываете файл для ввода, ваши программы должны проверить, что файл существует.

Аналогично, если ваша программа пишет данные в файл, вам необходимо убедиться, что операция прошла успешно (к примеру, отсутствие места на диске, скорее всего, помешает записи данных). Чтобы помочь вашим программам следить за ошибками, вы можете использовать функцию fail файлового объекта. Если в процессе файловой операции ошибок не было, функция возвратит ложь (0). Однако, если встретилась ошибка, функция fail возвратит истину. Например, если программа открывает файл, ей следует использовать функцию fail, чтобы определить, произошла ли ошибка, как это показано ниже:

```
ifstream input_file(«FILENAME.DAT»);
if (input_file.fail())

{
cerr << «Ошибка открытия FILENAME.EXT» << endl;
exit(1);
}
```

Таким образом, программы должны убедиться, что операции чтения и записи прошли успешно. Следующая программа TEST_ALL.CPP использует функцию fail для проверки различных ошибочных ситуаций:

```
#include <iostream.h>

#include <fstream.h>

void main(void)

{
char line[256] ;
ifstream input_file(«BOOKINFO.DAT») ;
if (input_file.fail()) cerr << «Ошибка открытия BOOKINFO.DAT» << endl;
else

{
while ((! input_file.eof()) && (! input_file.fail()))

{
input_file.getline(line, sizeof(line)) ;
if (! input_file.fail()) cout << line << endl;
}
}
}
```

ЗАКРЫТИЕ ФАЙЛА, ЕСЛИ ОН БОЛЬШЕ НЕ НУЖЕН

При завершении вашей программы операционная система закроет открытые ею файлы. Однако, как правило, если вашей программе файл больше не нужен, она должна его закрыть. Для закрытия файла ваша программа должна использовать функцию close, как показано ниже:

```
input_file.close ();
```

Когда вы закрываете файл, все данные, которые ваша программа писала в этот файл, сбрасываются на диск, и обновляется запись каталога для этого файла.

УПРАВЛЕНИЕ ОТКРЫТИЕМ ФАЙЛА

В примерах программ, представленных в данном уроке, файловые операции ввода и вывода выполнялись с начала файла. Однако, когда вы записываете данные в выходной файл, вероятно, вы захотите, чтобы программа добавляла информацию в конец существующего файла. Для открытия файла в режиме добавления вы должны при его открытии указать второй параметр, как показано ниже:

```
ifstream output_file(«FILENAME.EXT», ios::app);
```

В данном случае параметр ios::app указывает режим открытия файла. По мере усложнения ваших программ они будут использовать сочетание значений для режима открытия файла, которые перечислены в табл. 34.

Таблица 34. Значения режимов открытия.

Режим открытия	Назначение
ios::app	Открывает файл в режиме добавления, располагая файловый указатель в конце файла.
ios::ate	Располагает файловый указатель в конце файла.
ios::in	Указывает открыть файл для ввода .
ios::nocreate	Если указанный файл не существует, не создавать файл и возвратить ошибку.
ios::noreplace	Если файл существует, операция открытия должна быть прервана и должна возвратить ошибку.
ios::out	Указывает открыть файл для вывода.
ios::trunc	Сбрасывает (перезаписывает) содержим, з существующего файла.

Следующая операция открытия файла открывает файл для вывода, используя режим ios::noreplace, чтобы предотвратить перезапись существующего файла:

```
ifstream output_file(«FIlename.EXT», ios::out | ios::noreplace);
```

ВЫПОЛНЕНИЕ ОПЕРАЦИЙ ЧТЕНИЯ И ЗАПИСИ

Все программы, представленные в данном уроке, выполняли файловые операции над символьными строками. По мере усложнения ваших программ, возможно, вам понадобится читать и писать массивы и структуры. Для этого ваши программы могут использовать функции readи write. При использовании функций read и write вы должны указать буфер данных, в который данные будут читаться или из которого они будут записываться, а также длину буфера в байтах, как показано ниже:

```
input_file.read(buffer, sizeof(buffer)) ;
output_file.write(buffer, sizeof(buffer));
```

Например, следующая программа STRU_OUT.CPP использует функциюwrite для вывода содержимого структуры в файл EMPLOYEE.DAT:

```
#include <iostream.h>

#include <fstream.h>
```

```
void main(void)

{
struct employee

{
char name[64];
int age;
float salary;
} worker = { «Джон Дой», 33, 25000.0 };

ofstream emp_file(«EMPLOYEE.DAT») ;
emp_file.write((char *) &worker, sizeof(employee));
}
```

Функция write обычно получает указатель на символьную строку. Символы (char *) представляют собой оператор приведения типов, который информирует компилятор, что вы передаете указатель на другой тип. Подобным образом следующая программа STRU_IN.CPP использует метод read для чтения из файла информации о служащем:

```
#include <iostream.h>

#include <fstream.h>

void main(void)

{
struct employee

{
char name [6 4] ;
int age;
float salary;
} worker = { «Джон Дой», 33, 25000.0 };

ifstream emp_file(«EMPLOYEE.DAT»);
emp_file.read((char *) &worker, sizeof(employee));
cout << worker.name << endl;
cout << worker.age << endl;
cout << worker.salary << endl;
}
```

ЧТО ВАМ НЕОБХОДИМО ЗНАТЬ

По мере усложнения ваших программ вы регулярно будете использовать файловые операции. Выберите время для экспериментов с программами, представленными в этом уроке. Из урока 35 вы узнаете, как улучшить производительность ваших программ с использованием встроенных функций. Прежде чем перейти к уроку 35, убедитесь, что вы изучили следующее:

1. Заголовочный файл fstream.h определяет классы ifstream иofstream, с помощью которых ваша программа может выполнять операции файлового ввода и вывода.
2. Для открытия файла на ввод или вывод вы должны объявить объект типа ifstream или ofstream, передавая конструктору этого объекта имя требуемого файла.
3. После того как ваша программа открыла файл для ввода или вывода, она может читать или писать данные, используя операторы извлечения (>>) и вставки (<<).

4. Ваши программы могут выполнять ввод или вывод символов в файл или из файла, используя функции get и put.
5. Ваши программы могут читать из файла целую строку, используя функцию getline.
6. Большинство программ читают содержимое файла, пока не встретится конец файла. Ваши программы могут определить конец файла с помощью функции eof.
7. Когда ваши программы выполняют файловые операции, они должны проверять состояние всех операций, чтобы убедиться, что операции выполнены успешно. Для проверки ошибок ваши программы могут использовать функцию fail.
8. Если вашим программам необходимо вводить или выводить такие данные, как структуры или массивы, они могут использовать методы read и write.
9. Если ваша программа завершила работу с файлом, его следует закрыть с помощью функции close.