**Programs:**

1. **Compare cin, cin.get() and cin.getline()**

#include<iostream>

#include<conio.h>

using namespace std;

int main()

{

const int s = 100;

char str1[s], str2[s], str3[s];

cout << "Enter a sentence: " << endl;

cin >> str1;

cin.ignore(std::numeric\_limits<std::streamsize>::max(), '\n');

cout << "\nThe string read with cin was:\n" << str1 << endl;

cout << "\nEnter a sentence: " << endl;

cin.get(str2, 10);

cin.ignore(std::numeric\_limits<std::streamsize>::max(), '\n');

cout << "\nThe string read with cin.get was:\n" << str2 << endl;

cout << "\nEnter a sentence: " << endl;

cin.getline(str3, 20);

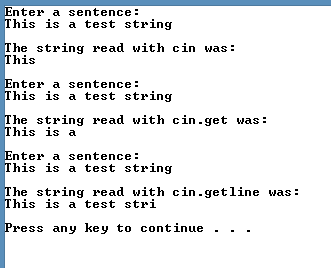
cin.ignore(std::numeric\_limits<std::streamsize>::max(), '\n');

cout << "\nThe string read with cin.getline was:\n" << str3 << endl << endl;

return 0;

}

**Output:**



## Demostrate use of peek, putback and ignore

#include<iostream>

#include<conio.h>

using namespace std;

int main()

{

char ch;

cout << "Enter a sentence: " << endl;

while (cin.get(ch))

{

if (ch == '!')

cin.putback('$');

else

cout << ch;

while (cin.peek() == '#')

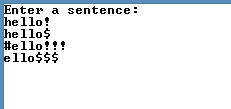
cin.ignore(1, '#');

}

\_getch();

}

**Output:**



## Demostrate use of read, write and gcount.

## #include<iostream>

## #include<conio.h>

## using namespace std;

## int main()

## {

## 

## const int s = 100;

## char str[s];

## cout << "Enter sentence:" << endl;

## cin.read(str, 20);

## cout << "The string read was:" << endl;

## cout.write(str,cin.gcount());

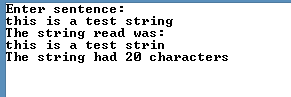
## cout << "\nThe string had " << cin.gcount() << " characters";

## \_getch();

## return 0;

## }

**Output:**



## Print an integer value in octal, hexadecimal and decimal and setbase.

## #include<iostream>

## #include<conio.h>

## #include<iomanip>

## using namespace std;

## int main()

## {

## int n;

## cout << "Enter a decimal number" << endl;

## cin >> n;

## cout << n << " in HexaDecimal is: " << hex << n << endl;

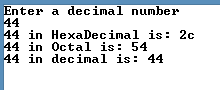
## cout << dec << n << " in Octal is: " << oct << n << endl;

## cout << setbase(10) << n << " in decimal is: " << n << endl;

## \_getch();

## }

**Output:**



## Demostrate use of precision and setprecision

## #include<iostream>

## #include<conio.h>

## #include<iomanip>

## #include<cmath>

## using namespace std;

## int main()

## {

## double r2 = sqrt(2.0);

## 

## cout << "Square root of 2 with precision 0-9.\n";

## cout << "Precision set by ios\_base member function precision: \n";

## cout << fixed;

## 

## for (int i = 0;i <= 9;i++)

## {

## cout.precision(i);

## cout << r2 << endl;

## }

## cout << "\nPrecision set by io manipulator member function setprecision: \n";

## for (int i = 0;i <= 9;i++)

## cout << setprecision(i) << r2 << endl;

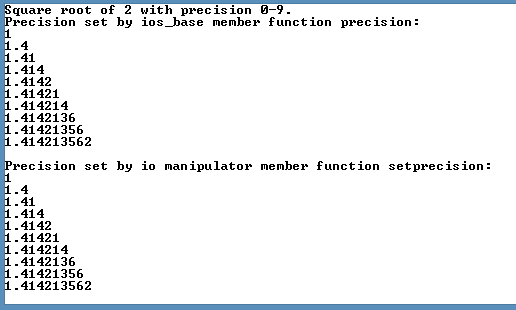
## 

## \_getch();

## return 0;

## }

**Output:**



## Demostrate use of

## showpoint,

## left, right and internal justification

## scientific and fixed notation

## boolalpha

#include <iostream>

#include<iomanip>

#include<conio.h>

using namespace std;

int main()

{

cout << "Implemening showpoint:\n";

cout << "Before using showpoint" << endl

<< "9.9900 prints as: " << 9.9900 << endl

<< "9.9000 prints as: " << 9.9000 << endl

<< "9.0000 prints as: " << 9.0000 << endl;

cout << showpoint

<< "After using showpoint" << endl

<< "9.9900 prints as: " << 9.9900 << endl

<< "9.9000 prints as: " << 9.9000 << endl

<< "9.0000 prints as: " << 9.0000 << endl << endl;

cout << "Implementing left, right and internal justification:\n";

int a = 12345;

cout << "Default is right justified:" << endl << setw(10) << a;

cout << "\nUse std::left to left justify x:\n" << left << setw(10) << a;

cout << "\nUse std::right to right justify x:\n" << right << setw(10) << a << endl << endl;

cout << "Implementing scientific and fixed notation:\n";

double x = 0.001234567;

double y = 1.946e9;

cout << "Displayed in default format:" << endl << x << '\t' << y << endl;

cout << "Displayed in scientific format:" << endl << scientific << x << '\t' << y << endl;

cout << "Displayed in fixed format:" << endl << fixed << x << '\t' << y << endl << endl;

cout << "Implementing boolalpha:\n";

bool booleanValue = true;

cout << "BooleanValue is " << booleanValue << endl;

cout << "BooleanValue (after using boolalpha) is " << boolalpha << booleanValue << endl;

cout << "Switch booleanValue and use noboolalpha" << endl;

booleanValue = false;

cout << noboolalpha;

cout << "BooleanValue is " << booleanValue << endl;

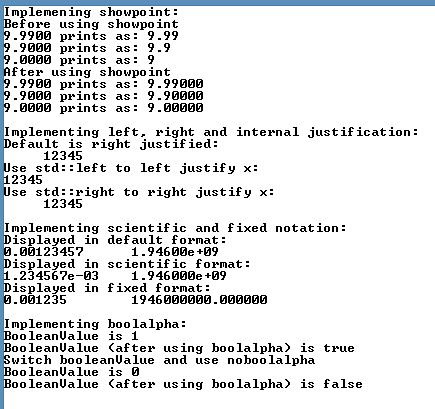
cout << "BooleanValue (after using boolalpha) is " << boolalpha << booleanValue << endl;

\_getch();

return 0;

}

**Output:**



## program to create user defined output stream manipulators.

## #include<iostream>

## #include<conio.h>

## using namespace std;

## ostream& bell(ostream& output){

## return output << "\a";

## }

## ostream& carraigeReturn(ostream& output){

## return output << "\r";

## }

## ostream& tab(ostream& output){

## return output << "\t";

## }

## ostream& endLine(ostream& output){

## return output << "\n" << flush;

## }

## int main(){

## cout << "Use Of tab and endline manipulator" << endLine;

## cout << "a" << tab << "b" << tab << "c" << endLine;

## cout << "Use of carraigeReturn and bell maniplulator" << endLine;

## cout << bell;

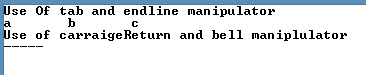
## cout << carraigeReturn << "-----" << endLine;

## \_getch();

## return 0;

## }

**Output:**



## Show stream error states with examples

## using namespace std;

## int main()

## {

## int integerValue;

## cout << " Before a bad input operation:"

## << "\n cin.rdstate(): " << cin.rdstate()

## << "\n cin.eof(): " << cin.eof()

## << "\n cin.fail(): " << cin.fail()

## << "\n cin.bad(): " << cin.bad()

## << "\n cin.good(): " << cin.good()

## << "\n\n Expects an integer, but enter a character: ";

## cin >> integerValue;

## cout << endl;

## cout << " After a bad input operation:"

## << "\n cin.rdstate(): " << cin.rdstate()

## << "\n cin.eof(): " << cin.eof()

## << "\n cin.fail(): " << cin.fail()

## << "\n cin.bad(): " << cin.bad()

## << "\n cin.good(): " << cin.good() << endl << endl;

## cin.clear();

## cout << " After cin.clear()" << "\n cin.fail(): " << cin.fail() << "\n cin.good(): " << cin.good() << endl;

## \_getch();

## }

## Output:

