```
Script started on Tue 21 Feb 2012 11:13:50 PM CST
\033]0;georgia@georgia-MT6017: ~/cplusplus\007georgia@georgia-MT6017:~/cplusplus$ p
wd
/home/georgia/cplusplus
\033]0;georgia@georgia-MT6017: ~/cplusplus\007georgia@georgia-MT6017:~/cplusplus$ D
\033[KCPP --version
This is CPP version 1.219 executing under perl v5.12.4 and compiling with:

g++ (Ubuntu/Linaro 4.6.1-9ubuntu3) 4.6.1
Copyright (C) 2011 Free Software Foundation, Inc.
This is free software; see the source for copying conditions.  There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.


\033]0;georgia@georgia-MT6017: ~/cplusplus\007georgia@georgia-MT6017:~/cplusplus$ c
at pointinpointout.in\033[Kfo
Name: Jakob Hansen
Class: CSC121 - Evening Section
Project: Point(in), out=Point()

Levels attempted:

4 - For just getting the basic instructions down.

+1 - For repeatability

+2 - For the menu structure

attempted another +2 for forceful input validation...although definitely failed.

+? - For putting the calculations in functions.
Not sure how it applies point-wise. Not sure if they are syntactically correct.

Program Description:

A program that calculates the midpoint and length of a line segment defined
by two coordinates on a cartesian plane. The user may enter the points in
any way they want, but (well, this is what I planned anyway) depending on
how strict the error handling is chosen to be, you will either be allowed to
continue with input (with handy reminder messages!) or not allowed to continue
until you get the notation correct. A 3-item menu structure allows the user to
1. calculate some points, 2. set some error handling strictness, or 3. quit.









\033]0;georgia@georgia-MT6017: ~/cplusplus\007georgia@georgia-MT6017:~/cplusplus$ c
at pointinpointout.cpp
#include <iostream>
#include <cctype>
#include <climits>
#include <cmath>
#include <string>

using namespace std;

double midpointCalc(double point_a, double point_b);


double distanceCalc(double xa, double xb, double ya, double yb);


int main(void)
{
    const char holder = 'x';
          double xa;
          double ya;
          double xb;
          double yb;
          char t;
          char choice;
          string wlevel = "WARNING";
          bool warning = true;
          bool errmes1 = false;
          bool errmes2 = false;
          bool errmes3 = false;
          double midx;
          double midy;
          double distance;



         cout << "\n2d Point Calculation Menu:\n\n"
         << "1. Perform Calculations!\n"
         << "2. Set Notation Complaint Level [" << wlevel
         << "]\n3. Quit\n\n";

    cin >> choice;

    cin.ignore(INT_MAX, '\n');

    while (toupper(choice) != 'Q' && choice != '3')
    {
        if (toupper(choice) == 'S' || toupper(choice) == 'N' ||
           toupper(choice) == 'L' || choice == '2')
        {
            cout << "Type \"E\" to set notation complaint level to \"ERROR\"."
             << "\nThis means you will not be able to continue input unless"
             << " done correctly\n\n"
             << "Type \"W\" to set notation complaint level to \"WARNING\"."
             << "\nThis means you will get warning messages,\nbut will"
             << "still be able to enter coordinates even in improper form.\n\n";

             cin >> t;
             cin.ignore(INT_MAX, '\n');

             if (toupper(t) == 'E')
             {
                 wlevel = "ERROR";
                 warning = false;
             }
             else
             {
                 wlevel = "WARNING";
                 warning = true;
             }

             cout << "\n2d Point Calculation Menu:\n\n"
                  << "1. Perform Calculations!\n"
                  << "2. Set Notation Complaint Level [" << wlevel
                  << "]\n3. Quit\n\n";

             cin >> choice;

             cin.ignore(INT_MAX, '\n');

        }

        else if ( (toupper(choice) == 'P' || choice == '1') && warning == true )
        {
            errmes1 = false;
            errmes2 = false;
            errmes3 = false;
```

```
cout << "\n\tExcellent. \n\nPlease enter the first point: ";

cin >> ws; // get whitespace

t = cin.peek();        //find out what's after whitespace

if (!isdigit(t) &&   // if its not a number or a + or -
   t != '-' &&
   t != '+')
{
    cin >> t;            // then we'll take it
}
else
{
    t = holder; // if a number has been entered, give t a value
    errmes1 = true; // turn on condition for error message later
}

cin >> xa;       // then actually read the number

cin >> ws;    // repeat

t = cin.peek();

if (!isdigit(t) && t != '-' && t != '+')
{
    cin >> t;
}
else
{
    t = holder;
    errmes2 = true;
}

cin >> ya;

while (cin.peek() != '\n' && isspace(cin.peek()))
{
    cin.ignore();
}
if (cin.peek() != '\n')
{
    cin >> t;
}
else
{
    t = holder;
    errmes3 = true;
}

if (errmes1 == true)
{
    cerr << "\nDon't forget your opening parenthesis!\n";
}

if (errmes2 == true)
{
    cerr << "\nDon't forget the comma between x and y values!\n";
}

if (errmes3 == true)
{
    cerr << "\nDon't forget your closing parenthesis!\n";
}

errmes1 = false;  // reset error message switches
errmes2 = false;
errmes3 = false;
```

```
cout << "\nPlease Enter the second Point:  ";

cin >> ws;

t = cin.peek();

if (!isdigit(t) && t != '-' && t != '+')
{
    cin >> t;
}
else
{
    t = holder;
   errmes1 = true;
}

cin >> xb;

cin >> ws;

t = cin.peek();

if (!isdigit(t) && t != '-' && t != '+')
{
    cin >> t;
}
else
{
    t = holder;
    errmes2 = true;
}

cin >> yb;

while (cin.peek() != '\n' && isspace(cin.peek()))
{
    cin.ignore();
}
if (cin.peek() != '\n')
{
    cin >> t;
}
else
{
    t = holder;
    errmes3 = true;
}

if (errmes1 == true)
{
    cerr << "\nDon't forget your opening parenthesis!\n";
}

if (errmes2 == true)
{
    cerr << "\nDon't forget the comma between x and y values!\n";
}

if (errmes3 == true)
{
    cerr << "\nDon't forget your closing parenthesis!\n";
}


midx = midpointCalc(xa, xb);
midy = midpointCalc(ya, yb);
distance = distanceCalc(xa, xb, ya, yb);
```

```
        cout << "\nThe line segment connecting points "
            << "(" << xa << ", " << ya << ")"
            << " and (" << xb << ", " << yb << ") is: "
            << distance << " units long.\n\nThe midpoint is: "
            << "(" << midx << ", " << midy << ").\n\n"
            << "\n2d Point Calculation Menu:\n\n"
            << "1. Perform Calculations!\n"
            << "2. Set Notation Complaint Level [" << wlevel
            << "]\n3. Quit\n\n";

        cin >> choice;

        cin.ignore(INT_MAX, '\n');
    }

    else if ( (toupper(choice) == 'P' || choice == '1') &&
        warning == false )
    {

        cout << "\nYou have chosen an option which, unfortunately, the"
         << " creator of this program\n\n\t!!Cannot figure out how to make"
         << " happen!! Sorry! \n";

    /*cin >> t;


    while ((t != '(')
    {
        cin.ignore(INT_MAX, '\n');
        cin.clear();

        cout << "Please enter your coordinates in proper form, starting"
             << " with a \"(\" symbol!\n"
             << "Please enter the first point: ";

        cin >> t;
    }

    cin >> xa;


    while (!isdigit(xa))
    {
        cin.ignore(INT_MAX, '\n');
        cin.clear();

        cout << "Please enter your coordinates in proper form! After"
             << " the \"(\" symbol, type only a number!  ";

        cin >> xa;
    }


    cin >> t;

    while (t != ',')
    {
        cin.ignore(INT_MAX, '\n');

        cout << "Please enter your coordinates in proper form! After"
             << " the first digit, type a comma!  ";

        cin >> t;
    }

    cin >> ya;

    while (!isdigit(ya))
    {
        cin.ignore(INT_MAX, '\n');

        cout << "Please enter your coordinates in proper form! After"
             << " the comma, type only a number!  ";
        cin >> ya;
    }

    while (cin.peek() != '\n' && isspace(cin.peek()))
    {
        cin.ignore();
    }
    if (cin.peek() == '\n')
    {
        cout << "You haven't finished yet! You need a closing"
        << "Parenthesis! ";

        cin >> t;

        if (t != ')')
        {
        cout << "Please enter your coordinates in proper form! After"
             << " the second number, type a \")\" symbol!  ";
        cin >> t;
        }
    }
    else
    {
        cin >> t;

        if (t != ')')
        {
        cout << "Please enter your coordinates in proper form! After"
             << " the second number, type a \")\" symbol!  ";
        cin >> t;
        }
    }



    cout << "\nPlease Enter the second Point:  ";

    cin >> t;

    while (t != '(')
    {
        cout << "Please enter your coordinates in proper form, starting"
             << " with a \"(\" symbol!";
        cin >> t;
    }


    cin >> xb;

    while (!isdigit(xb))
    {
         cout << "Please enter your coordinates in proper form! After"
             << " the \"(\" symbol, type only a number!  ";
         cin >> xb;
    }


    cin >> t;

    while (t != ',')
    {
        cout << "Please enter your coordinates in proper form! After"
             << " the first digit, type a comma!  ";
        cin >> t;
    }
```

```
      cin >> yb;

      while (!isdigit(yb))
      {
           cout << "Please enter your coordinates in proper form! After"
               << " the comma, type only a number!  ";
           cin >> yb;
      }

      while (cin.peek() != '\n' && isspace(cin.peek()))
      {
          cin.ignore();
      }
      if (cin.peek() == '\n')
      {
          cout << "You haven't finished yet! You need a closing"
          << "Parenthesis! ";

          cin >> t;

          while (t != ')')
          {
          cout << "Please enter your coordinates in proper form! After"
               << " the second number, type a \")\" symbol!  ";
          cin >> t;
          }
      }
      else
      {
          cin >> t;

          while (t != ')')
          {
          cout << "Please enter your coordinates in proper form! After"
               << " the second number, type a \")\" symbol!  ";
          cin >> t;
          }
      }


      midx = midpointCalc(xa, xb);
      midy = midpointCalc(ya, yb);
      distance = distanceCalc(xa, xb, ya, yb);


      cout << "\nThe line segment connecting points "
           << "(" << xa << ", " << ya << ")"
           << " and (" << xb << ", " << yb << ") is: "
           << distance << " units long.\n\nThe midpoint is: "
           << "(" << midx << ", " << midy << ").\n\n"*/
           cout << "\n2d Point Calculation Menu:\n\n"
               << "1. Perform Calculations!\n"
               << "2. Set Notation Complaint Level [" << wlevel
               << "]\n3. Quit\n\n";

           cin >> choice;

           cin.ignore(INT_MAX, '\n');
      }

  }



  cout << "\nOkay, then. See ya.\n\n";

  return 0;
}


double midpointCalc(double point_a, double point_b)
   {
      return (point_a + point_b)/2;
   }

double distanceCalc(double xa, double xb,
                double ya, double yb)
   {
      return sqrt(pow((xa-xb), 2) + pow((ya-yb), 2));
   }

\033]0;georgia@georgia-MT6017: ~/cplusplus\007georgia@georgia-MT6017:~/cplusplus$ C
PP ./pointinpointout
./pointinpointout.cpp***
./pointinpointout.cpp: In function â\200\230int main()â\200\231:
./pointinpointout.cpp:91:26: warning: conversion to â\200\230charâ\200\231 from
â\200\230std::basic_istream<char>::int_type {aka int}â\200\231 may alter its value
[-Wconversion]
./pointinpointout.cpp:109:26: warning: conversion to â\200\230charâ\200\231 from
â\200\230std::basic_istream<char>::int_type {aka int}â\200\231 may alter its value
[-Wconversion]
./pointinpointout.cpp:160:26: warning: conversion to â\200\230charâ\200\231 from
â\200\230std::basic_istream<char>::int_type {aka int}â\200\231 may alter its value
[-Wconversion]
./pointinpointout.cpp:176:26: warning: conversion to â\200\230charâ\200\231 from
â\200\230std::basic_istream<char>::int_type {aka int}â\200\231 may alter its value
[-Wconversion]


\033]0;georgia@georgia-MT6017: ~/cplusplus\007georgia@georgia-MT6017:~/cplusplus$ .
/pointinpointout.out

2d Point Calculation Menu:

1. Perform Calculations!
2. Set Notation Complaint Level [WARNING]
3. Quit

1

        Excellent.

Please enter the first point: 4 8

Don't forget your opening parenthesis!

Don't forget the comma between x and y values!

Don't forget your closing parenthesis!

Please Enter the second Point:  ( 5 7)

Don't forget the comma between x and y values!

The line segment connecting points (4, 8) and (5, 7) is: 1.41421 units long.

The midpoint is: (4.5, 7.5).


2d Point Calculation Menu:

1. Perform Calculations!
2. Set Notation Complaint Level [WARNING]
3. Quit

2
```

```
Type "E" to set notation complaint level to "ERROR".
This means you will not be able to continue input unless done correctly

Type "W" to set notation complaint level to "WARNING".
This means you will get warning messages,
but willstill be able to enter coordinates even in improper form.

e

2d Point Calculation Menu:

1. Perform Calculations!
2. Set Notation Complaint Level [ERROR]
3. Quit

1

You have chosen an option which, unfortunately, the creator of this program

        !!Cannot figure out how to make happen!! Sorry!

2d Point Calculation Menu:

1. Perform Calculations!
2. Set Notation Complaint Level [ERROR]
3. Quit

2
Type "E" to set notation complaint level to "ERROR".
This means you will not be able to continue input unless done correctly

Type "W" to set notation complaint level to "WARNING".
This means you will get warning messages,
but willstill be able to enter coordinates even in improper form.

w

2d Point Calculation Menu:

1. Perform Calculations!
2. Set Notation Complaint Level [WARNING]
3. Quit

1

        Excellent.

Please enter the first point: (-34 , 58)

Please Enter the second Point:  25 , 48

Don’t forget your opening parenthesis!

Don’t forget your closing parenthesis!

The line segment connecting points (-34, 58) and (25, 48) is: 59.8415 units long.

The midpoint is: (-4.5, 53).


2d Point Calculation Menu:

1. Perform Calculations!
2. Set Notation Complaint Level [WARNING]
3. Quit

q

Okay, then. See ya.
```

```
\033]0;georgia@georgia-MT6017: ~/cplusplus\007georgia@georgia-MT6017:~/cplusplus$ c
at\033[K033[K033[K033[Kexit
exit

Script done on Tue 21 Feb 2012 11:16:34 PM CST
```