User Id : weizhe.goh@digipen.edu | Started : 2020.05.26 19:38:38 | Score : 100%

Dynamic Memory, References And Namespaces

DigiPen — Assignment

Rules

Read carefully and check all rules you agree with:

- ☐ Your code must represent your own individual work. Cheating of any kind (copying someone else's work, allowing others to copy your work, collaborating, etc.) will not be tolerated and will be dealt with SEVERELY.
- ☐ Each exercise has description which must be strictly followed.
- ☐ All programs must pass all tests in the main function (when given) to get the final grade. You are not allowed to make any change in the main function in this case.
- ☐ Keep the code proper formatted (correct indentation, max line width is 40 characters).
- ☐ Every week the instructor is available **during the lab time** to discuss following matters:
  - your disagreement with rule in this card,
  - misunderstanding of the current assignment specs,
  - solution for given problems.

Part 1. Specs

- Same specs as for C++ code from the previous lab.
  - Copy/paste your code here.
  - Change it to make it 100% C++ code if it's not yet. (I am going to review your submissions in the class before this lab)
  - Minimize the use of local variables. No globals are allowed.
  - Think about efficiency. (Ex: use if-else if instead of if-if),

★ Part 1. Code

Run

```
#include <iostream>

struct Point
{
  int x, y;
};

struct AABB
{
  Point min, max;
};

AABB aabb(Point pp[], int size)
{
  AABB aabb;

  aabb.min.x = pp[0].x;
  aabb.max.x = pp[0].x;
  aabb.min.y = pp[0].y;
  aabb.max.y = pp[0].y;

  for(int i=1;i<size;i++)
  {
    if(pp[i].x <= aabb.min.x)
    {
      aabb.min.x = pp[i].x;
    }
    else if(pp[i].x >= aabb.max.x)
    {
```

Part 2. Specs

1. Define a struct BS for a bounding sphere with member data for its position (x,y,z) in 3D space and radius (r).
2. Implement a function inter() that computes existence of intersection between a point and BS and returns a boolean value based on the result.
   - The first parameter of the function is a pointer to BS.
   - The second parameter is a pointer to the point.
   - Try to implement the function without using any square root function, like sqrt().
3. Make tests for the function inter() in the main function.
   - Create all tested points and BSs in the heap using **new** operator. Delete all created points after the use.
   - Make two tests: first one for intersection case that produces true result, second for false result. Expected output is "true false". Be precise in the output!

```
      aabb.max.x = pp[i].x;
    }


    if(pp[i].y <= aabb.min.y)
    {
      aabb.min.y = pp[i].y;
    }
    else if(pp[i].y >= aabb.max.y)
    {
      aabb.max.y = pp[i].y;
    }
  }
  return aabb;
}

int main()
{
  Point pp[] =
            {{1,4},{-1,8},{4,-7},{0,3}};
  AABB value = aabb(pp,4);

  std::cout << "(" << value.min.x << ","
  << value.min.y << ")(" << value.max.x
  << ","<< value.max.y << ")";

  return 0;
}
```

```
(-1,-7)(4,8)
```

Part 2. Code 1 ☑

- Make your code with C++ that comply with given specs

Run

```
#include <iostream>

struct Point
{
  int x, y, z;
};

struct BS
{
  Point p;
  int r;
};

bool inter(BS * circle, Point *pt)
{
 int x = 0, y = 0, z = 0;
 int distanceSq = 0, radiusSq = 0;

 x = (pt->x) - circle->p.x;
 y = (pt->y) - circle->p.y;
 z = (pt->z) - circle->p.z;
```

Part 3. Specs

- Copy your code from Part 1.
- Change the function aabb to return extreme min and max points using references in the parameter list. (Do not use AABB struct at all in this code.)
- Change the main function accordingly.
- Put structs and the function into a namespace, call it DP.

```
 distanceSq = (x*x) + (y*y) + (z*z);
 radiusSq = (circle->r)*(circle->r);

 if(distanceSq <= radiusSq)
   return true;
 else
   return false;
}

int main(void)
{
  BS *sphere = new BS;

  sphere->p.x = 0;
  sphere->p.y = 0;
  sphere->p.z = 0;
  sphere->r = 5;

  Point *p1 = new Point;

  p1->x = 1;
  p1->y = 2;
  p1->z = 3;

 bool test1 = inter(sphere, p1);

  p1->x = 10;
  p1->y = 10;
  p1->z = 10;

 bool test2 = inter(sphere, p1);

 std::cout << std::boolalpha;
 std::cout << test1 << " " << test2;

 delete p1;
 delete sphere;

 return 0;
}
```

```
true false
```

Part 3. Code

Run

```
#include <iostream>

namespace DP
{
  struct Point
  {
    int x, y;
  };

  void aabb(Point*pp, int size,
            Point &min, Point &max)
  {
   min.x = pp[0].x;
```

Survey

- What is approximate number of hours you spent implementing this assignment?

5hrs

- Indicate the specific portions of the assignment that gave you the most trouble

References and Namespaces

```
   max.x = pp[0].x;
   min.y = pp[0].y;
   max.y = pp[0].y;

    for(int i=1;i<size;i++)
    {
      if(pp[i].x <= min.x)
      {
        min.x = pp[i].x;
      }
      else if(pp[i].x >= max.x)
      {
        max.x = pp[i].x;
      }

      if(pp[i].y <= min.y)
      {
        min.y = pp[i].y;
      }
      else if(pp[i].y >= max.y)
      {
        max.y = pp[i].y;
      }
    }
  }
}

using namespace DP;

int main()
{
  Point min, max;

  Point pp[] =
            {{1,4},{-1,8},{4,-7},{0,3}};

  aabb(pp, 4, min, max);

  std::cout << "(" << min.x << "," <<
  min.y << ")(" << max.x << "," <<
  max.y << ")";

  return 0;
}
```

```
(-1,-7)(4,8)
```