**НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ**

**«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ**

**імені ІГОРЯ СІКОРСЬКОГО»**

**Інститут прикладного системного аналізу**

**Кафедра системного проектування**

**Звіт**

**про виконання лабораторної роботи №13**

**з дисципліни «Алгоритмізація та програмування»**

Виконав:  
студент I курсу, групи ДА-02

Рудік Андрій Миколайович  
Прийняв:

к.т.н., доцент Безносик О. Ю.

Київ – 2020

Завдання

Створити текстовий файл, рядки якого містять числа, що задають координати вершин трикутника. Визначити вид трикутника за значеннями його сторін. Дописати в кожний рядок вхідного файлу слово, що визначає вид трикутника (прямокутний, рівнобедрений тощо).

Код програми

#include <stdio.h>

#include <string.h>

#include <locale.h>

#include <stdbool.h>

#include <stdlib.h>

#include <malloc.h>

#include <math.h>

typedef *struct* Point2d {

*float* x;

*float* y;

} Point2d;

Point2d constructPoint2d(*float* *x*, *float* *y*) {

  Point2d tmp;

  tmp.x = *x*;

  tmp.y = *y*;

  return tmp;

}

typedef *struct* Vector2d {

*float* x;

*float* y;

*float* \_length;

} Vector2d;

Vector2d constructVector2d(Point2d *a*, Point2d *b*) {

  Vector2d tmp;

  tmp.x = *b*.x - *a*.x;

  tmp.y = *b*.y - *a*.y;

  tmp.\_length = -1;

  return tmp;

}

*float* getVector2dLength(Vector2d \**v*) {

  return *v*->\_length != -1 ? *v*->\_length : (*v*->\_length = sqrt(pow(*v*->x, 2) + pow(*v*->y, 2)));

}

*float* dotProduction(Vector2d *a*, Vector2d *b*) {

  return *a*.x \* *b*.x + *a*.y \* *b*.y;

}

typedef *struct* Triangle2d {

  Point2d \*points;

  Vector2d \*sides;

} Triangle2d;

Triangle2d constructTringle2d(Point2d \**points*) {

  Triangle2d tmp;

  tmp.points = *points*;

  tmp.sides = (Vector2d\*)malloc(3 \* sizeof(Vector2d));

  tmp.sides[0] = constructVector2d(*points*[0], *points*[1]);

  tmp.sides[1] = constructVector2d(*points*[1], *points*[2]);

  tmp.sides[2] = constructVector2d(*points*[2], *points*[0]);

  return tmp;

}

*void* destructTriangle2d(Triangle2d *triangle*) {

  free(*triangle*.points);

  free(*triangle*.sides);

}

typedef *struct* Triangle2dArray {

  Triangle2d \*arr;

*int* length;

} Triangle2dArray;

Triangle2dArray constructTriangle2dArray(Triangle2d \**arr*, *int* *length*) {

  Triangle2dArray tmp;

  tmp.length = *length*;

  tmp.arr = *arr*;

  return tmp;

}

*void* destructTriangle2dArray(Triangle2dArray *triangles*) {

  for (*int* i = 0; i < *triangles*.length; i++) {

    destructTriangle2d(*triangles*.arr[i]);

  }

  free(*triangles*.arr);

}

bool isTriangle(Triangle2d *triangle*) {

  return getVector2dLength(&*triangle*.sides[0]) + getVector2dLength(&*triangle*.sides[1]) > getVector2dLength(&*triangle*.sides[2]) &&

    getVector2dLength(&*triangle*.sides[0]) + getVector2dLength(&*triangle*.sides[2]) > getVector2dLength(&*triangle*.sides[1]) &&

    getVector2dLength(&*triangle*.sides[1]) + getVector2dLength(&*triangle*.sides[2]) > getVector2dLength(&*triangle*.sides[0]);

}

bool isEquilateralTriangle(Triangle2d *triangle*) {

  return getVector2dLength(&*triangle*.sides[0]) == getVector2dLength(&*triangle*.sides[1]) &&

    getVector2dLength(&*triangle*.sides[0]) == getVector2dLength(&*triangle*.sides[2]);

}

bool isIsoscelesTriangle(Triangle2d *triangle*) {

  return getVector2dLength(&*triangle*.sides[0]) == getVector2dLength(&*triangle*.sides[1]) ||

    getVector2dLength(&*triangle*.sides[0]) == getVector2dLength(&*triangle*.sides[2]) ||

    getVector2dLength(&*triangle*.sides[1]) == getVector2dLength(&*triangle*.sides[2]);

}

bool isRightTriangle(Triangle2d *triangle*) {

  return !(dotProduction(*triangle*.sides[0], *triangle*.sides[1]) &&

    dotProduction(*triangle*.sides[0], *triangle*.sides[2]) &&

    dotProduction(*triangle*.sides[2], *triangle*.sides[1]));

}

*char*\* getName() {

*char* \*name = (*char*\*)malloc(128);

  printf("Введите имя файла с которым желаете работать: ");

  fflush(stdin);

  gets(name);

  name = (*char*\*)realloc(name, strlen(name) + 1);

  return name;

}

*void* strip(*char* \**str*, *char* *c*) {

*int* length = strlen(*str*);

*int* si = 0;

  while (*str*[si] == *c* && si < length) si++;

  if (si) strcpy(*str*, *str* + si);

*int* ei = length - si - 1;

  while (*str*[ei] == *c* && ei > 0) ei--;

*str*[ei + 1] = 0;

}

bool isDigit(*char* *c*) {

  return '0' <= *c* && *c* <= '9';

}

*int* getNumberLength(*char* \**buff*) {

*int* i = 1;

  if (!isDigit(\**buff*)) {

    if (\**buff* != '-') return 0;

    if (!isDigit(\*(*buff* + i++))) return 0;

  }

  while (isDigit(\*(*buff* + i)) || \*(*buff* + i) == ',') i++;

  return i;

}

bool isValidPointString(*char* \**buff*) {

  strip(*buff*, ' ');

*int* shift = 0;

  if (

    !(shift = getNumberLength(*buff*)) ||

    \*(*buff* += shift) != ' ' ||

    !(shift = getNumberLength(++*buff*))

  ) return false;

  return !\*(*buff* + shift);

}

*char*\* initTriangleFile() {

*char* \*fname = getName();

  FILE \*f = fopen(fname, "w");

*char* \*buff = (*char*\*)malloc(10);

*int* n;

  printf("\nВведите количество треугольников которые вы хотите добавить: ");

  fflush(stdin);

  scanf("%d", &n);

  fprintf(f, "%d\n", n);

  printf("\n");

  for (*int* i = 0; i < n; i++) {

    for (*int* j = 0; j < 3;) {

      printf("Введите координаты x, y вершини треугольника через пробел: ");

      fflush(stdin);

      gets(buff);

      if (!isValidPointString(buff)) {

        printf("Не валидные координаты вершины\n");

        continue;

      }

      fputs(buff, f);

      if (j != 2) {

        fputc(';', f);

      }

      j++;

    }

    printf("\n");

    fputc('\n', f);

  }

  fclose(f);

  free(buff);

  return fname;

}

Triangle2dArray readTriangleFile(*char* \**fname*) {

  FILE \*f = fopen(*fname*, "r");

*int* n;

  fscanf(f, "%d\n", &n);

  Triangle2d \*triangles = (Triangle2d\*)malloc(n \* sizeof(Triangle2d));

  for (*int* i = 0; i < n; i++) {

    Point2d \*points = (Point2d\*)malloc(3 \* sizeof(Point2d));

*char* buff[100];

    fgets(buff, sizeof buff, f);

*char* \*pch = strtok(buff, ";\n");

*int* j = 0;

    while (pch != NULL) {

*float* x, y;

      sscanf(pch, "%f %f", &x, &y);

      points[j] = constructPoint2d(x, y);

      pch = strtok(NULL, ";\n");

      j++;

    }

    triangles[i] = constructTringle2d(points);

  }

  fclose(f);

  return constructTriangle2dArray(triangles, n);

}

*void* getTypeOfTriangle(*char* \**type*, Triangle2d *triangle*) {

  if (isTriangle(*triangle*)) {

    if (isEquilateralTriangle(*triangle*)) {

      strcpy(*type*, "Правильный треугольник");

    } else if (isIsoscelesTriangle(*triangle*)) {

      if (isRightTriangle(*triangle*)) {

        strcpy(*type*, "Равнобедренный прямоугольный треугольник");

      } else {

        strcpy(*type*, "Равнобедренный треугольник");

      }

    } else if (isRightTriangle(*triangle*)) {

      strcpy(*type*, "Прямоугольный треугольник");

    } else {

      strcpy(*type*, "Треугольник");

    }

  } else {

    strcpy(*type*, "Не треугольник");

  }

}

*void* rewriteTriangleFile(*char* \**fname*, Triangle2dArray *triangles*) {

  FILE \*f = fopen(*fname*, "w");

  fprintf(f, "%d\n", *triangles*.length);

  for (*int* i = 0; i < *triangles*.length; i++) {

*char* \*type = (*char*\*)malloc(128);

    getTypeOfTriangle(type, *triangles*.arr[i]);

    Point2d \*points = *triangles*.arr[i].points;

    for (*int* i = 0; i < 3; i++) {

      fprintf(f, "%.2f %.2f", points[i].x, points[i].y);

      if (i != 2) {

        fputc(';', f);

      } else {

        fputc(' ', f);

      }

    }

    fputs(type, f);

    fputc('\n', f);

    free(type);

  }

  fclose(f);

}

*int* main() {

  setlocale(LC\_ALL, "rus");

*char* \*fname = initTriangleFile();

  Triangle2dArray triangles = readTriangleFile(fname);

  rewriteTriangleFile(fname, triangles);

  printf("Готово!");

  free(fname);

  destructTriangle2dArray(triangles);

  return 0;

}

Результат роботи програми наведено на рисунках 1, 2

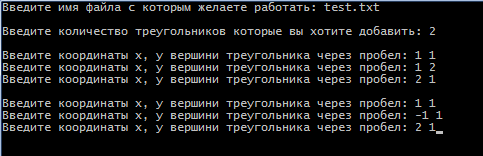


Рисунок 1 – Вхідні данні

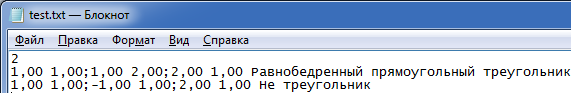


Рисунок 1 – Файл test.txt

Висновки

В ході виконання лабораторної роботи №13 я отримав навички роботи з файлами на мові С.