acm.mipt.ru

олимпиады по программированию на Физтехе

Поиск
Поиск
Раздел «Язык Си»
Главная
Зачем учить С?
Определения
Инструменты:
Поиск
Изменения
Index
Статистика
Разделы
Информация
Алгоритмы
Язык Си
Язык Ruby
Язык Ассемблера
El Judge
Парадигмы
Образование
Сети
Objective C
Logon>>

Проект или совместная работа над задачей

Если Вы решаете небольшую задачу, то ее, конечно, удобнее решать одному.

НО, если:

1. в задаче приходится выполнять множество действий
2. задача требует особых подходов к реализации функций и эти подходы знают разные люди
3. у Вас очень мало времени
4. наработки для этой задачи Вы хотели бы использовать в будущем

, то разумно представить реализацию задачи как программный проект.

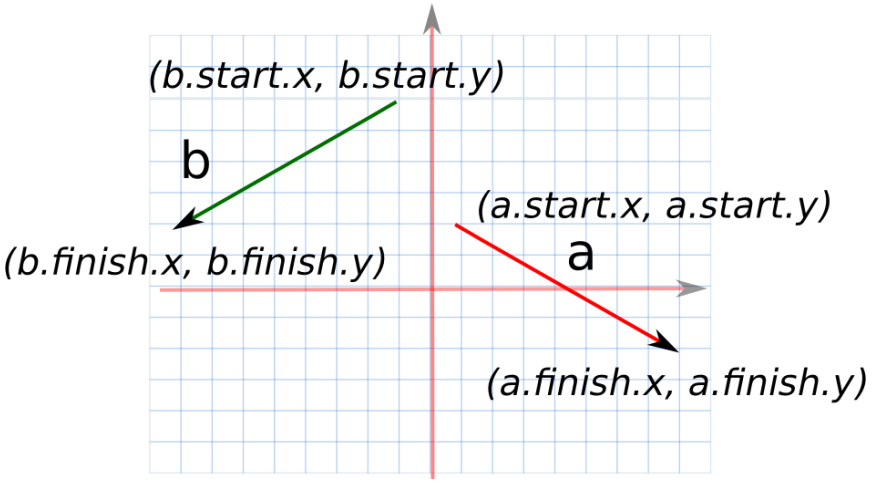
Проект требует более подробную проработку следующих вопросов: какие типы данных, какие инструменты нужны или желательны для решения задачи.

Когда эти вопросы будут решены, необходимо описать типы данных и все функции-инструменты. Все участники проекта должны СТРОГО следовать этому описанию.

Маленькая задача про отрезки*

Два направленных отрезка на плоскости заданы началом и концом (координаты x и y).

Требуется определить, являются ли эти отрезки равными при наложении их друг на друга.



То есть, нужно начало одного отрезка совместить с началом другого и проверить, совпадают ли после этого их концы.

Для решения нужно:

1. вычислять длину отрезка
2. получать координаты отрезка
3. печатать координаты отрезка
4. двигать отрезок на плоскости
5. сравнивать координаты концов отрезка

Рассмотрим, какими понятиями было бы удобно пользоваться при решении этой задачи. Это: **отрезок, концы отрезка, точка, координаты**

Инструменты, которые облегчили бы решение задачи: **получение и печать координат отрезков, перемещение отрезка, сравнение координат.**

Опишем так называемый **процедурный** подход.

Каждый инструмент - это некоторая процедура (функция), которая работает с нужными типами данных. Правильная последовательность использования процедур (функций) приведет к решению задачи.

В проекте все создаваемые типы данных и интерфейсы функций описываются в заголовочных файлах. Простейший вариант проекта выглядит так:

Заголовочный файл (prim.h)	Файл реализации функций (prim.c)	Программа, которая использует функции (test.c)

`// Новый :) тип данных`
`typedef int Coord;`
`// Интерфейс функции`
`int add(Coord,Coord);`
	`//включить заголовочный файл`
	`#include "prim.h"`
	`// Реализация функции add`
	`int add(int a, int b){`
		`// Использование функции add`
		`#include <stdio.h>`
		`//включаем НАШ заголовочный файл`
		`#include "prim.h"`

```
   return a + b;
};
```

```
int main(){
  int a, b, c;
  scanf("%d%d", &a, &b);
// Используем функцию add
  c = add(a, b);
  printf("a + b = %d\n",c);
}
```

Как собрать проект*

Конечно, все написанное должно быть откомпилировано и собрано.

Можно сразу скомпилировать все нужные файлы и собать в работающий файл, а можно это сделать по частям.

Перый способ (все вместе)

```
gcc test.c prim.c -o test
```

 Заголовочный файл НИКОГДА в строку компиляции не включается. Компилятор ищет его самостоятельно.

Второй способ (по-отдельности)

```
>gcc -c prim.c
>gcc -c test.c
>gcc prim.o test.o -o test
```

В этом случае получаются два объектных файла (.o). Если с файл не изменялся, то его не обязательно компилировать.

Вернемся к задаче про отрезки. Опишем нужные типы. Можно описать их в одном заголовочном файле, а можно использовать несколько.

Описание координаты

Заголовочный файл (coord.h)

```
// описание типа "координата"
typedef struct{
  int x,y;
}Coord;
// интерфейсы функций:
// чтение значений полей с консоли
void setCoord(Coord*);
// печать координат
void printCoord(Coord);
// сравнение координат
int cmpCoord(Coord, Coord);
```

Файл реализации функций (coord.c)

```
#include <stdio.h>
// включаем НАШ заголовок
#include "coord.h"
// реализация функций:

void setCoord(Coord *a){
   scanf("%d%d",&(a->x),&(a->y));
};

void printCoord(Coord a){
  printf("%d %d",a.x,a.y);
};

int cmpCoord(Coord a, Coord b){
   if(a.x == b.x && a.y == b.y)
     return 1;
   return 0;
};
```

Описание отрезка

Заголовочный файл (line.h)

```
// Заголовок coord здесь
// нужно включать
// потому что используется
// тип Coord и функции к нему
#include "coord.h"

// тип "отрезок"
typedef struct{
  Coord start; // начало отрезка
  Coord finish; // конец
}Line;
// считать значения с консоли
void setLine(Line*);
// напечатать концы отрезка
void printLine(Line);
// вычисление длины
float Length(Line);
```

Файл реализации функций (line.c)

```
#include <stdio.h>
#include <math.h>
#include "line.h"

void setLine(Line *lin){
   Coord a,b;
   setCoord(&a);
   setCoord(&b);
   lin->start = a;
   lin->finish = b;

};

void printLine(Line lin){
   printCoord(lin.start);
   printCoord(lin.finish);
};
```

```
// сравнение двух отрезков
int cmpLine(Line, Line);

// перемещение начала отрезка в точку Coord
// и вычисление положения конца отрезка
void move(Line, Coord);
```

```
float Length(Line lin){
  float r;
  float kv1, kv2;
  kv1 = (lin.start.x-lin.finish.x);
  kv2 = (lin.start.y-lin.finish.y);
  r = sqrt(kv1 * kv1 + kv2 * kv2);
  return r;
};
```

Проверка работы функций

```
#include <stdio.h>
#include <stdlib.h>
// здесь включаем файл line.h
// там описание типа и функций
#include "line.h"
/*
Заметим, что не нужно больше описывать
типы и функции.
Они все уже описаны и реализованы
Здесь можно это все использовать
*/

int main(){

  Line otrezok; // объявление отрезка
  setLine(&otrezok); // заполнение значений
// вычисление длины отрезка
  printf("lin=%0.2f\n",Length(otrezok));
// печать отрезка
  printLine(otrezok);

  return 0;
}
```

Статическая библиотека.

Допустим, Вы написали и отладили все функции, и они прекрасно работают. Кроме этой задачи есть еще множество других задач, в которых эти функции будут полезны.

Писать их заново или переносить файлы в новый прект хлопотно и неразумно.

Чтобы избежать этого скомпилированные функции обычно включают в **библиотеки**. Рассмотрим использование **статической библиотеки**. Функции статической библиотеки при линковке включаются в исполняемый файл (все функции и библиотеки). Это влияет на размер исполняемого файла. Именно поэтому при создании библиотеки нужно придерживаться принципа "ничего лишнего".

При создании библиотеки используется архиватор **ar**.

Создание библиотеки

1. создание объектных файлов реализаций функций
2. создание архива с названием libназвание_библиотеки.a
3. включение объектных файлов в архив
4. ранжирование архива для быстрого поиска функций
5. помещение библиотечного файла в специальный каталог

Как правило, заголовочные файлы проекта собираются в каталог **include**, а библиотечные - в **lib**.

При компиляции и линковке должны использоваться ключи: **-Iкаталог_с_заголовками**, **-Lкаталог_с_библиотечными_файлами** и **-lназвание_библиотеки**

```
>gcc -c line.c
>gcc -c coord.c
>ar -rc libline.a line.o coord.o
>ranlib libline.a
```

Далее создаем каталоги для заголовков и библиотек, помещаем их туда. После этого можно уже компилировать программу, которая использует функции из библиотеки.

```
>mkdir lib
>mkdir include
>mv libline.a lib/
>mv coord.h line.h include/
```

Компиляция

```
>gcc test.c -I./include -L./lib -lline -lm
```

Одна используемая библиотека - наша библиотека **line**, а вторая - системная математическая **m**.

Задачи

1. Создать библиотеку из описанных выше функций.
2. Реализовать функции **int cmpCoord(Coord, Coord)**, **int cmpLine(Line, Line)** и **void move(Line, Coord)** в отдельных файлах реализации. Добавить их в существующую библиотеку. Не забыть использовать **ranlib**.
3. Решить задачу про отрезки.
4. Используя данную библиотеку решить задачу сравнения двух треугольников.

Примеры использования библиотек и классов для работы с системными файлами. systemprim.tar.gz:

-- TatyanaOvsyannikova2011 - 27 Oct 2014

Attachment	Action	Size	Date	Who	Comment
otrezok_class2.png	manage	57.3 K	27 Oct 2014 - 17:42	TatyanaOvsyannikova2011	
otrezok_class3.png	manage	53.1 K	27 Oct 2014 - 20:14	TatyanaOvsyannikova2011	
systemprim.tar.gz	manage	4.4 K	13 Oct 2016 - 10:50	TatyanaOvsyannikova2011	
script_begint.pdf	manage	310.9 K	16 May 2019 - 15:30	TatyanaOvsyannikova2011	