# Волга ИТ - 2012. Прикладное Программирование

## Заочный тур

### Постановка задачи

Требуется создать приложение, которое помогло бы пользователю наилучшим образом спланировать закупку продуктов в нескольких окрестных магазинах.

В качестве входных данных приложение получает файл с информацией о магазинах: названия, координаты и прайс-листы (списки продуктов и цен на них).
Пользователь (покупатель) задает собственные координаты, выбирает нужные ему продукты из сводного списка и критерий оптимальности - минимизировать общую стоимость покупок или суммарную длину маршрута с возвращением в исходную точку.
Программа составляет для него оптимальный план похода за покупками: в каких магазинах, в каком порядке и что покупать.

### Входные данные

Информация о магазинах задается в XML-файле, например:

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<!DOCTYPE shops SYSTEM "shops.dtd">

<shops>
	<shop id="No1">
		<name>Food Lion</name>
		<coordinates x="100" y="50" />
		<products>
			<product code="eggs" price="200">Eggs</product>
			<product code="sausage" price="150">Sausage</product>
			<product code="bacon" price="400">Bacon</product>
			<product code="spam" price="99.90">Spam</product>
		</products>
	</shop>
	<shop id="No2">
		<name>Bottom Dollar Food</name>
		<coordinates x="-100" y="-200" />
		<products>
			<product code="sausage" price="150">Sausage</product>
			<product code="bacon" price="500">Bacon</product>
			<product code="spam" price="49.90">Spam</product>
		</products>
	</shop>
</shops>
```

DOCTYPE (`shops.dtd`) определен следующим образом:

```
<!ELEMENT shops (shop+)>

<!ELEMENT shop (name,coordinates,products)>
<!ATTLIST shop
      id    ID #REQUIRED
>

<!ELEMENT name (#PCDATA)>

<!ELEMENT coordinates EMPTY>
<!ATTLIST coordinates
      x     CDATA #REQUIRED
      y     CDATA #REQUIRED
>

<!ELEMENT products (product+)>

<!ELEMENT product (#PCDATA)>
<!ATTLIST product
      code  CDATA #REQUIRED
      price CDATA #REQUIRED
>
```

Идентификаторы магазинов (`id`) уникальны, названия (`name`) предназначены только для отображения на карте и в плане покупок, если Вы выводите его в текстовом виде.
Товары из разных магазинов с одинаковым кодом (`code`) вида товара считаем одинаковыми (взаимозаменяемыми), текст названия товара играет декоративную роль.
При покупке товаров по списку также будем учитывать только совпадение кодов.
Примем, что в ассортименте каждого магазина не более одного наименования товара каждого вида. Количество товара в штуках не учитывается (в данном магазине его либо неограниченно много, либо вообще нет).
Будем считать, что по городским улицам из каждого магазина можно дойти до любого другого и до начальной точки, при этом расстояние определяется как сумма абсолютных разностей координат, например, длина пути от (0, 0) до (100, -50) равна 150.
Программа должна поддерживать расчет не менее чем для 10 магазинов.

## Интерфейс командной строки

Программа должна поддерживать запуск из командной строки:
`$ your-shopping-planner` *`shops.xml shopping-list.xml shopping-plan.xml`*

где `your-shopping-planner` - ваша программа,
*`shops.xml`* - имя входного файла с информацией о магазинах,
*`shopping-list.xml`* - имя входного файла “заказа” (исходная точка, список покупок, критерий оптимизации),
*`shopping-plan.xml`* - имя выходного файла с планом покупок.

“Файл заказа” имеет следующий формат:

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<!DOCTYPE shoppingList SYSTEM "shopping-list.dtd">

<shoppingList>
      <coordinates x="50" y="50" />       <!-- home address -->
      <parameters>
<!-- minimize either cost... -->
            <cost>MIN</cost>
<!-- ...or distance traveled
            <distance>MIN</distance>
 -->
      </parameters>
      <products>
            <product code="spam">Spam</product>
            <product code="sausage">Sausage</product>
            <product code="eggs">Eggs</product>
      </products>
</shoppingList>
```

DOCTYPE `shopping-list.dtd`:

```
<!ELEMENT shoppingList (coordinates,parameters,products)>

<!ELEMENT coordinates EMPTY>
<!ATTLIST coordinates
      x     CDATA #REQUIRED
      y     CDATA #REQUIRED
>

<!ELEMENT parameters (cost|distance)>
<!ELEMENT cost (#PCDATA)>           <!-- MIN -->
<!ELEMENT distance (#PCDATA)>       <!-- MIN -->

<!ELEMENT products (product+)>

<!ELEMENT product (#PCDATA)>
<!ATTLIST product
      code  CDATA #REQUIRED
>
```

В этом файле задан список продуктов для покупки, определены координаты начальной точки маршрута (`coordinates`), выбран критерий оптимизации -

- минимальная общая стоимость (`cost = MIN`)
- или минимальная общая длина маршрута (`distance = MIN`) из начальной точки через выбранные магазины с возвращением обратно в начальную точку.

Имена продуктов в этом файле не учитываются, мы ищем совпадение кодов вида товара (`code`) с соответствующими позициями в ассортименте магазина.

Программа должна сгенерировать под заданным именем выходной XML-файл “плана покупок” следующего вида:

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<!DOCTYPE shoppingPlan SYSTEM "shopping-plan.dtd">
<shoppingPlan>
	<coordinates x="50" y="50"/>
	<totals>
		<cost>399.9</cost>
		<distance>900.0</distance>
	</totals>
	<shops>
		<shop id="No1">
			<name>Food Lion</name>
			<products>
				<product code="sausage">Sausage</product>
				<product code="eggs">Eggs</product>
			</products>
		</shop>
		<shop id="No2">
			<name>Bottom Dollar Food</name>
			<products>
				<product code="spam">Spam</product>
			</products>
		</shop>
	</shops>
</shoppingPlan>
```

DOCTYPE `shopping-plan.dtd`:

```
<!ELEMENT shoppingPlan (coordinates,totals,shops)>

<!ELEMENT coordinates EMPTY>
<!ATTLIST coordinates
	x	CDATA #REQUIRED
	y	CDATA #REQUIRED
>

<!ELEMENT totals (cost,distance)>
<!ELEMENT cost (#PCDATA)>
<!ELEMENT distance (#PCDATA)>

<!ELEMENT shops (shop*)>

<!ELEMENT shop (name?,products)>
<!ATTLIST shop
```

```
      id    ID #REQUIRED
>

<!ELEMENT name (#PCDATA)>

<!ELEMENT products (product+)>

<!ELEMENT product (#PCDATA)>
<!ATTLIST product
      code  CDATA #REQUIRED
      price CDATA #IMPLIED
>
```

Это список покупок в порядке обхода магазинов, начиная и заканчивая в заданной точке (`coordinates`). Суммарная стоимость и общая длина маршрута явно указаны в элементе `totals` для удобства сравнения планов. Обязательные поля - это `id` магазинов и `code` товаров, прочие названия - опционально для наглядности.

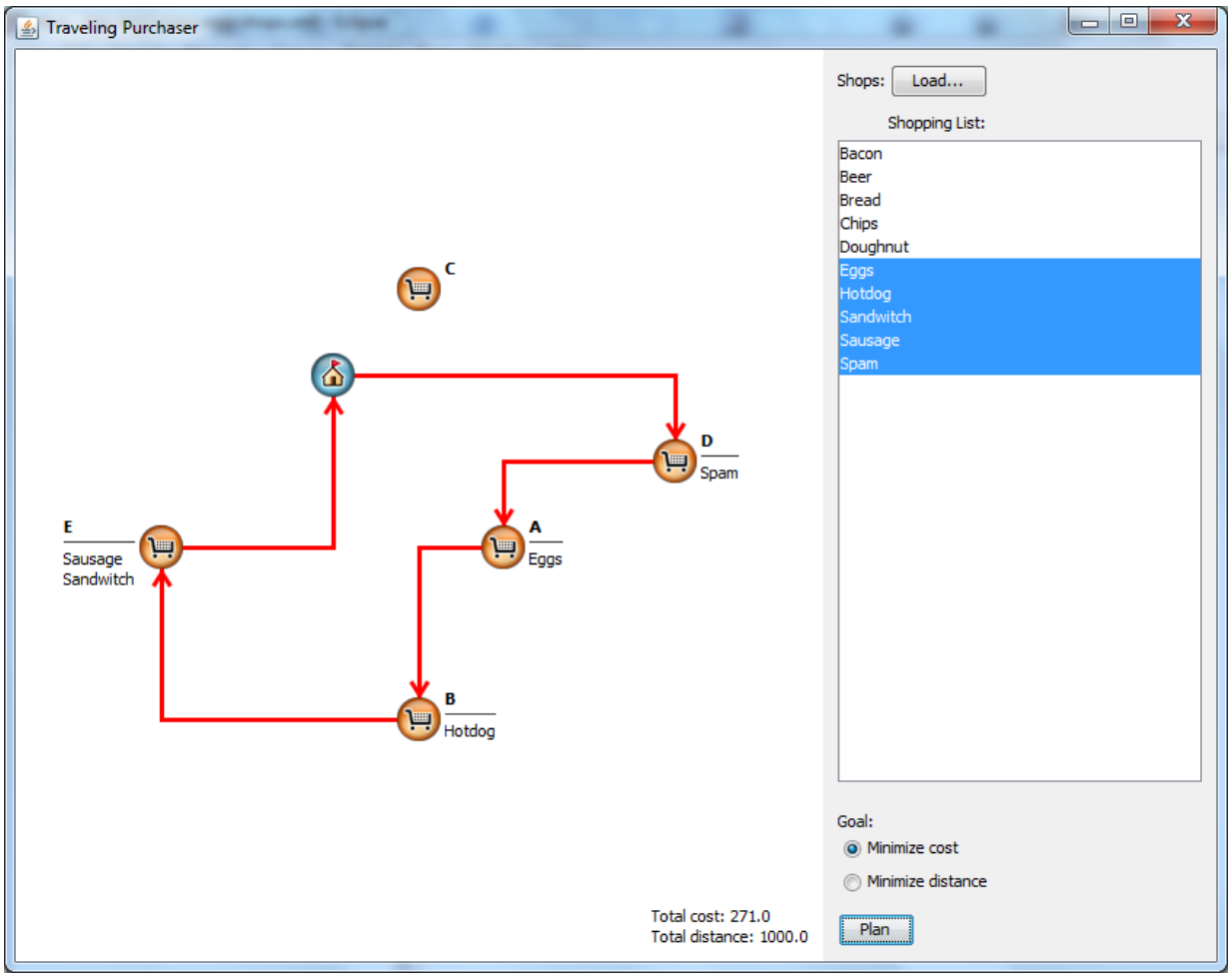
Пожалуйста, укажите в `README.TXT`, находит ли Ваша программа оптимальное или “близкое к оптимальному” решение и по каким критериям выбирает итоговый вариант, если находит несколько решений одинаковой “стоимости”.

### Графический интерфейс пользователя

Будучи запущена без параметров, программа должна отобразить графический интерфейс, который предоставляет пользователю возможность:

- загрузить данные о магазинах,
- выбрать продукты для покупки из сводного ассортимента,
- выбрать начальную точку маршрута,
- выбрать критерий минимизации - стоимость или расстояние,
- увидеть предлагаемый план покупок на схеме расположения магазинов - в каком порядке их обходить и что покупать в каждом, а также общую стоимость и суммарную длину маршрута.

Например:




Допускаются и приветствуются другие варианты интерфейса, удовлетворяющие перечисленным требованиям.

## Дополнительные условия

Программа предоставляется в виде исходных текстов и файла проекта для среды разработки:

- C++ : Visual Studio 2010 Express или Qt Creator
- C#, VB.NET : Visual Studio 2010 Express на .NET 4
- Java 6 : Eclipse 3.7.

Соберите все файлы в архив с названием следующего формата на латинице: `имя.фамилия_год-месяц-день.zip` (например, `sergey.brin_2012-10-31.zip`). Включите все необходимые файлы проекта для соответствующей IDE (`.vcproj` / `.csproj` + `.sln`, `.project` + `.classpath`...). Исключите файлы, создаваемые IDE заново в процессе импорта, компиляции и сборки (`.exe`, `.obj`, `.class`...). В идеале, программа должна собираться без дополнительных настроек после импорта вашего проекта в нашу среду разработки, установленную “с параметрами по умолчанию”.

В общем случае, не допускается использование сторонних библиотек, не входящих в стандартную поставку среды разработки, за исключением:

- библиотеки для чтения/записи XML в C++, можно выбрать вариант, подключаемый в виде исходных кодов, например, TinyXML или PugiXML (разместите их в отдельном каталоге вашего проекта);
- библиотеки Google Guava в Java (включите guava-13.x.x.jar в каталог lib вашего проекта).

Если вы считаете, что не можете обойтись без некоторой сторонней библиотеки (например, MFC или ATL/WTL для реализации графического интерфейса), предварительно напишите нам на volgait2012appprog@gmail.com для согласования.

Проект должен включать файл `README.TXT`, в который нужно включить характеристику алгоритма планирования покупок (см. выше - оптимальность и используемые эвристики для оценки решений, чтобы мы могли однозначно интерпретировать результаты на тестовых данных) и инструкцию по сборке программы из исходных текстов (дополнительные условия, настройки среды - постарайтесь свести их к минимуму).

## Оценка

Не обязательно решать задачу на 100% - мы готовы принимать и рассматривать решения, не удовлетворяющие полностью всем перечисленным требованиям; напишите об ограничениях в `README.TXT`.

Итоговая оценка будет складываться из результатов прогона на тестовых задачах с командной строки, результатов ручного тестирования GUI и, возможно, анализа исходных текстов.

Вопросы и уточнения направляйте по адресу: volgait2012appprog@gmail.com

Желаем успеха!
Команда “Прикладное программирование” Волга ИТ-2012