**Desenvolvimento**

**Dados**

Os dados utilizados para ordenação nessa aplicação foram do desmatamento da Amazônia Legal, fornecidos em quilômetros quadrados pelo Boletim Transparência Florestal da Amazônia Legal, através do site Imazon. Estes dados são detectados mensalmente pelo Sistema de Alerta de Desmatamento (SAD).

O Imazon é um instituto de pesquisa cuja missão é promover o desenvolvimento sustentável na Amazônia por meio de estudos, apoio à formulação de políticas públicas, disseminação ampla de informações e formação profissional.

Segue o link de onde os dados utilizados no programa foram importados:

<http://www.imazon.org.br/publicacoes/transparencia-florestal/transparencia-florestal-amazonia-legal?b_start:int=0>



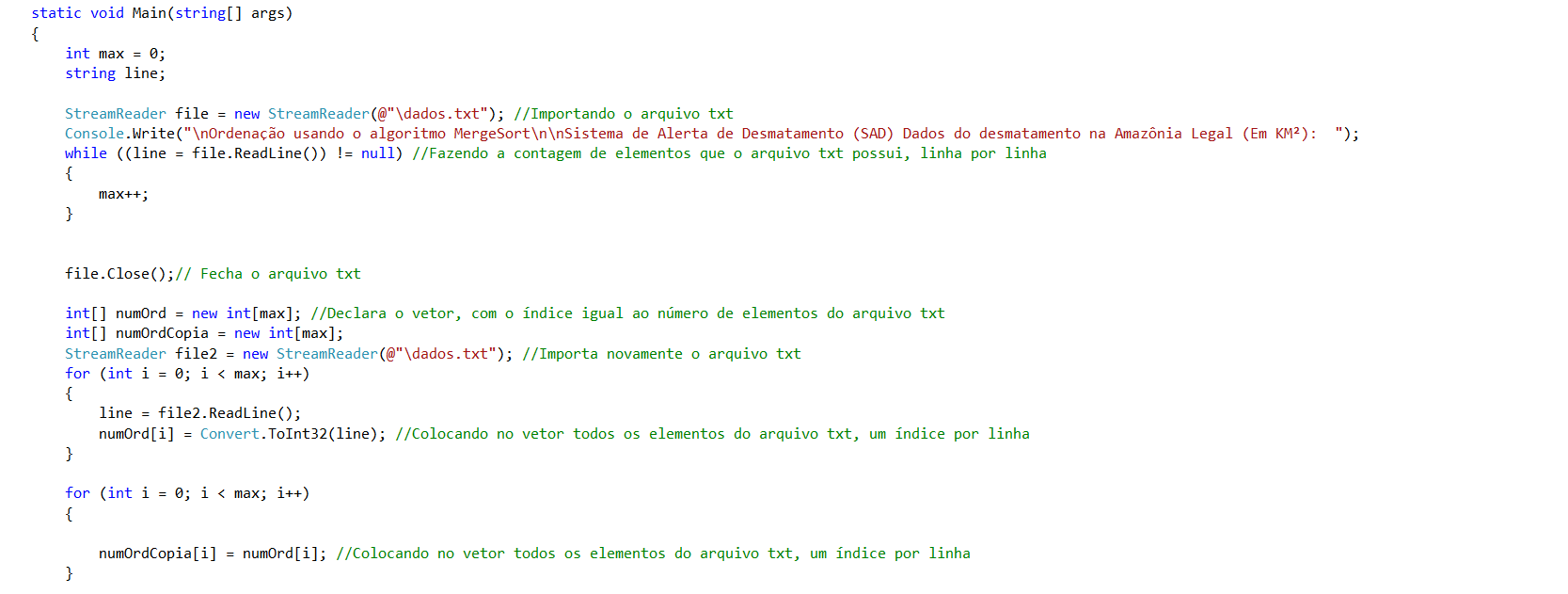
Conforme os dados eram puxados do site Imazon, eram colocados em um documento .txt criado na unidade C: do computador, linha por linha e em ordem mensal, no caso o primeiro valor colocado no documento corresponde ao valor de abril de 2008, e assim sucessivamente até o último valor que é de março de 2014. Conforme inserido valores uma linha abaixo no documento .txt, o software irá ordena-los e já mostrar para o usuário como valor correspondente ao mês seguinte. Isso será mostrado mais a frente neste tópico. Segue a imagem do documento dados.txt que foi usado para armazenamento dos dados:



Conforme foi dito, os dados organizados linha por linha, que é a forma que o programa irá identificar cada valor e coloca-los em um vetor, para assim começar a realizar a operação de ordenação em ambos os métodos MergeSort e QuickSort.

**MergeSort**

Começa agora a parte do desenvolvimento da aplicação, vamos começar pelo MergeSort. Após criado o .txt com os dados colocados linha por linha, é necessário que estes sejam transferidos para um vetor. Então é necessário que o programa localize o arquivo dados.txt que está no C:/. Para isso foi importada a classe System.IO e foi usado o comando StreamReader. Feito isso, inicialmente é necessário uma contagem de elementos contidos no .txt (para definir o índice do vetor), e depois é criado o vetor e os valores são armazenados, cada linha é correspondente a um espaço no vetor numOrd:

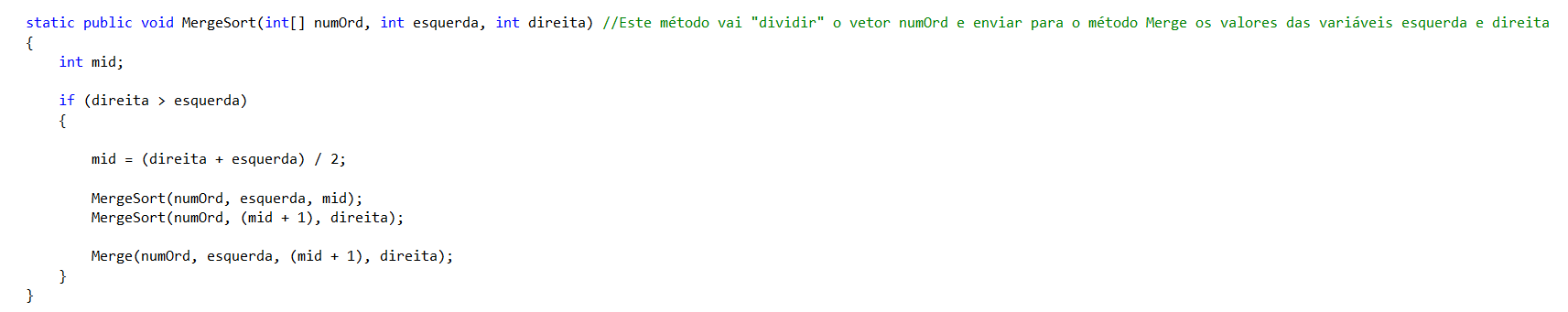


O código acima apenas dá uma introdução ao programa, faz a contagem de dados do arquivo .txt, cria os vetores que serão usados no programa e passa os dados do arquivo para os mesmos vetores (O vetor numOrdCopia será usado para integração dos dados com os meses e anos correspondentes, isso será mostrado no final da apresentação do programa).

Agora que os dados já estão devidamente alocados no vetor, começará a ordenação dos mesmos. Um método específico é chamado para iniciar a ordenação, já enviando os dados do vetor numOrd e dos índices da ponta do vetor, no caso o zero e o valor de max (que contém o número de dados do .txt) menos 1, através dessa linha de código:

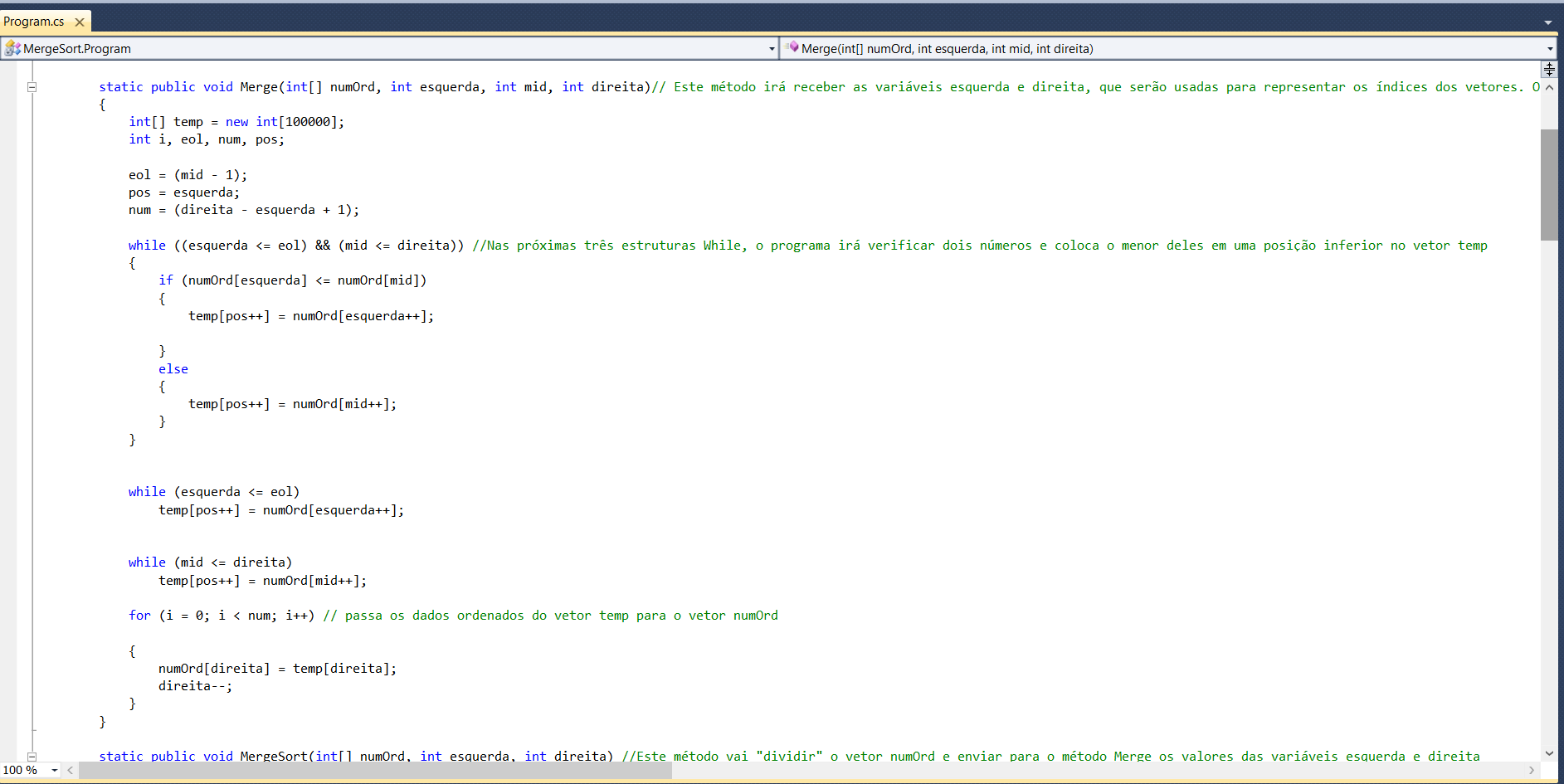


Depois disso o método MergeSort é executado e segue as riscas do algoritmo. No método MergeSort ele “divide” os índices dos vetores e envia os valores para esquerda (que no caso seria o menor valor do vetor) e direita (o maior valor do vetor). Assim ele chama o método Merge, que esse sim fará de fato o processo de ordenação. A princípio veja o código do método MergeSort:



Perceba que no final do método ele está chamando o outro método (Merge) que será apresentado na próxima página.

Até agora, pouco se viu de ordenação de dados e realmente como o método MergeSort funciona na prática. Porém, a ordenação em si ocorre no método que será apresentado agora. O método Merge recebe os dados, tanto do vetor numOrd, quanto os valores dos índices que deverão ser ordenados. No MergeSort os dados são divididos, comparados e depois são mesclados novamente, já ordenados. Para auxiliar nessa operação foi criado o vetor temp, que terá um papel importante na ordenação. São criadas também três variáveis locais (eol, num, pos) para auxiliarem também na ordenação dos dados. Após criado todas essas estruturas, o programa entra em um laço de repetição while, que será mais usado perto do final da execução do programa, já que no começo ele compara apenas duas posições no vetor, tendo em vista que este está virtualmente “dividido” em várias duplas pelo método MergeSort. Portanto, primeiramente será comparado cada dupla do vetor, e depois de todas ordenadas, estas serão mescladas gradativamente. A comparação dos valores é feito através de uma estrutura IF/ELSE que verifica qual dos dois valores é menor, e este é colocado em uma posição inferior no vetor temp. Feito isso o outro valor em questão é colocado em uma posição maior do vetor temp e depois estes mesmos valores são transferidos do vetor temp novamente para o vetor numOrd. Este processo será repetido diversas vezes durante a execução do programa, quanto mais dados o .txt armazenar, mais vezes os métodos serão executados. Segue a tela do código mostrando o método Merge:



Ao final do método Merge, o programa volta automaticamente para o método MergeSort, que irá indicar quais são os índices do vetor numOrd que deverão ser ordenados agora.

O vetor temp é zerado toda vez que o método Merge começa a ser executado. Isso possibilita que os valores se aloquem temporariamente em outros índices do vetor, enquanto o mesmo não está completamente ordenado.

Feita a ordenação do vetor temp e passado para o vetor numOrd, se encerra aqui o algoritmo MergeSort. Agora começa a etapa de saída de dados, ou seja, a exibição dos dados para o usuário. Além de simplesmente mostrar o valor contido no arquivo .txt, é necessário também ligar o valor numérico em KM² com o mês e ano correspondente. Porém esta parte do desenvolvimento será apresentada depois do método QuickSort, pois as linhas de código usadas para exibição dos dados foram basicamente idênticas.

**QuickSort**

Baseado no mesmo arquivo .txt, será feita a ordenação de dados usando agora o método QuickSort. É importante ressaltar que, para o usuário, não será vista nenhuma diferença, pela forma de que os dados serão ordenados de qualquer maneira. A única diferença que pode ser vista entre os dois algoritmos é no tempo de execução, e mesmo assim uma diferença muito ligeira, ainda mais com a pouco quantidade de dados utilizados.

Os procedimentos iniciais, (criação do vetor, alocação dos dados do .txt no vetor) são realizados da mesma forma que no programa anterior. Após feito esses processos é chamado o método QuickSort,

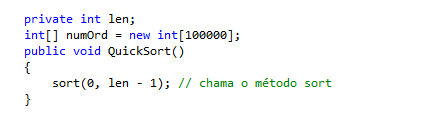


Ao chamar esse método, ele obviamente sai do método principal (main) e inicia o processo de ordenação de dados com o algoritmo QuickSort.

O método que será apresentado agora é básico e simples, porém é essencial na ordenação dos dados. Ele tem uma função parecida com a do método MergeSort do programa anterior, apenas enviar o valor dos índices que deverão ser ordenados através de variáveis nomeadas como esquerda e direita.

Uma diferença no desenvolvimento dessa aplicação, é que não tem necessidade da utilização de um vetor auxiliar, ele vai substituindo os valores no próprio vetor numOrd. Isso pode ser uma grande vantagem no tempo de execução deste programa em comparação com o algoritmo MergeSort.

Segue a tela do método QuickSort:

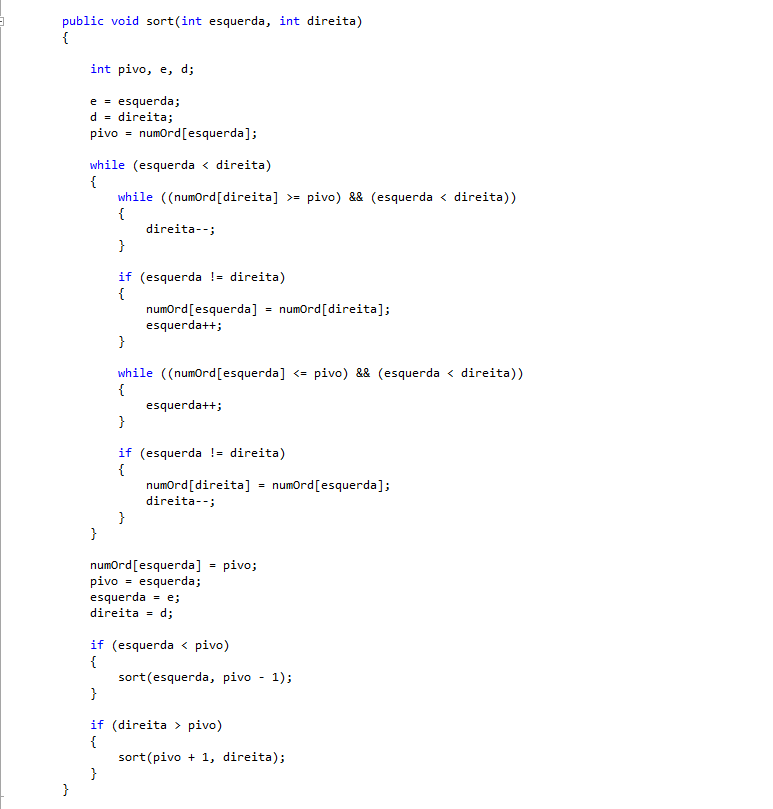


Nota-se que o vetor numOrd não é criado dentro do método main, exatamente pelo fato de que todos os métodos main e sort irão utilizar o mesmo vetor. Uma vez que uma variável é criada dentro de um método, ela pode ser utilizada apenas nesse método.

Inicia-se agora a ordenação em si, a parte do código que faz a troca de posicionamento entre dos índices dos vetores. Como se sabe e já foi explicado, o método de ordenação QuickSort utiliza uma posição do vetor que é usada como pivô. No programa, foi criada uma variável para receber o valor deste. Todos os valores menores que essa variável serão visto como valores pequenos, e os maiores como valores grandes. Entender isso é o princípio para começar a desenvolver um algoritmo QuickSort.

A variável pivot, onde será alocado os valores correspondentes ao pivô, inicia-se com o primeiro valor do .txt, correspondente ao numOrd[0].

Segue a tela do método sort:

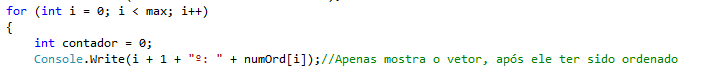


Percebe-se que no final do método é chamado novamente o mesmo método, aplicando a recursividade (chamada da própria função).

Ao executar esses procedimentos, os dados que estão armazenados no .txt estão devidamente ordenados no vetor. Agora está na hora de mostrar para o usuário que os dados estão ordenados e que o programa está funcionando corretamente.

Primeiramente vamos executar uma estrutura de repetição *for*, e cada execução da estrutura representa a um valor que será mostrado para o usuário.

Segue o print da tela mostrando apenas o número:

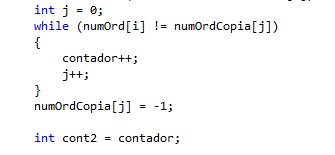


É criada uma variável nomeada como contador, essa terá uma participação essencial para ligar o número com o mês e o ano correspondente, seguindo os dados informados pelo Imazon.

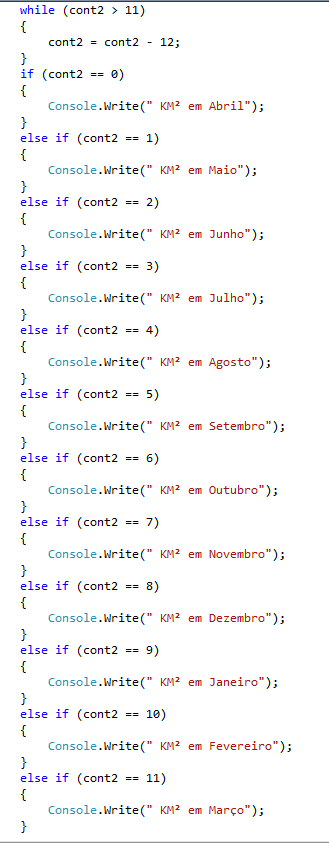
Para auxiliar neste processo de ligação, foi criado também outro vetor que deverá ser inicialmente igual ao vetor numOrd. Este recebe o nome de numOrdCopia. Porém, a ordenação acontece apenas no vetor numOrd, e o vetor numOrdCopia fica com a mesma ordem do arquivo .txt. Assim é possível ver em qual posição o valor ordenado estava antes de ter sido ordenado. Portanto, com a ajuda de uma variável nomeada contador, o numOrdCopia “percorre” o vetor ordenado até encontrar o valor idêntico, assim encontrando a posição que ficará armazenada na variável contador.

Depois de feito isso, este valor que foi encontrado no vetor numOrdCopia será substituído por -1, para que não ocorra problemas de duplicidade. Vamos começar a explicar esse processo com um exemplo:

O valor 170 corresponde à posição 29 no vetor desordenado (numOrdCopia). Quando ele é ordenado ele vai para a posição 52 do vetor numOrd. Através da seguinte linha de código ele irá percorrer o vetor desordenado até encontrar o 170, e no final da execução, a variável contador irá receber +1. Portanto, neste exemplo a variável contador ficará com o valor de 29 (posição do 170 no vetor numOrdCopia).



Como os dados estão ordenados por mês, a contagem de mês começa por abril. Portanto 0 representa abril, 1 representa maio, 2 representa junho, e assim por diante... E foi essa forma que foi usada para mostrar ao usuário qual mês que é correspondente à aquele dado. A variável contador recebe o valor de até no máximo o número de dados do arquivo. Enquanto esse número não for menor que 11, ele vai diminuindo 12, ou seja, está mantendo o mês correspondente. É importante lembrar que o que vale é o vetor desordenado, pois este está organizado por meses e anos. Portanto se no vetor desordenado o valor está na posição 29, este cálculo é feito: 29 - 12 = 17. 17 > 11 = V. 17 – 12 = 5. 5 > 11 = F. Mês correspondente é: setembro. Veja a linha de código para entender melhor:



Após descoberto o mês, agora precisamos descobrir qual é o ano. Sabemos que os dados informados são a parir de abril de 2008, portanto o menor ano será 2008. A variável contador ainda será usada. Sabemos também que os 8 primeiros índices do vetor correspondem ao ano de 2008, e a partir disso, a cada 12 índices é um ano. Com base nessas informações foi criada uma variável ano, com valor inicial = 2008. Enquanto ela contador for maior que 8, ele acrescenta 1 à variável ano, e decrescente 12 à variável contador, dessa forma vamos chegar ao ano correspondente. Vamos seguir o exemplo anterior:

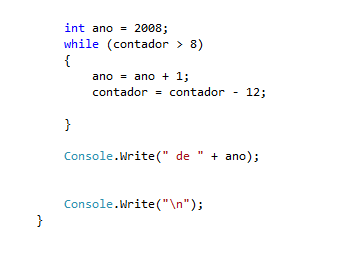
Ano = 2008. Contador = 29. Contador > 8 = V. Contador = contador – 12. Contador = 17. Ano = ano + 1. Ano = 2009. Contador > 8 = V. Contador = contador – 12. Contador = 5. Ano = ano + 1. Ano = 2010. Contador > 8 = F.

Ano = 2010.

Portanto, o valor 170 corresponde ao mês de setembro e ao ano de 2010. Vamos conferir no site da Imazon:

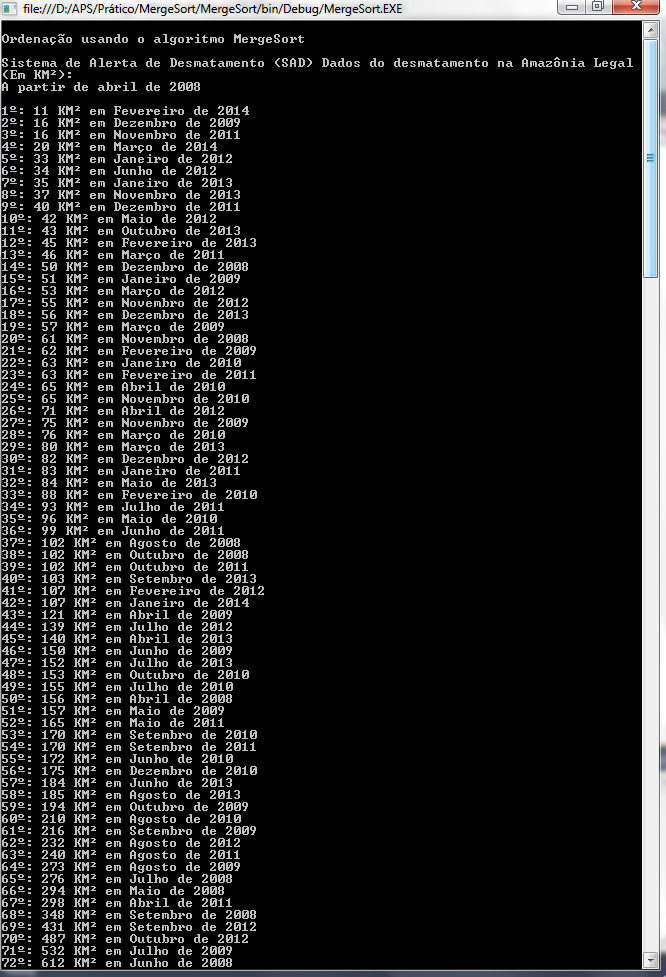


Foi comprovado que a lógica apresentada é correta. Agora vamos mostrar a linha de código que calcula o ano e em seguida o print da tela do programa em execução funcionando corretamente, tanto o MergeSort quanto o QuickSort. Pode ser verificado que todos os valores batem com os meses e anos informados no site Imazon. E um detalhe que o programa também oferece é que, quando necessário informar mais algum valor, basta adicioná-lo uma linha abaixo do último valor no dados.txt. Ao fazer isso o programa já irá ordená-lo na próxima execução, e também calculará o mês e o ano correspondente (Lembrando que os dados informados são de abril de 2008 até março de 2014, sendo assim se adicionado um valor em baixo, ele irá calcular como valor referente à abril de 2014 e assim por diante).



Os programas foram desenvolvidos na plataforma Visual Studio, utilizando a linguagem de programação Orientada a Objetos C#.

Segue a tela do programa MergeSort em execução:



Segue a tela do programa QuickSort em execução:

