FACULDADES INTEGRADAS DE CARATINGA

FACULDADE DE CIÊNCIA DA COMPUTAÇÃO

PROGRAMAÇÃO DE HORÁRIO ESCOLAR: O CASO DOS CURSOS DE EXATAS DA FIC

WENDEL FORMAGGINE

CARATINGA
2013

Wendel Formaggine

PROGRAMAÇÃO DE HORÁRIO ESCOLAR: O CASO DOS CURSOS DE EXATAS DA FIC

Monografia apresentada ao Curso de Ciência da Computação das Faculdades Integradas de Caratinga como requisito parcial para obtenção do título de Bacharel em Ciência da Computação orientado pelo Professor Wanderson Miranda Nascimento.

Caratinga
2013

Wendel Formaggine

PROGRAMAÇÃO DE HORÁRIO ESCOLAR: O CASO DOS CURSOS DE EXATAS DA FIC

Monografia submetida à Comissão examinadora designada pelo Curso de Graduação em Ciência da Computação como requisito para obtenção do grau de Bacharel.

_____

Prof. Wanderson Miranda Nascimento
Faculdades Integradas de Caratinga

_____

_____

Caratinga, / /

AGRADECIMENTOS

Agradeço primeiramente a Deus pela vida que me concedeu, pela força e coragem durante essa longa trajetória.

Aos meus pais Mauro e Penha, pelo apoio, pelos ensinamentos e pelos investimentos em meus estudos. Minha esposa Glenda pelo incentivo, companheirismo e paciência. Também a minha tia Helena e minha prima Karina que me acolheram durante os anos de estudo.

Ao professor Wanderson pela orientação, disponibilidade e atenção. Ao professor Paulinho por ter me auxiliado bastante no início do desenvolvimento deste trabalho e a professora Fabrícia, sempre muito prestativa.

A instituição e principalmente as funcionárias da secretaria que foram bastante atenciosas me fornecendo informações que foram cruciais para a conclusão deste trabalho.

RESUMO

O Problema da programação da grade horária escolar é comum em qualquer instituição de ensino. A dificuldade de se obter o resultado tem estimulado o estudo de métodos que possam tornar esta tarefa menos árdua. Há casos em que a tabela de horários tem que ser adaptada à instituição devido à existência de características específicas encontradas na mesma. Este tipo de problema tem sido comumente tratado com técnicas heurísticas tais como GRASP, em que uma solução inicial é formada e tratada até que se chegue num resultado satisfatório. Este trabalho consiste em utilizar uma adaptação do método GRASP na construção da tabela de horário, de forma que atenda a característica existente nas Faculdades Integradas de Caratinga que consiste na junção de várias turmas numa mesma sala e mesmo horário, utilizando dados reais colhidos na instituição. A ideia é a elaboração de um algoritmo que produza resultados que serão comparados na questão qualidade e desempenho, com o método utilizado hoje.

Palavras chaves: Problema da programação da grade horária, Heurística, metaheurística GRASP.

LISTA DE FIGURAS

LISTA DE TABELAS

SUMÁRIO

1 INTRODUÇÃO

Todo início de semestre letivo as instituições de ensino enfrentam um grande problema de otimização: a construção da tabela de horário das aulas. Este problema é considerado complexo devido à vários fatores conflitantes além das necessidades específicas de cada instituição.

Na FIC (Faculdades Integradas de Caratinga) não é diferente. São gastos horas de combinações em rascunhos até que se chegue numa tabela que atenda todos os requisitos necessários tais como disponibilidade dos professores, matérias que serão lecionadas no semestre, turmas que serão juntas numa mesma sala, entre outros.

O trabalho trata os cursos da área de exatas da instituição, sendo eles Ciência da computação, engenharia civil e engenharia elétrica. Juntos eles possuem 116 matérias distribuídas em 21 turmas.

Tendo visto a complexidade do problema, algumas técnicas heurísticas vêm sendo estudadas a fim de solucionar o caso, dentre as quais destacam-se a Busca Tabu (GLOVER E LAGUNA, 1997), a Programação Genética (Ueda et. al. 2001 apud OLIVEIRA 2006) e a metaheurística GRASP (Festa e Resende 2002), que será vista com mais detalhe neste trabalho.

Com o problema de otimização em questão pode-se formular a seguinte pergunta: Como desenvolver a tabela de horário da FIC do curso de exatas de uma forma mais rápida e automatizada utilizando algumas das técnicas existentes adaptando às características da instituição?

A partir do estudo do caso, o desenvolvimento de uma aplicação que se consiga implementar as regras utilizadas para a construção da tabela de horário da instituição é um passo importante para alcançar esse feito.

O objetivo é que este trabalho de pesquisa contribua para que a instituição poupe tempo na construção da tabela de horários além de auxiliar na organização do início do semestre letivo para alunos e professores. Espera-se ainda que o sistema apresente várias opções de resultados para que seja escolhida a grade horária mais adequada.

No próximo capítulo será feita uma abordagem sobre a definição de algoritmo, sobre a geração de grade horária, o que diz a literatura à respeito das soluções desenvolvidas e os resultados obtidos e de que forma os métodos existentes podem contribuir para o tratamento do problema.

No capítulo 3 será abordado a metodologia utilizada. O método escolhido, a linguagem de programação, a forma como os dados são entrados no sistema, como os dados são processados, as regras utilizadas, como o resultado é apresentado.

O capítulo 4 será responsável pelos resultados. Será apresentado o que foi produzido: as tabelas geradas e um comparativo destacando a qualidade e o tempo gasto para geração das tabelas de horário.

Por fim teremos as considerações finais e as referências.

2 REFERENCIAL TEÓRICO

2.1 INTRODUÇÃO A ALGORITMOS

O conceito do que é um algoritmo pode ser interpretado de forma sucinta como passos que devem ser seguidos para solucionar determinado problema. Mas a prática por traz desse conceito em muitas vezes é de elevada dificuldade tendo em vista que se trata de um problema não polinomial.

A sociedade da computação está sempre em busca de melhores soluções, seja levando em consideração a qualidade da solução, seja levando em consideração um tempo interessante de execução e a memória utilizada.

> O projeto de algoritmos é fortemente influenciado pelo estudo de seus comportamentos. Depois que um problema é analisado e decisões de projeto são finalizadas, o algoritmo tem que ser implementado em um computador onde os aspectos de tempo de execução e espaço ocupado são considerações importantes. (ZIVIANI, 2002, P.3)

Assim, percebe-se que ao decidir por um algoritmo deve-se analisar tanto o tempo que demora para emitir uma resposta quanto a quantidade de memória necessária para tal fato.

Ziviani(2002) ainda informa o que deve ser levado em consideração para medir o custo de utilização de um algoritmo e destaca o fato que utilizar-se da medida do tempo de execução em um computador parece, em muitos momentos, injusta, já que existem inúmeros modelos, marcas, qualidade, isto é, *hardwares* distintos e outros fatores consideráveis para cada máquina, como por exemplo, o próprio compilador. Ainda tem o fato da implementação de um algoritmo ser algo totalmente pessoal, influenciando, às vezes, de maneira insatisfatória no resultado obtido. Nesse caso, o autor indica como melhor método medir o custo de um algoritmo através de um modelo matemático.

Conforme Cormen(2002, p 18) “O tempo de execução do algoritmo é a soma dos tempos de execução para cada instrução executada”. Para exemplificar essa afirmação de como encontrar uma formulação matemática consideremos o algoritmo Ordenação por Seleção da FIG. 1. Assim, para encontrar essa formulação final deve-se considerar cada linha do algoritmo. Por exemplo, de forma reduzida, o laço de repetição da linha 1 será executado $n-1$ vezes, já o laço de repetição da linha 4 será executado

$n-i$ vezes. E somando-se os tempos de todas as linhas o que obtêm-se um tempo, em média, $T(n^2)$, portanto ela é uma função quadrática de n. O pior caso de um algoritmo é "*o* tempo de execução mais longo para qualquer entrada de tamanho *n"* (CORMEN, 2002, p. 19).

```
1   para (i = 1; i<= n-1; i = i+1) faça
2   Início
3         min = i;
4         para (j = i+1; j<= n; j = j+1)
faça
5         Início
6            se A[j].Chave < A[min].Chave
então
7                  min = j;
8         Fim
9         x = A[min];
10        A[min] = A[i];
11        A[i] = x;
12   Fim
```

FIGURA 1 - Algoritmo de ordenação por seleção

FONTE: ZIVIANI(2002)

Diz-se que a medição do custo de um algoritmo, como feito anteriormente, determina a complexidade do algoritmo. Assim, pode-se concluir que "a complexidade de um algoritmo reflete o esforço computacional requerido para executá-lo" (TOSCANI, 2002, p. 7.).

Deve-se notar o fato de que um problema pode ter mais de uma solução. (TOSCANI, 2002) indica a comparação das complexidades dos algoritmos para que seja feita a escolha da solução mais adequada. Considere, por exemplo, dois algoritmos X e Y e suas respectivas complexidades $T_X(n) = 2n$ e $T_Y(n) = n^2$. Se os dois resolvem o mesmo problema mas o algoritmo X possui um tempo polinomial, já o algoritmo Y possui um tempo quadrática, a escolha mais viável seria o algoritmo X. Está aí a Otimização Combinatória, área da computação que busca encontrar, dentre todas as soluções possíveis, a solução cujo custo seja o menor possível.

Segundo o Operations Research Center do MIT(MIT 2012), a Pesquisa Operacional consiste em unir matemática e computação objetivando auxiliar na tomada de decisões da vida real. Através de modelos matemáticos, estatísticos e de algoritmos

busca-se minimizar ou maximizar as chamadas funções objetivo. Isto é, dada uma formulação matemática que descreve determinado problema computacional, busca-se reduzir, por exemplo, os riscos ou custos ou, ainda, aumentar por exemplo, o lucro ou o rendimento agrícola.

Como bem dito por (NEMHAUSER e WOLSEY, 1988.) apud (ANDRADE, 2005) uma boa formulação do problema é importantíssima para alcançar uma boa solução.

Segundo (TOSCANI, 2002, p. 219) “o mais importante questionamento sobre um problema é a sua computabilidade”. Neste caso, ser computável significa dizer que o problema poderá ser resolvido por um algoritmo. Mas ser computável não é tudo, também é necessário saber se o algoritmo que resolve o problema é eficiente. Neste caso, ser eficiente significa dizer que o algoritmo possui complexidade polinomial.

Os problemas são classificados conforme sua dificuldade de resolução e (TOSCANI, 2002) descreve bem sobre os algoritmos e seus problemas. Os algoritmos para os quais existe algoritmo de ordem polinomial pertencem a classe P (polinomiais). Os problemas para os quais existe algoritmo de ordem polinomial que verifica se tem a resposta Sim ou Não, isto é, têm algoritmo não-determinístico pertencem a classe NP (não-determinístico polinomial). Um algoritmo é dito não eficiente se os algoritmos que o resolvem são não-polinomiais. Um problema é dito tratável “se o limite superior de complexidade é polinomial”. Já um problema intratável é aquele que possui “o limite inferior de complexidade exponencial”. Os chamados problemas NP-Completos são os algoritmos que não se tem certeza que possuem ou não algum algoritmo polinomial que o resolva. Já a classe de problemas NP difícil é uma classe de problemas que possui problemas mais difíceis do que estão em NP-Completo, e não se conhecem algoritmos não-determinísticos de tempo polinomial para eles, e portanto não se sabe se pertencem ou não a NP. A FIG. 2 mostra bem a relação entre as classes de problemas.

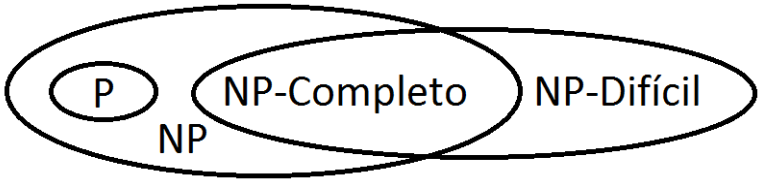

FIGURA 2 - Classes de Problemas

Fonte: (TOSCANI, p. 252)

A resolução de problemas NP-Difícil por técnicas de programação matemática, é praticamente inviável devido ao elevado tempo de processamento requerido. Sendo assim problemas desta classe são normalmente abordados por técnicas heurísticas. Schaefer (1990, apud COSTA, 2003).

2.2 TÉCNICAS HEURÍSTICAS

Heurística (do grego *ευρίσκω, heurísko*) segundo Silva (2005, pag. 8) significa "descobrir" ou "achar". Dessa forma dizemos que este método é capaz de achar soluções nem sempre a melhor, mas normalmente válidas conforme FOULDS (1984, apud BECCENERI, 2008, p. 4), que consideram também o termo como associado a um conhecimento circunstancial, não verificável, nem matematicamente verificável.

Silva (2005, pag. 9) aponta heurísticas classificadas em duas categorias:

- Heurísticas construtivas: consiste em construir uma solução possível completa de um determinado problema adicionando elementos um por vez atendendo ao máximo todos os requisitos.
- Heurísticas de refinamento: através de uma solução possível, consiste em buscar a solução possível completa por meio de uma sequência de mudanças dos elementos.

Alvarenga (2004, apud OLIVEIRA, 2006) ainda cita duas características para se garantir bons resultados na utilização de heurísticas:

- Qualidade da solução: em tempo inferior ao tempo gasto dos métodos exatos disponíveis, encontrar soluções próximas do ótimo, justificando assim sua utilização;
- Robustez: não deve haver demasiada variação na qualidade da solução na aplicação da heurística para diferentes instâncias de um mesmo problema para o qual a mesma foi projetada ou mesmo quando a heurística é aplicada várias vezes para a mesma instância.

Cormen et. al. (2001) cita métodos com características de boas soluções semelhantes, denominados métodos de aproximação. Com esses métodos é possível encontrar soluções para problemas exponenciais onde os métodos exatos não se aplicam computacionalmente com uma boa distância da solução ótima.

A princípio pode-se dizer que os métodos heurísticos e os métodos aproximados parecem iguais, mas não são. Semelhantemente pode-se dizer que buscam de maneira viável, apresentar soluções próximas àquela ótima.

A diferença entre eles é que um método aproximativo busca a cada iteração aproximar do ótimo, enquanto os métodos heurísticos não têm essa mesma garantia de melhora. Em contrapartida as heurísticas convergem em tempo extremamente mais rápido comparadas aos métodos aproximativos. Este fato fez com que esse método tivesse uma grande disseminação nos últimos anos. Andrade (2004).

Dentre os métodos heurísticos conhecidos há a metaheurística. O termo deriva de duas palavras, sendo heurística, a palavra previamente estudada, e meta que significa após, indicando a grosso modo um "logo após à descoberta". Segundo BECCENERI(2008), a técnica pode ser vista como uma ferramenta algorítmica geral que pode ser aplicada a diferentes problemas de otimização. Uma estratégia de busca não especificando o problema, que procura explorar o espaço das soluções viáveis do problema. Ele ainda resume como mecanismos de alto nível para explorar espaços de busca, cada uma usando um determinado tipo de estratégia.

Algumas metaheurísticas bem conhecidas são: *Simulated annealing*, Algoritmos Genéticos e GRASP*(Greedy Randomized Adaptive Search Procedure*). Infobrasil (2009).

2.3 GRASP (*Greedy Randomized Adaptive Search Procedure*)

O Procedimento de GRASP (*Greedy Randomized Adaptive Search Procedure*) ou Busca Adaptativa Gulosa e Randomizada é um método iterativo proposto por Feo e Resende, que tem sido aplicado à uma grande variedade de problemas de otimização combinatória, variando desde agendamento e rotas até desenho e balanceamento de turbinas. (FESTA E RESENDE 2002)

O método consiste numa metaheurística com a função de localizar uma solução válida através de um procedimento guloso-aleatório e logo depois localizar uma condição melhor do que a primeira encontrada através de uma busca local. (MOURA, SCARAFICCI, SILVEIRA e SANTOS, 2004)

Um pseudo-código do GRASP pode ser representado conforme FIG. 3 abaixo:

```
proc grasp (MAX_ITER)
     Carrega_Instancia_Entrada()
     para k=1 ate MAX_ITER faca
          Solucao = constroiSolucao ()
          Solucao = buscaLocal ()
          atualizaSolucao (Solucao, Melhor_Solucao)
     fim para
     retorna (Melhor_Solucao)
fim grasp
```

FIGURA 3 - Pseudocódigo do GRASP

Fonte: Silva, Ochi e Martins(2006)

Aplicado na construção de grade horária escolar o método GRASP gera uma tabela inicial e depois efetua mudanças que melhore a condição inicial de acordo com regras utilizadas naquela instituição.

A cada iteração os elementos serão analisados e aqueles considerados mais críticos através de uma avaliação determinada (por exemplo, um professor com pouca disponibilidade é uma condição crítica) irão compor uma lista denominada LRC (Lista restrita de candidatos). Será escolhido aleatoriamente um elemento desta lista para ser usado no preenchimento da tabela de horário. Ao fim de cada iteração os elementos serão reavaliados para se saber a criticidade, ou importância dos mesmos para compor a grade horária.

2.4 PROBLEMAS DE GERAÇÃO DE GRADE HORÁRIA

O problema de quadro de horários trata-se de um problema de otimização combinatória considerado NP-difícil (Even. et al,1976 apud FERREIRA e GLAZAR 2005), em que consiste em estabelecer uma tabela de aulas conforme tabela 1, correspondente às matérias necessárias de cada semestre letivo, devendo respeitar um conjunto de restrições, segundo Schaefer (1990, apud COSTA, 2003).

A solução manual deste problema é uma tarefa árdua e normalmente requer vários dias de trabalho. Além disso, devem-se levar em conta as características particulares de cada instituição.

TABELA 1 - horário do curso de ciência da computação da FIC – 2º semestre de 2013

Dia	Horário	Sala	Computação2	Sala	Computação4	Sala	Computação6	Sala	Computação8
Segunda	19h00	D501	FAdm - Fabricia	D105	PAA - Wanderson	D205	PAA - Wanderson	D205	PO - Hebert
	19h50	D501	FAdm - Fabricia	D105	PAA - Wanderson	D205	PAA - Wanderson	D205	PO - Hebert
	20h40		INTERVALO		INTERVALO		INTERVALO		INTERVALO
	20h55	D501	ED - Glauber	D105	Redes1 - Hebert	D205	ESof2 - Fabricia	D201	Comp - Jonilson
	21h45	D501	ED - Glauber	D105	Redes1 - Hebert	D205	ESof2 - Fabricia	D201	Comp - Jonilson
Terça	19h00	D403	AC1 - Jonilson	D201	POO - Glauber	D105	MatCom - Hebert	D205	Opt4 - Fabricia
	19h50	D403	AC1 - Jonilson	D201	POO - Glauber	D105	MatCom - Hebert	D205	Opt4 - Fabricia
	20h40		INTERVALO		INTERVALO		INTERVALO		INTERVALO
	20h55	D501	ED - Glauber	D105	PAA - Wanderson	D205	PAA - Wanderson	D205	PO - Hebert
	21h45	D501	ED - Glauber	D105	PAA - Wanderson	D205	PAA - Wanderson	D205	PO - Hebert
Quarta	19h00	D501	FAdm - Fabricia	D205	POO - Glauber	D105	MatCom - Hebert	D201	
	19h50	D501	FAdm - Fabricia	D205	POO - Glauber	D105	MatCom - Hebert	D201	
	20h40		INTERVALO		INTERVALO		INTERVALO		INTERVALO
	20h55	D501	LabED- Glauber	LAB101	Redes1 - Hebert	D205	Comb- Vagner	D201	Comp - Jonilson
	21h45		LabED- Glauber	LAB101	Redes1 - Hebert	D205	Comb- Vagner	D201	Comp - Jonilson
Quinta	19h00		AC1 - Jonilson	D105	GAAL - Rogerio	D205	TC - Wanderson	D201	Opt4 - Fabricia
	19h50		AC1 - Jonilson	D105	GAAL - Rogerio	D205	TC - Wanderson	D201	Opt4 - Fabricia
	20h40		INTERVALO		INTERVALO		INTERVALO		INTERVALO
	20h55		Calc2 - Ricardo	D105	LabP - Glauber	LAB101			TCC2 - Fabricia
	21h45		Calc2 - Ricardo	D105	LabP - Glauber	LAB101			TCC2 - Fabricia
Sexta	19h00		LogM - Fabricia	D105	GAAL - Rogerio	D205	TC - Wanderson	D201	
	19h50		LogM - Fabricia	D105	GAAL - Rogerio	D205	TC - Wanderson	D201	
	20h40		INTERVALO		INTERVALO		INTERVALO		INTERVALO
	20h55		Calc2 - Ricardo	D105			ESof2 - Fabricia	D201	
	21h45		Calc2 - Ricardo	D105			ESof2 - Fabricia	D201	

Fonte: Coordenadoria Geral da FIC

Segundo LOBO (2005), devido ao grau de complexidade da implementação da solução, a comunidade acadêmica tem dado uma atenção especial ao problema de quadro de horários desde a década de 60.

Bianualmente acontece uma conferência internacional chamado PATAT – *Practice and Theory of Automated* que contribui como um fórum para pesquisadores e praticantes da construção de tabela de horários trocarem ideias. Os assuntos abordados são de áreas diversas tais como saúde, educação, esporte e transporte. Eles têm a oportunidade de mostrarem trabalhos e experiências com o objetivo de desenvolvimento de soluções eficientes e práticas. (PATAT 2012).

LOBO (2005) diz ainda que o problema da geração de grade horária pode ser tratado como um problema de busca, mas ao mesmo tempo de otimização no qual deve-se atender a todas as restrições necessárias e maximizar o atendimento das restrições desejáveis.

2.4.1 Variantes do problema

Num quadro de horários pode-se dizer que há muitas variantes que interferem em sua confecção. Elas se adequam particularmente a cada instituição. Lobo (2005 p. 4) cita algumas delas:

- Salas específicas: Determinadas matérias possuem didática de forma que precisam de salas específicas para serem ministradas. Como por exemplo aulas de programação, podem precisar de laboratórios com computadores para a prática, aulas de mecânica, podem precisar de laboratório com equipamentos e/ou máquinas, apresentações de trabalhos podem precisar de sala com recursos tais como projetor, microfone entre outros;
- Aulas com mais de uma turma na mesma sala: às vezes ocorre a situação de juntarem mais de uma turma numa mesma sala para que seja ministrada pelo mesmo professor, de matérias que coincidem entre cursos diferentes ou às vezes até do mesmo curso, ex: Optativa, Português, TCC entre outras;
- Dois professores numa mesma sala: pode ocorrer também a necessidade de ter dois professores ministrando a mesma matéria, no caso por exemplo de cada um ter um domínio maior que o outro de assuntos diferentes;
- Pouca disponibilidade do professor: o professor muitas vezes tem poucos horários disponíveis para ministrar aulas por motivos diversos: Leciona em outro curso, estuda, ou trabalhe até em outra faculdade;

2.4.2 O problema da geração da grade horária da FIC

Semelhantes à outras instituições de ensino a FIC encontra dificuldades para geração da grade horária por diversos fatores que quando somados aumentam exponencialmente o grau de dificuldade dessa tarefa.

Além das variantes citadas acima, a necessidade da junção de turmas é outro fator considerável, onde duas ou mais turmas concentram-se numa mesma sala a fim de proporcionar a redução de custos para a faculdade.

Esses problemas faz com que a geração da grade horária oficial ocorra em muito das vezes após a inicialização das aulas.

Deve-se também considerar que o profissional responsável pela geração desse quadro de horário tem outras atribuições e atualmente não utiliza nenhum software específico para geração total ou ao menos auxiliar nesta atividade.

Este trabalho refere-se ao tratamento do problema utilizando os dados dos cursos da área de exatas: ciência da computação, engenharia civil e engenharia elétrica, embora outros cursos também possam ser beneficiados com ele.

2.4.3 Soluções para o problema da geração da grade horária

Conforme MOURA et al., (2004) uma das formas utilizadas para desenvolvimento da grade horária é a resolução manual deste problema através de tentativa e erro, porém o método consome muito tempo e as soluções produzidas são de baixa qualidade. A ideia é tratar os casos mais restritivos primeiro, como por exemplo professores com pouca disponibilidade, e em seguida os demais casos.

LOBO(2005) destacou que com o crescimento da complexidade do problema surgiu a necessidade do desenvolvimento de técnicas de automatização da geração do quadro de horários, tais como utilização de algoritmos genéticos, IA (inteligência artificial) e técnicas heurísticas e metaheurísticas.

2.4.4 Formulação do Problema

LOBO (2005) descreve formalmente o problema da geração da grade horária da seguinte maneira: Considere um conjunto com *t* turmas (t_1,t_2,t_3...,t_t), *p* professores (p_1,p_2,p_3...p_p), *h* períodos (*h1*,*h2*,*h3*...*hh*), e ainda uma matriz de inteiros não negativos $R_{txp,}$ em que r_{ij} é a carga horária do professor *j* com a turma *i*.

O problema consiste na associação de aulas e períodos. Não deve deixar acontecer que um professor dê aula para mais de uma matéria num mesmo horário, e que cada matéria tenha apenas um único professor.

Werra (1971, apud LOBO 2005) apresenta uma formulação matemática para o problema:

Encontrar x_{ijk} (i=1...t; j=1...p; k=1...h), de forma que:

$x_{ijk} = r_{ij}$ (i=1...t; j=1...p)

$x_{ijk} \leq 1$ (i=1...t; k=1...h)

$x_{ijk} \leq 1$ (j=1...p; k=1...h)

$x_{ijk} = 0$ ou $x_{ijk} = 1$(i=1...t;j=1...p;k=1...h)

No caso, se o professor tem aula com a turma i no período k, a variável X_{ijk} terá valor = 1, caso contrário seu valor será = 0.

Pode-se citar as seguintes observações:

- A primeira restrição consiste em certificar que o professor *j* lecione a carga horária correta para a turma *i;*
- A segunda restrição consiste em certificar que não irá haver sobreposição de turmas;
- A terceira restrição certifica que não haja sobreposição de professor.

3 METODOLOGIA

Para alcançar o objetivo deste estudo, deve-se relembrar que trata-se do desenvolvimento de uma aplicação utilizando os conhecimentos adquiridos com o estudo do caso e suas possíveis soluções, adaptando às características da FIC fazendo um comparativo dos seus resultados e atestando seu desempenho com diferentes parâmetros configurados.

Entender as dificuldades da construção da grade horária escolar e principalmente saber quais são e procurar tratar as necessidades específicas da FIC são pontos fundamentais para a escolha do algoritmo que será utilizado e a forma de como ele será aplicado.

Os passos fundamentais para se chegar à esse objetivo consistem em:

- Levantamento dos dados de pesquisa;
- Escolha do algoritmo que será utilizado para o tratamento do problema;
- Implementação do algoritmo adaptando o método escolhido de forma que atenda a exigência da instituição;

3.1 OS DADOS DE PESQUISA

O trabalho aqui proposto tem por base os dados da FIC, escolhida principalmente pela facilidade da obtenção das informações nas quais serão fundamentais para a geração da grade horária. Em consequência disso espera-se contribuir com a instituição para sanar as dificuldades que a mesma encontra para a realização da tarefa, necessária a cada início de semestre letivo.

Foram selecionados os cursos de Ciência da computação, engenharia elétrica e engenharia civil. Desses, foram levantados junto à secretaria dos cursos os dados do segundo semestre do ano de 2013 e as seguintes informações foram obtidas:

- São 4 horários/aula por dia, de segunda a sexta;
- Há 21 turmas nos cursos envolvidos, detalhados abaixo:
    - Curso de Ciência da computação: Períodos 2º, 4º, 6º e 8º.
    - Curso de Engenharia Civil: Períodos 1º N, 2º A, 2º B, 3º N, 4º A, 4º B, 5º N, 6º A, 6º B, 8º N, 9º N e 10º N.
    - Curso de Engenharia Elétrica: Períodos 2º, 4º, 6º, 8º e 10º.

- São 35 professores envolvidos nas matérias;
- A quantidade de aula de cada matéria varia, podendo ser 2 ou 4 aulas;
- São 116 matérias no total de todos os períodos.

3.2 A ESCOLHA DO ALGORITMO

Uma característica existente na FIC foi fundamental para a escolha do algoritmo utilizado. Trata-se das matérias que alguns períodos possuem em comum, chamadas aqui de matérias combo podendo ser de períodos e até de cursos diferentes. Na prática são disciplinas ministradas com duas ou mais turmas presentes numa mesma sala, no mesmo dia e horário.

O tratamento dessa complexidade combinatória que envolve disponibilidade do professor e disponibilidade para a alocação da disciplina no mesmo dia e hora em todas as tabelas de horário dos períodos envolvidos, ficou a cargo da metaheurística GRASP.

3.3 IMPLEMENTAÇÃO DO ALGORITMO

A implementação do GRASP foi feita em linguagem C utilizando o compilador da BORLAND BCC32 versão 5.5. Foram utilizadas algumas bibliotecas para uso de algumas funções tais como geração de números aleatórios, tratamento de strings e manipulação de arquivos para escrita e leitura.

O GRASP pode ser encontrado na literatura em muitas variações. A definição normalmente depende do problema em que será aplicado. Na versão clássica o GRASP cria sua lista restrita de candidatos (LRC) através de critérios de avaliação que meça o benefício de sua inserção. A cada item utilizado da lista é feita uma nova avaliação reclassificando-os de acordo com o último item escolhido. Porém muito das vezes essa avaliação é imprecisa e acaba deixando itens críticos para trás.

Neste ponto o algoritmo foi adaptado para que ele pudesse tratar melhor o problema da FIC. No caso a lista passa a ser fixa, composta pelas matérias combo, passada sempre no início do processamento dos dados não sendo mais criada através de função de avaliação. A escolha dos itens é feita como antes, através de sorteio e isso é feito no início do processo de geração das tabelas de horário, quando a complexidade é menor.

Um outro detalhe é que o preenchimento que antes era feito por apenas um item escolhido aleatoriamente passa a ser feito por dois e o preenchimento agora deve atender um conjunto de regras. Isso faz com que a tabela gerada inicial tenda à uma qualidade superior se comparado ao processo de geração do algoritmo GRASP original antes da busca local.

Na FIG. 4 é possível ver o código GRASP original e onde entrou as modificações.

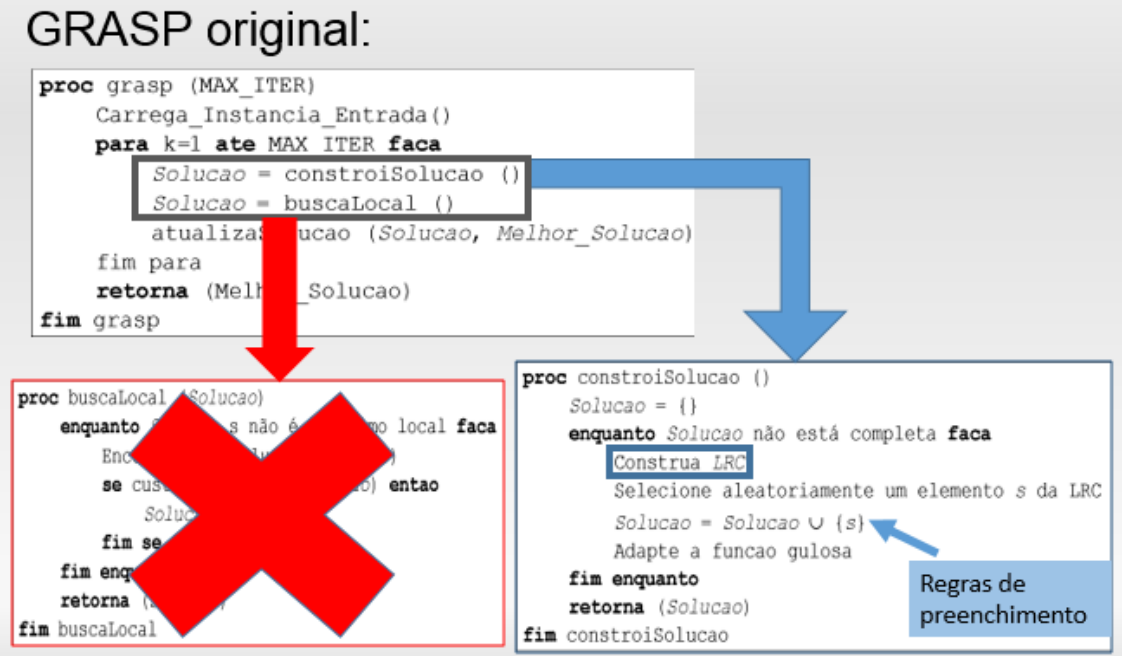

FIGURA 4 - Código GRASP original

3.3.1 Regras de preenchimento

Para garantir a qualidade da tabela gerada, o algoritmo possui algumas regras que irão “modelar” a tabela final. Para o entendimento das regras é mostrado na tabela 2 as possíveis posições para o preenchimento das matérias, enumeradas de 1 a 20.

TABELA 2 - Posições possíveis de preenchimento

Segunda-feira	Terça-feira	Quarta-feira	Quinta-feira	Sexta-feira
1	5	9	13	17
2	6	10	14	18
Intervalo	Intervalo	Intervalo	Intervalo	Intervalo
3	7	11	15	19
4	8	12	16	20

Para alocar matérias com 4 aulas, o preenchimento ocorrerá de 2 em 2 horários consecutivos como visto na tabela 3. EX: 1,2 e 7,8.

TABELA 3 - Posições possíveis de preenchimento

Segunda-feira	Terça-feira	Quarta-feira	Quinta-feira	Sexta-feira
CÁLCULO	5	9	13	17
CÁLCULO	6	10	14	18
Intervalo	Intervalo	Intervalo	Intervalo	Intervalo
3	**CÁLCULO**	11	15	19
4	**CÁLCULO**	12	16	20

Um detalhe importante é que, tirando os dados combo, da LRC, não ocorre o preenchimento de 4 aulas no mesmo dia da semana. Ex.: 9,10 e 11,12.

Além disso também não ocorre o preenchimento ficando o intervalo entre 2 aulas, exemplo 2,3 e 5,6. (Tabela 4)

TABELA 4 - Preenchimento inválido

Segunda-feira	Terça-feira	Quarta-feira	Quinta-feira	Sexta-feira
1	**ALGORITMOS**	9	**CÁLCULO**	17
ALGORITMOS	**ALGORITMOS**	10	**CÁLCULO**	18
Intervalo	Intervalo	Intervalo	Intervalo	Intervalo
ALGORITMOS	7	11	**CÁLCULO**	19
4	8	12	**CÁLCULO**	20

Para matérias que possuem 2 aulas, o preenchimento é feito com as 2 matérias em posições consecutivas em um único dia.

3.3.2 Dados de entrada

Antes da execução do programa, é preciso preencher arquivos com os dados que serão utilizados para o processo de geração da tabela de horário. Devem seguir a sequência abaixo:

- Arquivo com a disponibilidade dos professores;
- Arquivos dos períodos com as matérias, respectivos professores e quantidade de aula por semana;
- Arquivo com as matérias combo ou a LRC.

Para que a leitura dos dados seja feita de forma correta, é preciso seguir um padrão no preenchimento dos arquivos.

3.3.2.1 Preenchimento da disponibilidade dos professores

O arquivo que foi definido com o nome “prof.txt” (FIG. 5) possui os dados dos professores tais como:

- Número; (número de identificação interno)
- Nome;
- Disponibilidade.

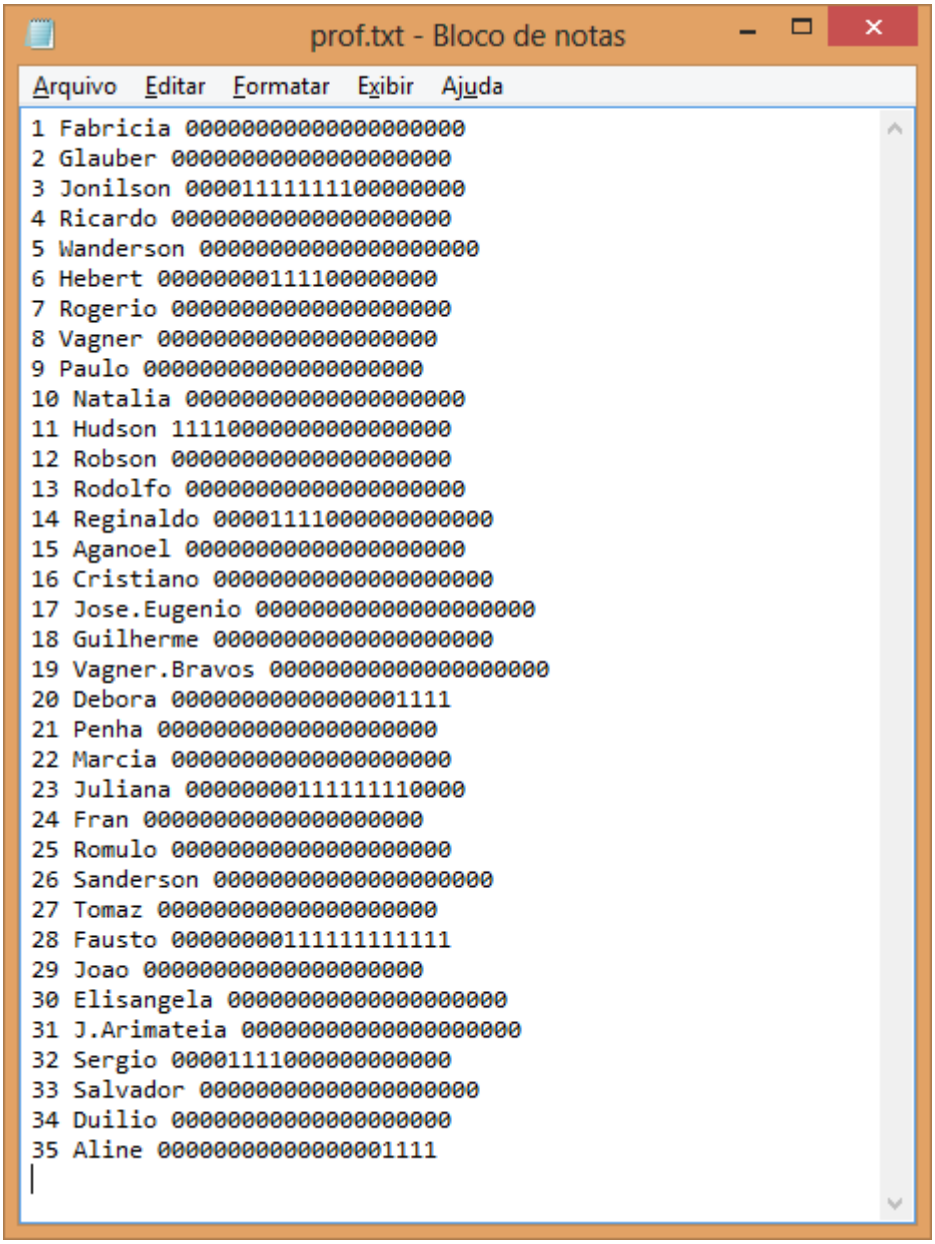

FIGURA 5 - Arquivo prof.txt, simulação com a disponibilidade dos professores

O número de cada professor foi definido simplesmente pela ordem em que os mesmos foram dispostos no arquivo. Após o número deve haver um espaço e após o espaço, vem o nome do professor. O tamanho desse nome deve ter no máximo 25 caracteres e caso seja composto não deve ser utilizado espaço entre as palavras, usar “.”(ponto) para separá-las.

A sequência de 20 dígitos a seguir corresponde aos 20 horários que cada professor tem na semana, respectivamente de segunda a sexta. Os dígitos devem ser preenchidos com “0” (zero) ou “1” (um), caso o professor esteja disponível ou não, respectivamente, para lecionar num determinado horário. Vide layout na FIG 6.

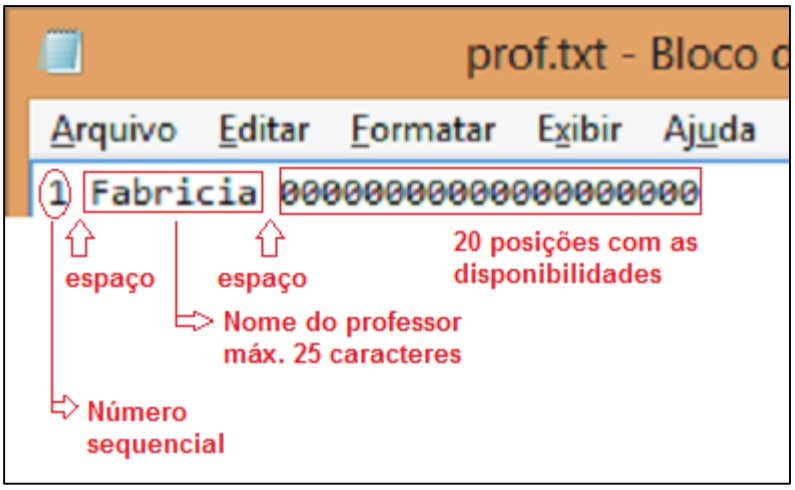

FIGURA 6 – Layout de preenchimento do arquivo com os dados dos professores

Observando a figura 4 como exemplo, pode-se ver que a professora 35, Aline, está disponível para lecionar de segunda a quinta-feira devido aos 16 primeiros dígitos (4 da segunda, 4 da terça, 4 da quarta e 4 da quinta) estarem preenchidos com “0”, e os 4 últimos (da sexta) preenchidos com “1”.

3.3.2.2 Preenchimento dos períodos

Os períodos consistem nos seguintes dados, que podem ser vistos na FIG. 7:

- Número do professor responsável
- Quantidade de aula por semana
- Nome da matéria

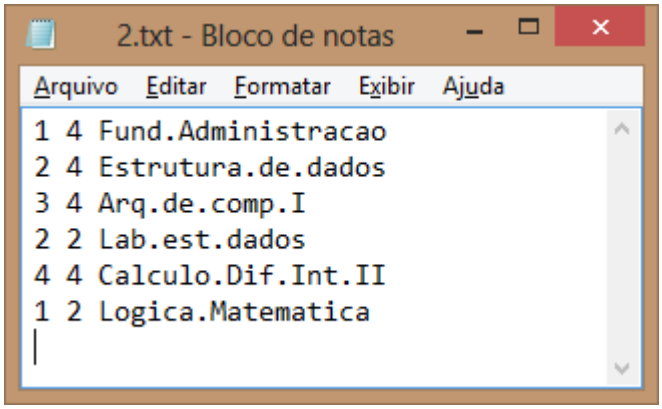

FIGURA 7 - Período com dados no padrão necessário

A estrutura é a seguinte: Número do professor que deve ser obtido do arquivo prof.txt, ”espaço”, quantidade de aulas da matéria na semana, ”espaço” e o nome da matéria, utilizando “.” para separar as palavras, sem espaço entre elas. O nome também deve respeitar a quantidade máxima de 25 caracteres. O layout está na FIG. 8.

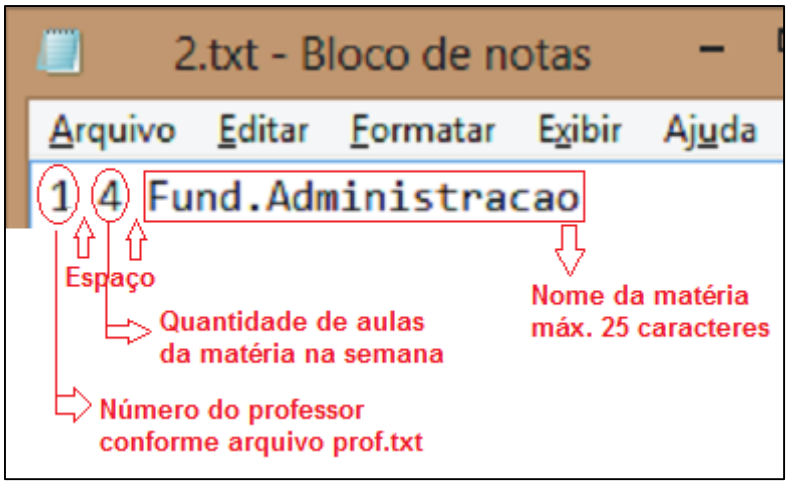

FIGURA 8 – Layout de preenchimento dos arquivos das turmas

Para o nome do arquivo deve ser utilizado um número que foi definido na implementação, de 1 a 39(.txt), sendo esse último valor o limite do total de períodos. O número pode ser qualquer um para qualquer período. Ele será utilizado para identificação na tabela final que for gerada.

3.3.2.3 Preenchimento da LRC (lista restrita de candidatos)

Após o preenchimento das restrições dos professores e dos períodos e suas respectivas matérias, é feito o preenchimento da LRC.

É necessário verificar matérias em comum entre os períodos, que serão ministradas por um mesmo professor. É possível combinar até 5 turmas. Depois é preciso passar essas informações para o programa, indicando quais matérias de quais períodos farão parte de um “combo”. Esses dados são definidos conforme FIG. 9, da seguinte forma:

- Número do período
- Número da posição da matéria no período

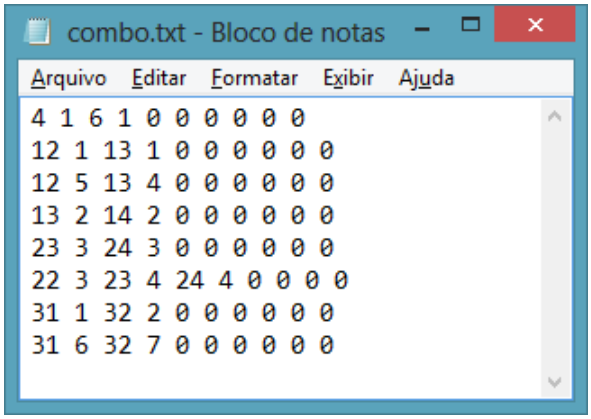

FIGURA 9 - Dados das matérias “combo”

As posições ímpares de cada linha referem-se ao período e as posições pares definem a posição(linha) no arquivo do período correspondente, em que a matéria se encontra. Cada linha deve conter 10 dígitos, sendo estes preenchidos com “0” (zero) caso não haja mais matéria em outros períodos que combinem com as que foram incluídas. Layout na FIG. 10.

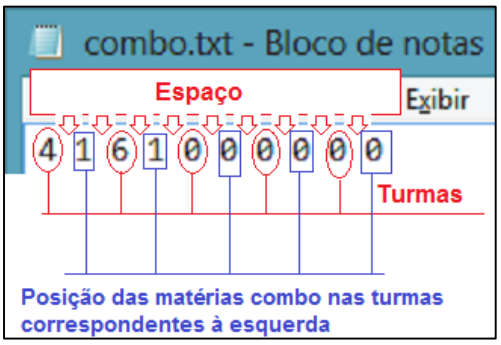

FIGURA 10 - Layout de preenchimento do arquivo com as matérias combo

No exemplo da FIG. 11 pode-se observar na primeira linha a indicação de uma mesma matéria que encontra-se no 4º período na primeira posição e no 6º período na mesma posição.

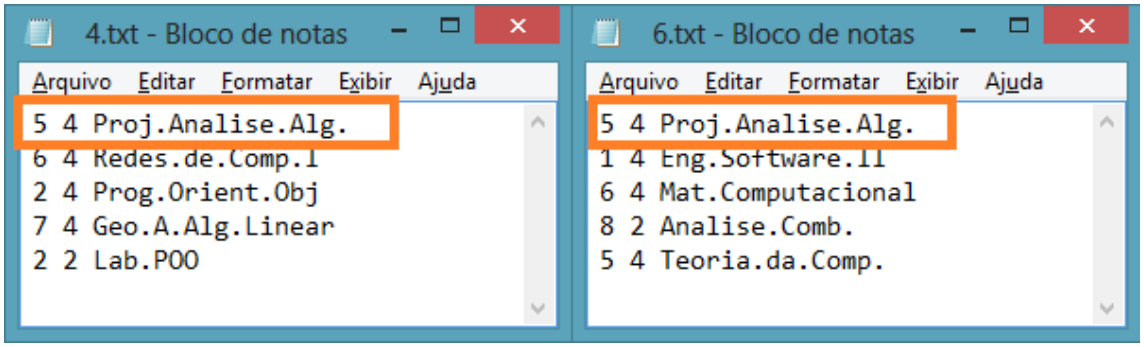

FIGURA 11 - Matérias de períodos diferentes que são dadas no mesmo horário pelo mesmo professor.

3.3.3 Processamento dos dados

A FIG. 12 demonstra como é o fluxo de trabalho do algoritmo.

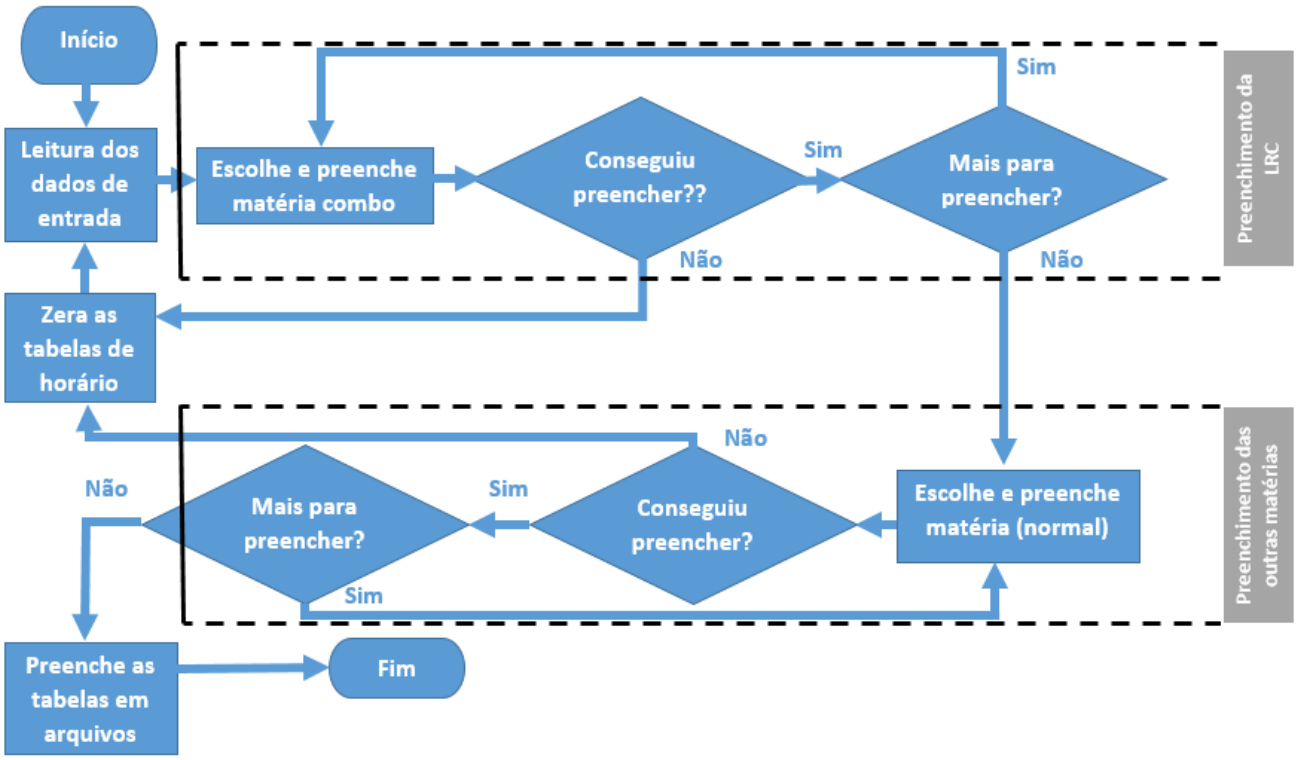

FIGURA 12 - Fluxograma macro do algoritmo desenvolvido

Descrevendo detalhadamente cada processo do algoritmo seguindo o fluxograma temos a seguinte ordem:

1) Inicia-se o programa: É exibido uma tela inicial antes de começar o processamento para que o usuário possa personalizar a saída desejada. As opções disponíveis são a escolha dos períodos desejados, ou todos, tabela de horário com janela ou não e quantidade de tabelas desejadas com o máximo de 10. A FIG. 13 ilustra as opções disponíveis.

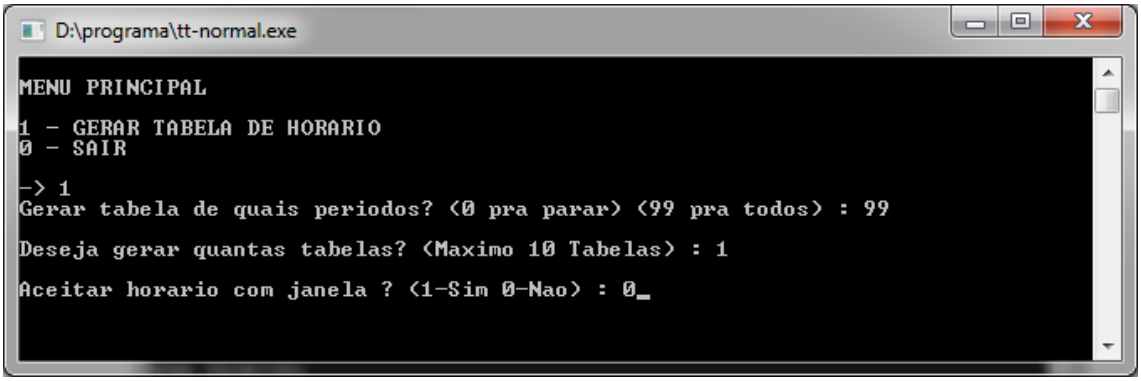

FIGURA 13 - Tela principal do programa

2) Faz a leitura dos dados de entrada: É feito o carregamento dos dados de arquivos para a memória, das matérias(xx.txt, sendo xx um número), disponibilidades de professores(prof.txt) e da LRC(combo.txt).
3) Escolhe aleatoriamente uma matéria combo da LRC para tentar preencher: sorteia-se uma combinação da lista e tenta preenchê-la. Verifica disponibilidade de horário do professor e na tabela de horário e preenche. (FIG. 14)

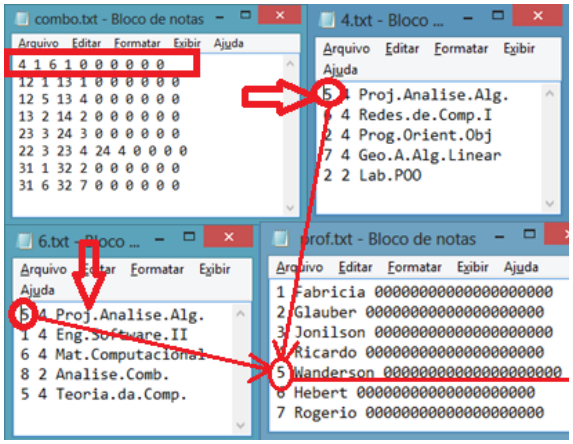

FIGURA 14 - Verificando disponibilidade do professor

TABELA 5 - Tabela do 4º período preenchida

Segunda-feira	Terça-feira	Quarta-feira	Quinta-feira	Sexta-feira
5 Proj.Analise.Alg	5 Proj.Analise.Alg	9	13	17
5 Proj.Analise.Alg	5 Proj.Analise.Alg	10	14	18
Intervalo	Intervalo	Intervalo	Intervalo	Intervalo
3	7	11	15	19
4	8	12	16	20

TABELA 6 - Tabela do 6º período preenchida

Segunda-feira	Terça-feira	Quarta-feira	Quinta-feira	Sexta-feira
5 Proj.Analise.Alg	5 Proj.Analise.Alg	9	13	17
5 Proj.Analise.Alg	5 Proj.Analise.Alg	10	14	18
Intervalo	Intervalo	Intervalo	Intervalo	Intervalo
3	7	11	15	19
4	8	12	16	20

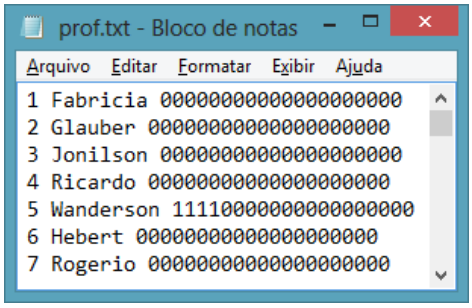

FIGURA 15 - Disponibilidade do professor após o preenchimento

Observa-se que na tabela 5 e na tabela 6 foram preenchidas as matérias que estão na LRC e que serão ministradas pelo professor Wanderson (nº 5) que teve sua disponibilidade atualizada conforme FIG. 15.

4) Se conseguiu preencher e ainda existirem itens combo na lista, volta ao passo 3, escolhe um item que não foi escolhido anteriormente e tenta preencher. Se não conseguiu, apaga o que já foi preenchido e recomeça no passo 2. Se já foi preenchido todos os itens da LRC avança para o passo 5;
5) Após o preenchimento das matérias da LRC escolhe aleatoriamente uma matéria restante de um período qualquer para tentar preencher;
6) Se conseguiu preencher e ainda há itens para serem preenchidos, volta ao passo 5. Se não conseguiu preencher apaga o que já foi preenchido e recomeça no passo 2. Se já foi preenchido todos os itens avança para o passo 7;
7) Escreve as tabelas preenchidas em arquivos;
8) Finaliza o programa.

Implementou-se um tempo limite para que as tabelas de horário sejam geradas, de 5 minutos. Caso esse tempo seja alcançado o programa para e emite um aviso. Logo após isso poderá ser executado novamente. Os resultados serão mostrados no próximo capítulo.

4 RESULTADOS

Para a realização dos testes, foram utilizados dados coletados na secretaria da instituição. Os dados correspondem aos mesmos que foram utilizados para a geração manual da tabela de horário desse semestre, com as mesmas matérias e professores totalmente disponíveis.

Os testes foram realizados em um notebook utilizando as seguintes configurações:

HARDWARE:

- Processador Intel® Core™ i7 720QM 2.8GHz;
- 4 GB RAM;
- HD 500GB HYBRID SSD (7200RPM);

SOFTWARE:

- Sistema operacional Microsoft® Windows™ 8 Pro.
- Notepad++ ver. 6.4.5
- Borland C++ Compiler version 5.5 *command-line.*

O programa foi executado com diferentes configurações de parâmetros e foi feito, através de um arquivo de log gerado pelo programa, comparativo entres. Todos esses parâmetros influenciam diretamente no tempo e qualidade das tabelas geradas.

Os parâmetros possíveis são listados abaixo:

- Preenchimento individual de matérias;
- Aceitação ou não de janelas;
- Quantidade de tabelas que deve gerar.

O primeiro item listado deve ser configurado diretamente no código e o programa recompilado. Os demais são feitos durante a execução da aplicação.

A disponibilidade de alocação da matéria na tabela de horário e a disponibilidade de horário do professor, é sempre verificada.

Para cada teste realizado foi mostrado o valor utilizado de cada variável.

Foi exibido apenas uma das três tabelas de horário que foram geradas em cada configuração. Além disso apenas o curso de ciência da computação foi mostrado como exemplo.

Detalhe importante: o tempo gasto mostrado das tabelas geradas abaixo tem o intuito de apenas fazer um comparativo entre os testes executados demonstrando a diferença de complexidade de cada configuração. A execução desses mesmos testes num Hardware/Software diferente muito provavelmente proporcionará um desempenho temporal diferente do que se vê aqui.

O primeiro teste foi feito com as seguintes configurações da tabela 7:

TABELA 7 - Parâmetros, teste 1.

Parâmetro	**Valor**
Preenchimento individual de matérias	Sim
Aceita janelas?	Sim
Quantidade de tabelas geradas	3

Pode-se dizer que essa configuração é a mais “relaxada”, tornando a geração pouco complexa. O resultado é uma geração normalmente rápida, porém com uma qualidade muito comprometida.

TABELA 8 - Quadro de horários, teste 1.

2º período				
Segunda-feira	**Terça-feira**	**Quarta-feira**	**Quinta-Feira**	**Sexta-Feira**
03 Arq.de.comp.I	04 Calculo.Dif.Int.II	02 Estrutura.de.dados	02 Lab.est.dados	01 Fund.Administracao
03 Arq.de.comp.I	04 Calculo.Dif.Int.II	02 Estrutura.de.dados	02 Lab.est.dados	01 Fund.Administracao
03 Arq.de.comp.I	04 Calculo.Dif.Int.II	02 Estrutura.de.dados	01 Logica.Matematica	01 Fund.Administracao
03 Arq.de.comp.I	04 Calculo.Dif.Int.II	02 Estrutura.de.dados	01 Logica.Matematica	01 Fund.Administracao

4º período				
Segunda-feira	**Terça-feira**	**Quarta-feira**	**Quinta-Feira**	**Sexta-Feira**
05 Proj.Analise.Alg.	02 Prog.Orient.Obj		07 Geo.A.Alg.Linear	02 Lab.POO
05 Proj.Analise.Alg.	02 Prog.Orient.Obj		07 Geo.A.Alg.Linear	02 Lab.POO
05 Proj.Analise.Alg.	02 Prog.Orient.Obj	06 Redes.de.Comp.I	07 Geo.A.Alg.Linear	06 Redes.de.Comp.I
05 Proj.Analise.Alg.	02 Prog.Orient.Obj	06 Redes.de.Comp.I	07 Geo.A.Alg.Linear	06 Redes.de.Comp.I

6º período				
Segunda-feira	**Terça-feira**	**Quarta-feira**	**Quinta-Feira**	**Sexta-Feira**
05 Proj.Analise.Alg.	05 Teoria.da.Comp.	08 Analise.Comb.	01 Eng.Software.II	06 Mat.Computacional
05 Proj.Analise.Alg.	05 Teoria.da.Comp.	08 Analise.Comb.	01 Eng.Software.II	06 Mat.Computacional
05 Proj.Analise.Alg.	05 Teoria.da.Comp.	01 Eng.Software.II	06 Mat.Computacional	
05 Proj.Analise.Alg.	05 Teoria.da.Comp.	01 Eng.Software.II	06 Mat.Computacional	

8º período				
Segunda-feira	**Terça-feira**	**Quarta-feira**	**Quinta-Feira**	**Sexta-Feira**
01 TCC.II	06 Pesq.Operacional	06 Pesq.Operacional	03 Compiladores	
01 TCC.II	06 Pesq.Operacional	06 Pesq.Operacional	03 Compiladores	
01 Optativa.IV	01 Optativa.IV	03 Compiladores		
01 Optativa.IV	01 Optativa.IV	03 Compiladores		

Na tabela 8 observa-se que os professores foram separados por cores, facilitando a visualização da existência ou não de conflitos de horário.

Um dos detalhes que pode-se notar é a existência de matéria combo no 4º e no 6º período na segunda-feira com a matéria “Proj.Analise.Alg” com o professor nº 05, de acordo com o arquivo de professores.

Fazendo uma análise qualitativa da tabela de horário gerada pode-se observar que há um grande número de ocorrências de 4 aulas seguidas de uma mesma matéria no mesmo dia. No 2º período por exemplo apenas a quinta-feira não ocorre tal situação. Um outro detalhe observado é a existência de uma “janela” no 4º período na quarta-feira, nos 2 primeiros horários.

O arquivo log.txt da FIG. 16 exibe detalhes do resultado da geração, tabela por tabela, tais como o tempo gasto em cada uma delas e a quantidade de vezes em que foi preciso recarregar, ou seja, houve um conflito durante algum momento da tentativa de incluir alguma matéria na tabela de horário e com isso foi necessário limpar toda a tabela e recomeçar o processo.

Observa-se após o teste 1 que a geração das 3 tabelas demorou menos que 1 segundo para cada uma delas. Em 10 execuções com as mesmas configurações de entrada o tempo em momento algum ultrapassou 1 segundo. Apenas a quantidade de vezes em que precisou recarregar, sofreu variação maior.

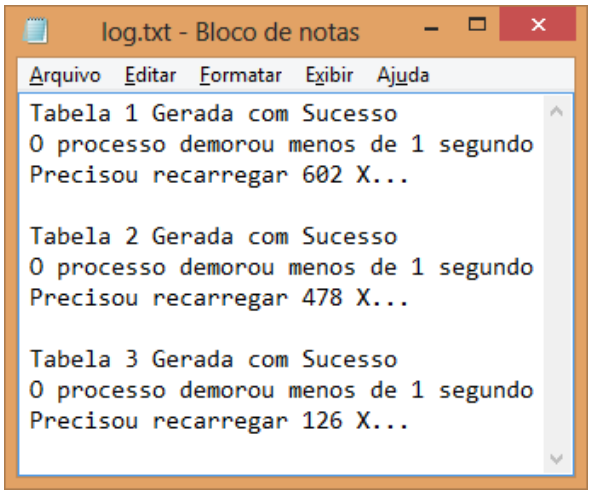

FIGURA 16 - Log.txt, resultado de geração do teste 1

O segundo teste foi efetuado com as seguintes configurações:

TABELA 9 – Parâmetros, teste 2.

Parâmetro	Valor
Preenchimento individual de matérias	Não
Aceita janelas?	Sim
Quantidade de tabelas geradas	3

Com a não ocorrência do preenchimento individual, o tempo aumentou pouca coisa e a tabela passou a ter uma qualidade melhor.

TABELA 10 - Quadro de horários, teste 2.

2º período				
Segunda-feira	**Terça-feira**	**Quarta-feira**	**Quinta-Feira**	**Sexta-Feira**
03 Arq.de.comp.I	03 Arq.de.comp.I	02 Estrutura.de.dados	02 Estrutura.de.dados	04 Calculo.Dif.Int.II
03 Arq.de.comp.I	03 Arq.de.comp.I	02 Estrutura.de.dados	02 Estrutura.de.dados	04 Calculo.Dif.Int.II
01 Fund.Administracao	01 Fund.Administracao	01 Logica.Matematica	04 Calculo.Dif.Int.II	02 Lab.est.dados
01 Fund.Administracao	01 Fund.Administracao	01 Logica.Matematica	04 Calculo.Dif.Int.II	02 Lab.est.dados

4º período				
Segunda-feira	**Terça-feira**	**Quarta-feira**	**Quinta-Feira**	**Sexta-Feira**
05 Proj.Analise.Alg.	05 Proj.Analise.Alg.	07 Geo.A.Alg.Linear		02 Prog.Orient.Obj
05 Proj.Analise.Alg.	05 Proj.Analise.Alg.	07 Geo.A.Alg.Linear		02 Prog.Orient.Obj
06 Redes.de.Comp.I	06 Redes.de.Comp.I	02 Lab.POO	02 Prog.Orient.Obj	07 Geo.A.Alg.Linear
06 Redes.de.Comp.I	06 Redes.de.Comp.I	02 Lab.POO	02 Prog.Orient.Obj	07 Geo.A.Alg.Linear

6º período				
Segunda-feira	**Terça-feira**	**Quarta-feira**	**Quinta-Feira**	**Sexta-Feira**
05 Proj.Analise.Alg.	05 Proj.Analise.Alg.		01 Eng.Software.II	01 Eng.Software.II
05 Proj.Analise.Alg.	05 Proj.Analise.Alg.		01 Eng.Software.II	01 Eng.Software.II
05 Teoria.da.Comp.	05 Teoria.da.Comp.	08 Analise.Comb.	06 Mat.Computacional	06 Mat.Computacional
05 Teoria.da.Comp.	05 Teoria.da.Comp.	08 Analise.Comb.	06 Mat.Computacional	06 Mat.Computacional

8º período				
Segunda-feira	**Terça-feira**	**Quarta-feira**	**Quinta-Feira**	**Sexta-Feira**
01 Optativa.IV	01 Optativa.IV	06 Pesq.Operacional	06 Pesq.Operacional	
01 Optativa.IV	01 Optativa.IV	06 Pesq.Operacional	06 Pesq.Operacional	
03 Compiladores	03 Compiladores		01 TCC.II	
03 Compiladores	03 Compiladores		01 TCC.II	

A matéria combo foi alocada nos 2 primeiros horários de segunda-feira e terça-feira. Há pelo menos 2 janelas, no 4º período na quinta-feira e 6º período na quarta-feira. Aqui conforme configurado não ocorrem 4 aulas seguidas de uma mesma matéria no mesmo dia da semana.

Verificando o arquivo de log na FIG. 17 observa-se que após 10 execuções o tempo medido foi de até 10 segundos para gerar uma tabela. A quantidade de vezes que precisou recarregar também aumentou consideravelmente.

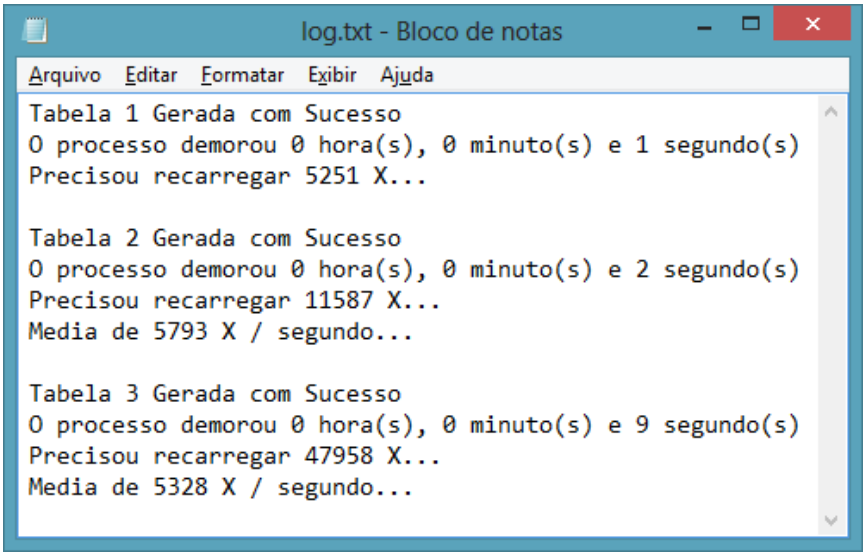

FIGURA 17 - Log.txt, resultado de geração do teste 2

O terceiro teste ficou configurado como está na tabela 11, assim:

TABELA 11 – Parâmetros, teste 3.

Parâmetro	Valor
Preenchimento individual de matérias	Sim
Aceita janelas?	Não
Quantidade de tabelas geradas	3

Com os valores dos parâmetros acima, a tabela passou a demorar mais do que os testes anteriores. A aceitação de janelas demonstrou mais “custosa” do que o preenchimento individual de matérias.

A tabela 12 do teste 3 está logo abaixo:

TABELA 12 – Quadro de horários, teste 3.

2º período				
Segunda-feira	**Terça-feira**	**Quarta-feira**	**Quinta-Feira**	**Sexta-Feira**
02 Estrutura.de.dados	01 Logica.Matematica	04 Calculo.Dif.Int.II	03 Arq.de.comp.I	01 Fund.Administracao
02 Estrutura.de.dados	01 Logica.Matematica	04 Calculo.Dif.Int.II	03 Arq.de.comp.I	01 Fund.Administracao
02 Estrutura.de.dados	04 Calculo.Dif.Int.II	02 Lab.est.dados	03 Arq.de.comp.I	01 Fund.Administracao
02 Estrutura.de.dados	04 Calculo.Dif.Int.II	02 Lab.est.dados	03 Arq.de.comp.I	01 Fund.Administracao

4º período				
Segunda-feira	**Terça-feira**	**Quarta-feira**	**Quinta-Feira**	**Sexta-Feira**
05 Proj.Analise.Alg.	07 Geo.A.Alg.Linear	02 Lab.POO	06 Redes.de.Comp.I	02 Prog.Orient.Obj
05 Proj.Analise.Alg.	07 Geo.A.Alg.Linear	02 Lab.POO	06 Redes.de.Comp.I	02 Prog.Orient.Obj
05 Proj.Analise.Alg.	07 Geo.A.Alg.Linear		06 Redes.de.Comp.I	02 Prog.Orient.Obj
05 Proj.Analise.Alg.	07 Geo.A.Alg.Linear		06 Redes.de.Comp.I	02 Prog.Orient.Obj

6º período				
Segunda-feira	**Terça-feira**	**Quarta-feira**	**Quinta-Feira**	**Sexta-Feira**
05 Proj.Analise.Alg.	06 Mat.Computacional	01 Eng.Software.II	05 Teoria.da.Comp.	05 Teoria.da.Comp.
05 Proj.Analise.Alg.	06 Mat.Computacional	01 Eng.Software.II	05 Teoria.da.Comp.	05 Teoria.da.Comp.
05 Proj.Analise.Alg.	06 Mat.Computacional	01 Eng.Software.II	08 Analise.Comb.	
05 Proj.Analise.Alg.	06 Mat.Computacional	01 Eng.Software.II	08 Analise.Comb.	

8º período				
Segunda-feira	**Terça-feira**	**Quarta-feira**	**Quinta-Feira**	**Sexta-Feira**
01 TCC.II	03 Compiladores	06 Pesq.Operacional	01 Optativa.IV	
01 TCC.II	03 Compiladores	06 Pesq.Operacional	01 Optativa.IV	
03 Compiladores	01 Optativa.IV	06 Pesq.Operacional		
03 Compiladores	01 Optativa.IV	06 Pesq.Operacional		

Aqui não há janelas, mas a ocorrência das matérias seguidas está presente. O tempo gasto para a geração das tabelas subiu. (FIG. 18.)

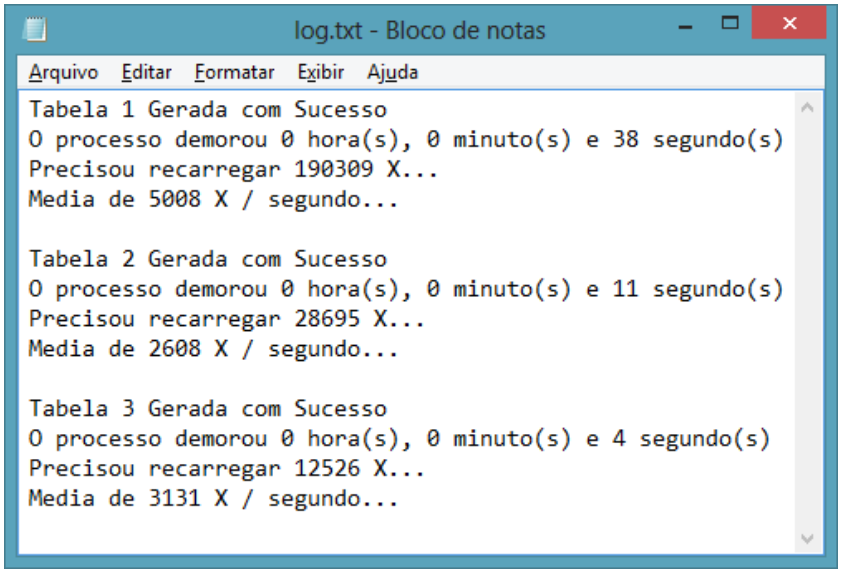

FIGURA 18 - Resultado de geração do teste 3

E por fim, o quarto e último teste:

TABELA 13 – Parâmetros, teste 4.

Parâmetro	Valor
Preenchimento individual de matérias	Não
Aceita janelas?	Não
Quantidade de tabelas geradas	3

Sem o preenchimento individual e com a não aceitação de janelas, a geração ficou bem mais complexa. Em vários testes executados o tempo limite estipulado de 5:00 minutos para a geração foi alcançado sem que tenha terminado as 3 tabelas.

TABELA 14 - Quadro de horários, teste 4.

2º período				
Segunda-feira	**Terça-feira**	**Quarta-feira**	**Quinta-Feira**	**Sexta-Feira**
02 Lab.est.dados	01 Logica.Matematica	01 Fund.Administracao	01 Fund.Administracao	03 Arq.de.comp.I
02 Lab.est.dados	01 Logica.Matematica	01 Fund.Administracao	01 Fund.Administracao	03 Arq.de.comp.I
02 Estrutura.de.dados	04 Calculo.Dif.Int.II	03 Arq.de.comp.I	04 Calculo.Dif.Int.II	02 Estrutura.de.dados
02 Estrutura.de.dados	04 Calculo.Dif.Int.II	03 Arq.de.comp.I	04 Calculo.Dif.Int.II	02 Estrutura.de.dados

4º período				
Segunda-feira	**Terça-feira**	**Quarta-feira**	**Quinta-Feira**	**Sexta-Feira**
05 Proj.Analise.Alg.	05 Proj.Analise.Alg.	02 Prog.Orient.Obj	02 Prog.Orient.Obj	07 Geo.A.Alg.Linear
05 Proj.Analise.Alg.	05 Proj.Analise.Alg.	02 Prog.Orient.Obj	02 Prog.Orient.Obj	07 Geo.A.Alg.Linear
06 Redes.de.Comp.I	06 Redes.de.Comp.I	02 Lab.POO	07 Geo.A.Alg.Linear	
06 Redes.de.Comp.I	06 Redes.de.Comp.I	02 Lab.POO	07 Geo.A.Alg.Linear	

6º período				
Segunda-feira	**Terça-feira**	**Quarta-feira**	**Quinta-Feira**	**Sexta-Feira**
05 Proj.Analise.Alg.	05 Proj.Analise.Alg.	05 Teoria.da.Comp.	05 Teoria.da.Comp.	06 Mat.Computacional
05 Proj.Analise.Alg.	05 Proj.Analise.Alg.	05 Teoria.da.Comp.	05 Teoria.da.Comp.	06 Mat.Computacional
01 Eng.Software.II	01 Eng.Software.II	08 Analise.Comb.	06 Mat.Computacional	
01 Eng.Software.II	01 Eng.Software.II	08 Analise.Comb.	06 Mat.Computacional	

8º período				
Segunda-feira	**Terça-feira**	**Quarta-feira**	**Quinta-Feira**	**Sexta-Feira**
01 TCC.II	03 Compiladores	06 Pesq.Operacional	06 Pesq.Operacional	
01 TCC.II	03 Compiladores	06 Pesq.Operacional	06 Pesq.Operacional	
03 Compiladores		01 Optativa.IV	01 Optativa.IV	
03 Compiladores		01 Optativa.IV	01 Optativa.IV	

A tabela acima está livre de janelas e 4 aulas seguidas. Mas para se alcançar esse feito teve um alto preço. Com as regras de preenchimento mais rígidas, isso resultou em maior complexidade e consequentemente acarretou num tempo muito maior do que as outras tabelas.

O tempo limite foi alcançado e apenas 2 tabelas foram geradas.

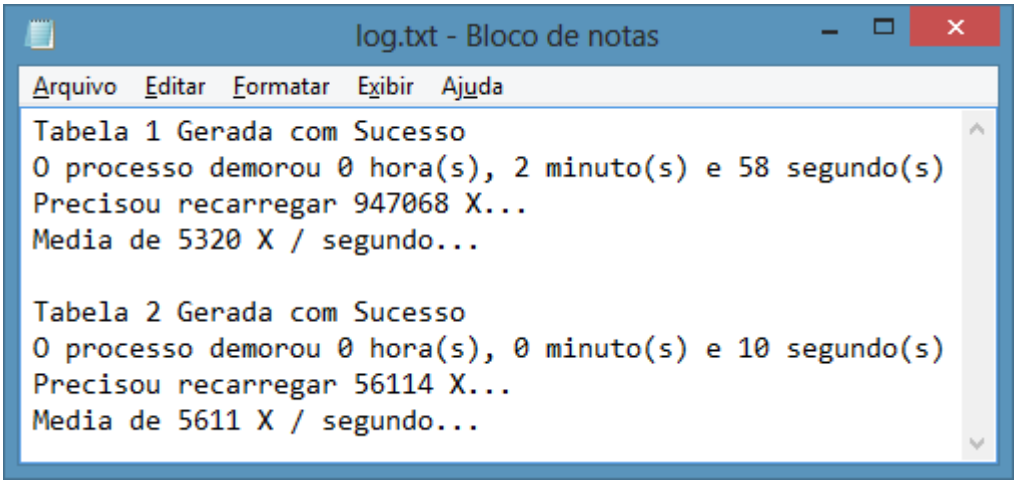

FIGURA 19 - Resultado de geração do teste 4

A tabela 15 abaixo reúne os resultados dos 4 testes, proporcionando uma comparação entre eles.

TABELA 15 - Comparativo entre os testes

	Teste 1			Teste 2			Teste 3			Teste 4		
Preenchimento individual	Sim			Não			Sim			Não		
Aceita Janelas	Sim			Sim			Não			Não		
Quantidade de tabelas	3			3			3			3		
	Tabela 1	Tabela 2	Tabela 3	Tabela 1	Tabela 2	Tabela 3	Tabela 1	Tabela 2	Tabela 3	Tabela 1	Tabela 2	Tabela 3
Tempo gasto de geração em(mm:ss)	00:01	00:01	00:01	00:01	00:02	00:09	00:38	00:11	00:04	02:58	00:10	-
Recarregou X vezes	602	478	126	5.251	11.587	47.958	190.309	28.695	12.526	947.068	56.114	-

A tendência em que se buscava uma tabela com uma melhor qualidade, foi aumentando o nível de complexidade e consequentemente gerou demora no processamento das informações e geração da tabela final.

5 CONCLUSÃO

O problema de geração de tabelas de horário tem sido amplamente difundido na literatura em busca de meios para se conseguir resultados mais satisfatórios no seu tratamento. Com o estudo realizado foi possível identificar a importância e ao mesmo tempo saber da dificuldade que as instituições de ensino enfrentam todo início de semestre letivo para a geração da tabela de horário.

Com o conhecimento obtido foi desenvolvido uma aplicação utilizando a metaheurística GRASP possibilitando a geração das tabelas de horário dos cursos de exatas da FIC de uma forma automatizada e mais rápida.

Devido a complexidade do problema o algoritmo apresentou dificuldade em alguns momentos para exibir o resultado final em tempo hábil. Para contornar o problema foi estipulado um tempo limite de execução.

Foi possível observar nos resultados gerados que a aplicação das regras de preenchimento buscando uma melhor qualidade na geração da tabela de horário, influenciava diretamente no tempo de processamento e exibição dos resultados.

Por fim, a aplicação desenvolvida oferece até 10 opções de escolha de tabelas geradas por cada execução do programa permitindo que o departamento de exatas da instituição proporcione melhor organização das aulas à alunos e professores.

6 TRABALHOS FUTUROS

Para trabalhos futuros sugere-se a implementação de modificações buscando melhorias no processo de desenvolvimento das tabelas de horário, como por exemplo:

- Desenvolvimento de uma interface gráfica para o algoritmo, afim de tornar a aplicação mais amigável;
- Implementação de funções que possam avaliar e escolher melhor os itens que serão preenchidos;
- Adaptação do algoritmo para atender todos os cursos da instituição e não apenas um departamento;

Por fim, fazer um comparativo com outros métodos existentes atestando melhor a performance deste.

REFERÊNCIAS

Alvarenga, G.B; Mateus, G. R. ***A Two-Phase Genetic and Set Partitioning Approach for the Vehicle Routing Problem with Time Windows***. 2004 IN: OLIVEIRA A. C. **Uso do algoritmo genético e recozimento simulado para o problema de alocação de salas.** Monografia Universidade Federal de Lavras (2006)

ANDRADE, C.E.; Batista, F.L.C; Toso, R.F. **Modelo de Otimização para Transporte de Cargas em Ambientes Reduzidos,** 2004. Disponível em: http://www.ic.unicamp.br/~andrade/publications/monografia.pdf. Acesso em Agosto de 2013.

ANDRADE, Eliana X. L. SAMPAIO, Rubens. SILVA, GERALDO. Introdução à Construção de Modelos de Otimização Linear e Inteira - São Carlos, SP : SBMAC; São Paulo: Plêiade, 2005, 82 p. - Notas em Matemática Aplicada; 18

BECCENERI, J. C. Minicurso Cap. 2, **Meta-Heurísticas e Otimização Combinatória: Aplicações em Problemas Ambientais**. Instituto Nacional de pesquisas espaciais (INPE), São José dos Campos. Disponível em: < http://www.lac.inpe.br/ELAC13/arquivos/MiniCurso_02ELAC2012.pdf>. Acesso em Agosto de 2013.

CORMEN, Tomas H.; LEISERSON, Charles E.; RIVEST, RONALD L.; STEIN, Clifford. **Algoritmos teoria e prática**. Rio de Janeiro. Editora Elsevier, 6ª Tiragem, 2002.

DOS SANTOS, HENRIQUE JOSÉ. **Curso de Linguagem C** - UFMG <**http://www.ead.cpdee.ufmg.br/cursos/C/**>. Acessado em Agosto de 2013

EVEN, S., Itai, A. and Shamir, A. **On the complexity of timetabling and multicommodity flow problems**, *SIAM Journal of Computation*, 5:691-703, 1976.

FERREIRA E GLAZAR. **Definição de parâmetros na utilização de metaheurísticas para a programação de horários escolares.** Revista Educação e Tecnologia. P. 2. 2005

FREITAS, F.G., MAIA, C.L.B., COUTINHO, D.P. CAMPOS, G.A.L, SOUZA J.T. **Aplicação de Metaheurísticas em problemas da Engenharia de Software.** II Congresso Tecnológico Infobrasil, 2009.

Glover, F. and Laguna, M. 1997 "Tabu Search", *Kluver academic Publishers*, Boston.

JAMSA, KRIS - **Programando em C/C++ A Bíblia.** 1ª edição São Paulo: Markron Books, 1999.

LOBO, E. L. M. **Uma solução do problema de horário escolar via algoritmo genético paralelo.** Dissertação de mestrado do curso de Modelagem Matemática e Computacional – CEFET-MG, 2005 Disponível em:

<http://www.mmc.cefetmg.br/info/downloads/D006-EduardoLuizMirandaLobo2005.pdf> acessado em: Agosto de 2013.

MACULAN, Nelson. Integer Programming. 2002. Disponível em: < http://www2.ucg.br/Institutos/LabPL/Arquivos/maculan-integer_programming.pdf>. Acessado em Novembro de 2012.

MIT, 2012 ***Operations Research Center*.** Disponível em: <http://www.mit.edu/~orc/>. Acessado em Novembro de 2013

MOURA, SCARAFICCI, SILVEIRA e SANTOS, 2004 **"Técnicas Metaheurísticas aplicadas à construção de grades horárias escolares".** XXXVI-SBPO

Nemhauser, G.L. e Wolsey, L., ***Integer and Combinatorial Otimization***, Wiley, 1988.

P. FESTA E M.G.C. RESENDE, 2002 ***GRASP: An annotated bibliography.*** Ensaios e Pesquisas sobre Metaheurísticas, Kluwer Academic Publishers. Disponível em: <http://www.research.att.com/~mgcr/doc/gabib.pdf>. Acessado em Outubro de 2013.

PATAT (2012) **–** ***9th International Conference on the Practice and Theory of Automated Timetabling***. Disponível em: <http://www.patat2012.com/information.html> acessado em Outubro de 2013.

SCHAEFER, A. "***A survey of automated timetabling***", 1990. In: COSTA, F. P. Programação de Horários em Escolas via GRASP e Busca Tabu. UFOP. 2003. P. 10.

Silva, A. S. N. ***Estudo* e Implementação, Mediante Recozimento Simulado, do Problema de Alocação de Salas**. (2005) http://www.bcc.ufla.br/wp-content/uploads/2013/2005/Estudo_e_implementacao_mediante_recozimento_simulado_do_problema_de_alocacao_de_sala.pdf. Acessado em Setembro de 2013

SILVA, G. C., OCHI L. S. MARTINS, S. L, 2006. **Proposta e avaliação de heurísticas grasp para o problema da diversidade máxima.** Pesquisa operacional. Disponível em: <http://www.scielo.br/scielo.php?script=sci_arttext&pid=S0101-74382006000200007#back10> acessado em Outubro de 2013.

SOUZA, M. J. F. **Problema do Caixeiro Viajante com Coleta de Prêmios.** Disponível em: <http://www.decom.ufop.br/prof/marcone/Disciplinas/OtimizacaoCombinatoria/PCVCP.pdf> acessado em Setembro de 2013.

TOSCANI, Laira Vieira; VELOSO, Paulo. A. S. **Complexidade de algoritmos**. Instituto de Informática da Universidade Federal do Rio Grande Do Sul, 1ª edição. Porto Alegre. Editora Sagra Luzzatto, 2002, página 7.)

Ueda et al. (2001) ***A Co-evolving Timeslot/Room Assignment Genetic Algorithm echnique for Universities Timetabling*** In: OLIVEIRA, A. C. 2006 **uso do algoritmo genético e recozimento simulado para o problema de alocação de salas.** Monografia Universidade Federal de Lavras 2006

WERRA, D., "***Construction of School Timetables by Flow Methods***", INFOR – Canadian Journal of Operations Research and Information Processing, 1971 IN: LOBO, E. L. M. **Uma solução do problema de horário escolar via algoritmo genético paralelo.** Dissertação de mestrado do curso de Modelagem Matemática e Computacional – CEFET-MG, 2005

ZIVIANI, Nivio. **Projeto de Algoritmos:** com Implementações em Pascal e C. 1ª edição. São Paulo. Editora Pioneira Thomson Learning, 2002.

APÊNDICE A – Código fonte da aplicação desenvolvida

```
#include<stdio.h>
#include<conio.h>
#include<stdlib.h>
#include<io.h>
#include<string.h>  //strlen
#include<time.h>    //rand
#include <fcntl.h>  //open

// periodo - Arquivo com os dados dos periodos(professor,qtde aula,materia)
// prof      - Arquivo com o numero do professor, nome e disponibilidade
// saida     - Arquivo de saida com o horario gerado
//combo    - Arquivo que contem a combinacao de horarios
//*************************************************************************
**************************
FILE *periodo,*prof,*saida,*combo,*log;

//Estrutura professores - recebe os dados do arquivo prof
//*************************************************************************
**************************
struct professores {
   char nome[25];      //nome do professor
   int restricao[21];  //disponibilidade do professor
}professor[40];

//Estrutura mater - recebe os dados do arquivo do periodo
//*************************************************************************
**************************
struct mater {
   char nome[10][25];  //nome da materia
   int qaula[10];      //quantidade de aula da materia no periodo
   int prof[10];       //numero do prof da materia
   int qmat;           //quantidade de materia no periodo
   int qaulamat;       //quantidade de todas as aulas de todas as materias do periodo
   int tres;           //quantidade de vezes que ocorrera separacao de materia 3e1 ou 1e3
}materia[40];

//Estrutura hora - é preenchida de acordo com a geração do horario, para depois exibir o
resultado final
//*************************************************************************
**************************
struct hora {
   char tabela[21][25];    //Armazena o nome da materia na respectiva posição
   int nprof[21];          //Numero do prof da materia
   int hor[21];            //Auxiliar que marca se posição já foi usada ou não
}horario[40];

//Variaveis utilizadas
//PRINCIPAIS:
```

```
//reloadrestricao[][] - Armazena as disponibilidades dos professores na memória para acesso mais rápido
//reloadqaula[][]     - Armazena quantidade de aulas das materias na memória para acesso mais rápido
//Combovet[][]        - Armazena materias combinadas de periodos diferentes
//periodos[]          - Armazena os periodos que serao usados para criar a tabela de horario
//
//tempper,temppos,tpos,tper,tper2 - Utilizadas para controlar o sorteio das materias que serão utilizadas para o
//                                  preenchimento do horario
//*************************************************************************************************************
int reloadrestricao[40][21];
int reloadqaula[40][10];
int b,i,j,k,z,h,qtprof,semana[7],quais,erro,cont1,cont2,tentativas,combovet[40][12],comboo k;
int op1=1,periodos[40],todastabelasok,posx,posy,posx2,posy2,janela,tempper[40],temppos[ 40][10],temp2[40],tpos,tper,tper2,valeu,comboa;
int jafoisemana,qualsaida,qs,qsaida,cjanela;
char restri[40],temp[1];
int qtjanela,fp;
int horas,regressivoinicial,tempoesgotado=0;
int mini=10;

div_t sec;  //Função / estrutura responsável em dividir o tempo em horas, minutos e segundos

time_t inicio=0,fim=0,inicioparcial=0,fimparcial=0,inicio2,fim2,regressivo,now;

//Recebe as posições que foram aceitas para o preenchimento na tabela de horarios, para materias de 2 ou 4 aulas
//*************************************************************************************************************
preenchehorario(int a,int b,int c,int d)
{
   for(k=0;k<=24;k++)  horario[c].tabela[a][k]=materia[c].nome[d][k];     //Posição na tabela recebe nome da matéria
   for(k=0;k<=24;k++) horario[c].tabela[b][k]=materia[c].nome[d][k];
   horario[c].hor[a]=1;                                  //Posição é marcada como utilizada
   horario[c].hor[b]=1;
   horario[c].nprof[a]=materia[c].prof[d];                          //Recebe o número do professor
   horario[c].nprof[b]=materia[c].prof[d];
   professor[materia[c].prof[d]].restricao[a]=1;                   //Horário do professor é marcado como ocupado
   professor[materia[c].prof[d]].restricao[b]=1;
```

```
   materia[c].qaula[d]=materia[c].qaula[d]-2;                    //Marca 2 aulas preenchidas

   return 0;
}

//Verifica se horario pode ser preenchido, avaliando a disponibilidade dos professores e posições livres na tabela, para materias de 2 ou 4 aulas
//*************************************************************************************************************
verifsemana(int a,int b,int c,int d,int e)
{

if((materia[c].qaula[d]>0)&&(semana[e]!=1)&&(horario[c].hor[a]==0)&&(horario[c].hor[b]==0)&&(professor[materia[c].prof[d]].restricao[a]==0)&&(professor[materia[c].prof[d]].restricao[b]==0))
   {
      preenchehorario(a,b,c,d);
      semana[e]=1;
   }

   return 0;
}

//Recebe as posições que foram aceitas para o preenchimento na tabela de horarios, para materias de 3 ou 4 aulas(divididas em 3e1 ou 1e3)
//*************************************************************************************************************
preenchetres(int a,int b,int e,int f,int c,int d)
{
   for(k=0;k<=24;k++)  horario[c].tabela[a][k]=materia[c].nome[d][k];      //Posição na tabela recebe nome da matéria
   for(k=0;k<=24;k++) horario[c].tabela[b][k]=materia[c].nome[d][k];
   for(k=0;k<=24;k++) horario[c].tabela[e][k]=materia[c].nome[d][k];
   horario[c].hor[a]=1;                                    //Posição é marcada como utilizada
   horario[c].hor[b]=1;
   horario[c].hor[e]=1;
   professor[materia[c].prof[d]].restricao[a]=1;                    //Horário do professor é marcado como ocupado
   professor[materia[c].prof[d]].restricao[b]=1;
   professor[materia[c].prof[d]].restricao[e]=1;
   horario[c].nprof[a]=materia[c].prof[d];                     //Recebe o número do professor
   horario[c].nprof[b]=materia[c].prof[d];
   horario[c].nprof[e]=materia[c].prof[d];
   if(f!=0)
   {
      horario[c].nprof[f]=materia[c].prof[d];
      for(k=0;k<=24;k++) horario[c].tabela[f][k]=materia[c].nome[d][k];
      horario[c].hor[f]=1;
      professor[materia[c].prof[d]].restricao[f]=1;
```

```
      materia[c].qaula[d]=materia[c].qaula[d]-4;                    //Marca 4 aulas
preenchidas
   }
   else  materia[c].qaula[d]=materia[c].qaula[d]-3;                  //Marca 3 aulas
preenchidas

   return 0;

}

//Verifica se horario pode ser preenchido, avaliando a disponibilidade dos professores e
posições livres na tabela, para materias de 3 ou 4 aulas
//*****************************************************************************
**************************
veriftres(int a,int b,int e,int f,int c,int d)
{
   if((materia[c].qaula[d]==4 || materia[c].qaula[d]==3)
      &&(horario[c].hor[a]==0)
      &&(horario[c].hor[b]==0)
      &&(horario[c].hor[e]==0)
      &&(horario[c].hor[f]==0 || f==0)
      &&(professor[materia[c].prof[d]].restricao[a]==0)
      &&(professor[materia[c].prof[d]].restricao[b]==0)
      &&(professor[materia[c].prof[d]].restricao[e]==0)
      &&(professor[materia[c].prof[d]].restricao[f]==0 || f==0))
   {
      preenchetres(a,b,e,f,c,d);
   }
   return 0;
}

//Preenche o arquivo de saída com a tabela de horários final gerada, formatando para
sair alinhada
//*****************************************************************************
**************************
imprime()
{  int tam;

   if(qualsaida==1) saida=fopen("dados/saida1.txt","w");
   if(qualsaida==2) saida=fopen("dados/saida2.txt","w");
   if(qualsaida==3) saida=fopen("dados/saida3.txt","w");
   if(qualsaida==4) saida=fopen("dados/saida4.txt","w");
   if(qualsaida==5) saida=fopen("dados/saida5.txt","w");
   if(qualsaida==6) saida=fopen("dados/saida6.txt","w");
   if(qualsaida==7) saida=fopen("dados/saida7.txt","w");
   if(qualsaida==8) saida=fopen("dados/saida8.txt","w");
   if(qualsaida==9) saida=fopen("dados/saida9.txt","w");
   if(qualsaida==10) saida=fopen("dados/saida10.txt","w");

   for(k=1;k<=periodos[0];k++)
```

```
   {
      fprintf(saida,"%d periodo \n\n",periodos[k]);

      for(i=1;i<=4;i++) {
         if(horario[periodos[k]].nprof[i]!=0)                    fprintf(saida,"%02d
",horario[periodos[k]].nprof[i]);
         else fprintf(saida,"   ");
         fprintf(saida,"%s",horario[periodos[k]].tabela[i]);
         tam=25-(strlen(horario[periodos[k]].tabela[i]));
         for(j=1;j<=tam;j++)  fprintf(saida," ");
         if(horario[periodos[k]].nprof[i+4]!=0)                  fprintf(saida,"%02d
",horario[periodos[k]].nprof[i+4]);
         else fprintf(saida,"   ");
         fprintf(saida,"%s",horario[periodos[k]].tabela[i+4]);
         tam=25-(strlen(horario[periodos[k]].tabela[i+4]));
         for(j=1;j<=tam;j++)  fprintf(saida," ");
         if(horario[periodos[k]].nprof[i+8]!=0)                  fprintf(saida,"%02d
",horario[periodos[k]].nprof[i+8]);
         else fprintf(saida,"   ");
         fprintf(saida,"%s",horario[periodos[k]].tabela[i+8]);
         tam=25-(strlen(horario[periodos[k]].tabela[i+8]));
         for(j=1;j<=tam;j++)  fprintf(saida," ");
         if(horario[periodos[k]].nprof[i+12]!=0)                 fprintf(saida,"%02d
",horario[periodos[k]].nprof[i+12]);
         else fprintf(saida,"   ");
         fprintf(saida,"%s",horario[periodos[k]].tabela[i+12]);
         tam=25-(strlen(horario[periodos[k]].tabela[i+12]));
         for(j=1;j<=tam;j++)  fprintf(saida," ");
         if(horario[periodos[k]].nprof[i+16]!=0)                 fprintf(saida,"%02d
",horario[periodos[k]].nprof[i+16]);
         fprintf(saida,"%s",horario[periodos[k]].tabela[i+16]);
         fprintf(saida,"\n");
      }
      fprintf(saida,"\n\n");
   }

   fclose(saida);

   return 0;
}

//Zera toda a tabela de horarios
//**************************************************************************
**************************
zerahorarios()
{
   for(i=0;i<40;i++)
   {
      for(j=0;j<21;j++)
      {
```

```
            horario[i].hor[j]=0;
            horario[i].tabela[j][0]='\0';
            horario[i].nprof[j]=0;
         }
      }
      return 0;
}

//Carrega as matérias dos períodos que serão usados para a geração da tabela de horário
//*****************************************************************************
**************************
carregamaterias()
{
   for(i=1;i<=40;i++) materia[i].qaulamat=0;

   for(i=1;i<=periodos[0];i++)
   {
      if(periodos[i]==1)  periodo=fopen("dados/1.txt","r");
      if(periodos[i]==2)  periodo=fopen("dados/2.txt","r");
      if(periodos[i]==3)  periodo=fopen("dados/3.txt","r");
      if(periodos[i]==4)  periodo=fopen("dados/4.txt","r");
      if(periodos[i]==5)  periodo=fopen("dados/5.txt","r");
      if(periodos[i]==6)  periodo=fopen("dados/6.txt","r");
      if(periodos[i]==7)  periodo=fopen("dados/7.txt","r");
      if(periodos[i]==8)  periodo=fopen("dados/8.txt","r");
      if(periodos[i]==9)  periodo=fopen("dados/9.txt","r");
      if(periodos[i]==10)  periodo=fopen("dados/10.txt","r");
      if(periodos[i]==11)  periodo=fopen("dados/11.txt","r");
      if(periodos[i]==12)  periodo=fopen("dados/12.txt","r");
      if(periodos[i]==13)  periodo=fopen("dados/13.txt","r");
      if(periodos[i]==14)  periodo=fopen("dados/14.txt","r");
      if(periodos[i]==15)  periodo=fopen("dados/15.txt","r");
      if(periodos[i]==16)  periodo=fopen("dados/16.txt","r");
      if(periodos[i]==17)  periodo=fopen("dados/17.txt","r");
      if(periodos[i]==18)  periodo=fopen("dados/18.txt","r");
      if(periodos[i]==19)  periodo=fopen("dados/19.txt","r");
      if(periodos[i]==20)  periodo=fopen("dados/20.txt","r");
      if(periodos[i]==21)  periodo=fopen("dados/21.txt","r");
      if(periodos[i]==22)  periodo=fopen("dados/22.txt","r");
      if(periodos[i]==23)  periodo=fopen("dados/23.txt","r");
      if(periodos[i]==24)  periodo=fopen("dados/24.txt","r");
      if(periodos[i]==25)  periodo=fopen("dados/25.txt","r");
      if(periodos[i]==26)  periodo=fopen("dados/26.txt","r");
      if(periodos[i]==27)  periodo=fopen("dados/27.txt","r");
      if(periodos[i]==28)  periodo=fopen("dados/28.txt","r");
      if(periodos[i]==29)  periodo=fopen("dados/29.txt","r");
                  if(periodos[i]==30)  periodo=fopen("dados/30.txt","r");
                  if(periodos[i]==31)  periodo=fopen("dados/31.txt","r");
                  if(periodos[i]==32)  periodo=fopen("dados/32.txt","r");
                  if(periodos[i]==33)  periodo=fopen("dados/33.txt","r");
```

```
                if(periodos[i]==34)  periodo=fopen("dados/34.txt","r");
                if(periodos[i]==35)  periodo=fopen("dados/35.txt","r");
                if(periodos[i]==36)  periodo=fopen("dados/36.txt","r");
                if(periodos[i]==37)  periodo=fopen("dados/37.txt","r");
                if(periodos[i]==38)  periodo=fopen("dados/38.txt","r");
                if(periodos[i]==39)  periodo=fopen("dados/39.txt","r");

        j=0;
        while(!feof(periodo))
        {
           j++;
           fscanf(periodo,"%d                                                    %d
%s",&materia[periodos[i]].prof[j],&materia[periodos[i]].qaula[j],&materia[periodos[i]].
nome[j]);
           materia[periodos[i]].qmat=j-1;

           reloadqaula[periodos[i]][j]=materia[periodos[i]].qaula[j]; // Faz cópia na
memória para leitora futura mais rápida

           materia[periodos[i]].qaulamat+=materia[periodos[i]].qaula[j]; // Soma a qtde de
aulas de todo o período
        }
        fclose(periodo);
    }
    return 0;
}

//Carrega as matérias que combinam com outras de outros períodos
//**********************************************************************
**************************
carregacombo()
{
    int tem=0;
    for(i=0;i<40;i++)
        for(j=0;j<12;j++)
           combovet[i][j]=0;

    combo=fopen("dados/combo.txt","r");

    i=1;
    j=1;
    while(!feof(combo))
    {
        fscanf(combo,"%d    %d    %d    %d    %d    %d    %d    %d    %d
%d",&combovet[i][1],&combovet[i][2],&combovet[i][3],&combovet[i][4],&combovet[
i][5],&combovet[i][6],&combovet[i][7],&combovet[i][8],&combovet[i][9],&combovet[
i][10]);
        i++;
    }
```

```
   combovet[0][0]=i-2; // Quantidade de combinações

//O Laço abaixo é responsável por verificar quais matérias do combo será utilizado eliminando as que não serão
   for(i=1;i<=combovet[0][0];i++)
   {
      for(j=1;j<=periodos[0];j++) if(combovet[i][1]==periodos[j]) tem=1;
      if(tem) {tem=0; for(j=1;j<=periodos[0];j++) if(combovet[i][3]==periodos[j]) tem=1;}
      if(tem) {tem=0; for(j=1;j<=periodos[0];j++) if(combovet[i][5]==periodos[j]||combovet[i][5]==0) tem=1;}
      if(tem) {tem=0; for(j=1;j<=periodos[0];j++) if(combovet[i][7]==periodos[j]||combovet[i][7]==0) tem=1;}
      if(tem) {tem=0; for(j=1;j<=periodos[0];j++) if(combovet[i][9]==periodos[j]||combovet[i][9]==0) tem=1;}

      if(!tem)
      {
         if(i==combovet[0][0]) combovet[0][0]--;
         else
            {
               for(k=1;k<=10;k++)
                  combovet[i][k]=combovet[combovet[0][0]][k];
               combovet[0][0]--;
               i--;
            }
      }
      else tem=0;
   }
   fclose(combo);

return 0;
}

//Carrega os dados dos professores. (numero, nome e disponibilidade)
//*****************************************************************************
**************************
carregaprofessores()
{
   prof=fopen("dados/prof.txt","r+");

   i=0;
   while(!feof(prof))
   {
      i++;
      qtprof++;
      fscanf(prof,"%d %s",&professor[i].restricao[0],&professor[i].nome);
      fscanf(prof,"%s",&restri);              // le do arquivo as restrições
      for(j=0;j<20;j++)
         {                          // Pega cada posição e distribui nas posições do vetor
```

```
            sprintf(temp,"%c",restri[j]);
            professor[i].restricao[j+1]=atoi(temp);
         }
         for(j=0;j<=20;j++) reloadrestricao[i][j]=professor[i].restricao[j]; //faz cópia na memória
   }
   fclose(prof);

   return 0;
}

//Recarrega os dados da memória após erro (após não conseguir gerar a tabela, retorna os dados iniciais)
//*************************************************************************************************
reload()
{
   tentativas++;       // conta a quantidade de vezes que precisou recarregar

            //mantem o cursor na mesma posição para que os "printf" abaixo substituam um ao outro

   z++;
   if(z==1) {gotoxy(posx,posy);printf("|");}
   if(z==1000) {gotoxy(posx,posy);printf("/");}
   if(z==2000) {gotoxy(posx,posy);printf("-");}
   if(z==3000) {gotoxy(posx,posy);printf("\\");z=-1000;}

         if(z==1||z==1000||z==2000||z==-1000)
         {
// Exibe a contagem regressiva
//*************************************************************************************************
                  time(&now);

                  sec = div(regressivoinicial-now,3600); //Faz a divisão do tempo calculado em segundos, para que seja exibido também em Horas, e minutos
                  horas = sec.quot;
                  sec = div(sec.rem,60);

                  printf("\n\n%02d:%02d:%02d\n\n", horas, sec.quot, sec.rem);

                  if(regressivoinicial==now)
                  {
                           printf("O tempo limite foi alcancado sem resultados... tente novamente!\n");
                           system("pause");

                           tempoesgotado=1;
                  }
```

```
   }

   for(i=1;i<=periodos[0];i++)
   {
      for(j=1;j<=materia[periodos[i]].qmat;j++)
      {
         materia[periodos[i]].qaula[j]=reloadqaula[periodos[i]][j];  //recarrega as qtdes de
aulas dos períodos
      }
   }

   for(i=1;i<=qtprof;i++)
      for(j=0;j<=20;j++)
         professor[i].restricao[j]=reloadrestricao[i][j];  //recarrega as disponibilidades dos
professores

   for(i=0;i<40;i++)
         combovet[i][11]=0;

         return 0;
}

//Verifica quantas matérias de 4 aulas serão divididas em 3e1 ou 1e3 por período
(>20=2 , <20=1)
//**********************************************************************
**************************
tres()
{

   for(i=1;i<=periodos[0];i++)
   {
      if(materia[periodos[i]].qaulamat<20) materia[periodos[i]].tres=1;
      else materia[periodos[i]].tres=2;
   }
   return 0;
}

//Organiza os períodos com as matérias para que seja feito os sorteios de forma a não
repetir o q já foi sorteado
//**********************************************************************
**************************
fastrand()
{
   for(i=1;i<=periodos[0];i++)
   {
      for(j=1;j<=materia[periodos[i]].qmat;j++)
         temppos[periodos[i]][j]=j;

      temppos[periodos[i]][0]=materia[periodos[i]].qmat;
```

```
      tempper[i]=periodos[i];
   }

   tempper[0]=periodos[0];

   return 0;
}
//Preenche combo em qualquer posição sem qualquer regra
//**************************************************************************
**************************

preenchecombopicado()
{
   int numprof;

   numprof=materia[combovet[comboa][1]].prof[combovet[comboa][2]];

   for(j=1;j<=20;j++)
   {
      if(materia[combovet[comboa][1]].qaula[combovet[comboa][2]]>0&&
        horario[combovet[comboa][1]].hor[j]==0&&
        horario[combovet[comboa][3]].hor[j]==0&&
       (horario[combovet[comboa][5]].hor[j]==0||combovet[comboa][5]==0)&&
       (horario[combovet[comboa][7]].hor[j]==0||combovet[comboa][7]==0)&&
       (horario[combovet[comboa][9]].hor[j]==0||combovet[comboa][9]==0)&&
        professor[numprof].restricao[j]==0)
      {
         for(k=0;k<=24;k++)
         {

horario[combovet[comboa][1]].tabela[j][k]=materia[combovet[comboa][1]].nome[com
bovet[comboa][2]][k];

horario[combovet[comboa][3]].tabela[j][k]=materia[combovet[comboa][3]].nome[com
bovet[comboa][4]][k];
            if(combovet[comboa][5]!=0)
horario[combovet[comboa][5]].tabela[j][k]=materia[combovet[comboa][5]].nome[com
bovet[comboa][6]][k];
            if(combovet[comboa][7]!=0)
horario[combovet[comboa][7]].tabela[j][k]=materia[combovet[comboa][7]].nome[com
bovet[comboa][8]][k];
            if(combovet[comboa][9]!=0)
horario[combovet[comboa][9]].tabela[j][k]=materia[combovet[comboa][9]].nome[com
bovet[comboa][10]][k];
         }

horario[combovet[comboa][1]].nprof[j]=materia[combovet[comboa][1]].prof[combovet
[comboa][2]];
```

```
horario[combovet[comboa][3]].nprof[j]=materia[combovet[comboa][3]].prof[combovet
[comboa][4]]; // número do professor responsável pela matéria
        if(combovet[comboa][5]!=0)
horario[combovet[comboa][5]].nprof[j]=materia[combovet[comboa][5]].prof[combovet
[comboa][6]];
        if(combovet[comboa][7]!=0)
horario[combovet[comboa][7]].nprof[j]=materia[combovet[comboa][7]].prof[combovet
[comboa][8]];
        if(combovet[comboa][9]!=0)
horario[combovet[comboa][9]].nprof[j]=materia[combovet[comboa][9]].prof[combovet
[comboa][10]];

        horario[combovet[comboa][1]].hor[j]=1;          // marca posição como
utilizada no horário
        horario[combovet[comboa][3]].hor[j]=1;
        if(combovet[comboa][5]!=0) horario[combovet[comboa][5]].hor[j]=1;
        if(combovet[comboa][7]!=0) horario[combovet[comboa][7]].hor[j]=1;
        if(combovet[comboa][9]!=0) horario[combovet[comboa][9]].hor[j]=1;


professor[materia[combovet[comboa][1]].prof[combovet[comboa][2]]].restricao[j]=1;
// marca posição do professor como ocupado

professor[materia[combovet[comboa][3]].prof[combovet[comboa][4]]].restricao[j]=1;
        if(combovet[comboa][5]!=0)
professor[materia[combovet[comboa][5]].prof[combovet[comboa][6]]].restricao[j]=1;
        if(combovet[comboa][7]!=0)
professor[materia[combovet[comboa][7]].prof[combovet[comboa][8]]].restricao[j]=1;
        if(combovet[comboa][9]!=0)
professor[materia[combovet[comboa][9]].prof[combovet[comboa][10]]].restricao[j]=1;


materia[combovet[comboa][1]].qaula[combovet[comboa][2]]=materia[combovet[comb
oa][1]].qaula[combovet[comboa][2]]-1;  //Reduz o número

materia[combovet[comboa][3]].qaula[combovet[comboa][4]]=materia[combovet[comb
oa][3]].qaula[combovet[comboa][4]]-1;
        if(combovet[comboa][5]!=0)
materia[combovet[comboa][5]].qaula[combovet[comboa][6]]=materia[combovet[comb
oa][5]].qaula[combovet[comboa][6]]-1;
        if(combovet[comboa][7]!=0)
materia[combovet[comboa][7]].qaula[combovet[comboa][8]]=materia[combovet[comb
oa][7]].qaula[combovet[comboa][8]]-1;
        if(combovet[comboa][9]!=0)
materia[combovet[comboa][9]].qaula[combovet[comboa][10]]=materia[combovet[com
boa][9]].qaula[combovet[comboa][10]]-1;
      }
   }
```

```
   if(materia[combovet[comboa][1]].qaula[combovet[comboa][2]]==0)
   {
      for(j=1;j<=temppos[combovet[comboa][1]][0];j++)
         if(temppos[combovet[comboa][1]][j]==combovet[comboa][2]) k=j;

temppos[combovet[comboa][1]][k]=temppos[combovet[comboa][1]][temppos[combov
et[comboa][1]][0]];          // atualiza a variável que armazena os períodos e as matérias
que
      temppos[combovet[comboa][1]][0]--;                    // serão sorteadas,
removendo a matéria que já foi preenchida

      for(j=1;j<=temppos[combovet[comboa][3]][0];j++)
         if(temppos[combovet[comboa][3]][j]==combovet[comboa][4]) k=j;

temppos[combovet[comboa][3]][k]=temppos[combovet[comboa][3]][temppos[combov
et[comboa][3]][0]];          // atualiza a variável que armazena os períodos e as matérias
que
      temppos[combovet[comboa][3]][0]--;                    // serão sorteadas,
removendo a matéria que já foi preenchida

      if(combovet[comboa][5]!=0)                    // mesmo q acima, porém para
"combo" com 3 turmas juntas
      {
         for(j=1;j<=temppos[combovet[comboa][5]][0];j++)
            if(temppos[combovet[comboa][5]][j]==combovet[comboa][6]) k=j;

temppos[combovet[comboa][5]][k]=temppos[combovet[comboa][5]][temppos[combov
et[comboa][5]][0]];          // atualiza a variável que armazena os períodos e as matérias
que
         temppos[combovet[comboa][5]][0]--;                 // serão sorteadas,
removendo a matéria que já foi preenchida
      }

      if(combovet[comboa][7]!=0)                    // mesmo q acima, porém para
"combo" com 4 turmas juntas
      {
         for(j=1;j<=temppos[combovet[comboa][7]][0];j++)
            if(temppos[combovet[comboa][7]][j]==combovet[comboa][8]) k=j;

temppos[combovet[comboa][7]][k]=temppos[combovet[comboa][7]][temppos[combov
et[comboa][7]][0]];          // atualiza a variável que armazena os períodos e as matérias
que
         temppos[combovet[comboa][7]][0]--;                 // serão sorteadas,
removendo a matéria que já foi preenchida
      }

      if(combovet[comboa][9]!=0)                    // mesmo q acima, porém para
"combo" com 5 turmas juntas
      {
         for(j=1;j<=temppos[combovet[comboa][9]][0];j++)
```

```
            if(temppos[combovet[comboa][9]][j]==combovet[comboa][10]) k=j;

temppos[combovet[comboa][9]][k]=temppos[combovet[comboa][9]][temppos[combov
et[comboa][9]][0]];          // atualiza a variável que armazena os períodos e as matérias
que
         temppos[combovet[comboa][9]][0]--;                          // serão sorteadas,
removendo a matéria que já foi preenchida
      }

      combovet[comboa][11]=1;
      combook++;

   }
   else erro=1;

   return 0;
}

//Preenche em qualquer posição sem qualquer regra
//**********************************************************************
**************************
preenchepicado(int per,int pos)
{
   int numprof;

   numprof=materia[per].prof[pos];

   for(j=1;j<=20;j++)
   {
      if(materia[per].qaula[pos]>0&&
        horario[per].hor[j]==0&&
        professor[numprof].restricao[j]==0)
      {
         for(k=0;k<=24;k++)
         {
            horario[per].tabela[j][k]=materia[per].nome[pos][k];
         }
         horario[per].nprof[j]=materia[per].prof[pos];

         horario[per].hor[j]=1;              // marca posição como utilizada no horário

         professor[numprof].restricao[j]=1;      // marca posição do professor como
ocupado

         materia[per].qaula[pos]=materia[per].qaula[pos]-1;  //Reduz o número
      }
   }

   if(materia[per].qaula[pos]!=0) erro=1;
```

```
   return 0;
}

//Após verificar a disponibilidade do professor e das matérias, preenche as matérias
"combo"
//**************************************************************************
**************************
                    //1    2    3    4    5    6    7    8    9    10
preenchecombo(int a,int b,int c,int d,int e,int f,int g,int h,int l,int m, int n, int o)
{
   for(k=0;k<=24;k++)
   {
      horario[c].tabela[a][k]=materia[c].nome[d][k];  // preenche na tabela o nome das
matérias
      horario[e].tabela[a][k]=materia[e].nome[f][k];
      horario[c].tabela[b][k]=materia[c].nome[d][k];
      horario[e].tabela[b][k]=materia[e].nome[f][k];

      if(g!=0&&h!=0)
      {
         horario[g].tabela[a][k]=materia[g].nome[h][k];
         horario[g].tabela[b][k]=materia[g].nome[h][k];
      }
      if(l!=0&&m!=0)
      {
         horario[l].tabela[a][k]=materia[l].nome[m][k];
         horario[l].tabela[b][k]=materia[l].nome[m][k];
      }
      if(n!=0&&o!=0)
      {
         horario[n].tabela[a][k]=materia[n].nome[o][k];
         horario[n].tabela[b][k]=materia[n].nome[o][k];
      }
   }

   horario[c].nprof[a]=materia[c].prof[d]; // número do professor responsável pela
matéria
   horario[e].nprof[a]=materia[e].prof[f];
   horario[c].nprof[b]=materia[c].prof[d];
   horario[e].nprof[b]=materia[e].prof[f];

   horario[c].hor[a]=1;              // marca posição como utilizada no horário
   horario[e].hor[a]=1;
   horario[c].hor[b]=1;
   horario[e].hor[b]=1;

   professor[materia[c].prof[d]].restricao[a]=1;    // marca posição do professor como
ocupado
   professor[materia[e].prof[f]].restricao[a]=1;
   professor[materia[c].prof[d]].restricao[b]=1;
```

```
   professor[materia[e].prof[f]].restricao[b]=1;

   materia[c].qaula[d]=materia[c].qaula[d]-2;  //Reduz o número de aulas nas matérias
   materia[e].qaula[f]=materia[e].qaula[f]-2;

   if(g!=0&&h!=0) // mesmo q acima, porém só executará se houver uma terceira matéria "combo"
   {
      horario[g].nprof[a]=materia[g].prof[h];
      horario[g].nprof[b]=materia[g].prof[h];
      horario[g].hor[a]=1;
      horario[g].hor[b]=1;
      professor[materia[g].prof[h]].restricao[a]=1;
      professor[materia[g].prof[h]].restricao[b]=1;
      materia[g].qaula[h]=materia[g].qaula[h]-2;
   }
   if(l!=0&&m!=0) // mesmo q acima, porém só executará se houver uma quarta matéria "combo"
   {
      horario[l].nprof[a]=materia[l].prof[m];
      horario[l].nprof[b]=materia[g].prof[m];
      horario[l].hor[a]=1;
      horario[l].hor[b]=1;
      professor[materia[l].prof[m]].restricao[a]=1;
      professor[materia[l].prof[m]].restricao[b]=1;
      materia[l].qaula[m]=materia[l].qaula[m]-2;
   }
   if(n!=0&&o!=0) // mesmo q acima, porém só executará se houver uma quinta matéria "combo"
   {
      horario[n].nprof[a]=materia[n].prof[o];
      horario[n].nprof[b]=materia[n].prof[o];
      horario[n].hor[a]=1;
      horario[n].hor[b]=1;
      professor[materia[n].prof[o]].restricao[a]=1;
      professor[materia[n].prof[o]].restricao[b]=1;
      materia[n].qaula[o]=materia[n].qaula[o]-2;
   }

   if(materia[combovet[comboa][1]].qaula[combovet[comboa][2]]==0)  // após preencher todas as aulas de uma determinada
   {                                                               // matéria, efetua alguns ajustes descritos abaixo
      for(i=1;i<=temppos[c][0];i++)
         if(temppos[c][i]==d) j=i;
      temppos[c][j]=temppos[c][temppos[c][0]];        // atualiza a variável que armazena os períodos e as matérias que
      temppos[c][0]--;                                // serão sorteadas, removendo a matéria que já foi preenchida
```

```
      for(i=1;i<=temppos[e][0];i++)
         if(temppos[e][i]==f) j=i;
      temppos[e][j]=temppos[e][temppos[e][0]];
      temppos[e][0]--;

      if(g!=0 && h!=0)                    // mesmo q acima, porém para "combo" com 3
turmas juntas
      {
         for(i=1;i<=temppos[g][0];i++)
            if(temppos[g][i]==h) j=i;
         temppos[g][j]=temppos[g][temppos[g][0]];
         temppos[g][0]--;
      }
      if(l!=0 && m!=0)                    // mesmo q acima, porém para "combo" com 3
turmas juntas
      {
         for(i=1;i<=temppos[l][0];i++)
            if(temppos[l][i]==m) j=i;
         temppos[l][j]=temppos[l][temppos[l][0]];
         temppos[l][0]--;
      }
      if(n!=0 && o!=0)                    // mesmo q acima, porém para "combo" com 3
turmas juntas
      {
         for(i=1;i<=temppos[n][0];i++)
            if(temppos[n][i]==o) j=i;
         temppos[n][j]=temppos[n][temppos[n][0]];
         temppos[n][0]--;
      }
      combook++;                  // +1 "combo" preenchido na tabela
      combovet[comboa][11]=1;           // Marca "combo" (linha do arquivo) como já
preenchido para não ser aceito de novo
   }
   jafoisemana=1;

   return 0;

}
//Após verificar a disponibilidade do professor e das matérias, preenche as matérias
"combo" 1e3 e 3e1
//**********************************************************************
**************************
                                                                  //1     2     3
4    5    6    7    8    9    10
preenchecombotres(int a,int b,int x,int y,int c,int d,int e,int f,int g,int h,int l,int m,int
n,int o)
{
   for(k=0;k<=24;k++)
   {
```

```
      horario[c].tabela[a][k]=materia[c].nome[d][k];  // preenche na tabela o nome das matérias
      horario[c].tabela[b][k]=materia[c].nome[d][k];
      horario[c].tabela[x][k]=materia[c].nome[d][k];
      horario[c].tabela[y][k]=materia[c].nome[d][k];
      horario[e].tabela[a][k]=materia[e].nome[f][k];
      horario[e].tabela[b][k]=materia[e].nome[f][k];
      horario[e].tabela[x][k]=materia[e].nome[f][k];
      horario[e].tabela[y][k]=materia[e].nome[f][k];

      if(g!=0&&h!=0)
      {
         horario[g].tabela[a][k]=materia[g].nome[h][k]; // preenche na tabela o nome das matérias
         horario[g].tabela[b][k]=materia[g].nome[h][k];
         horario[g].tabela[x][k]=materia[g].nome[h][k];
         horario[g].tabela[y][k]=materia[g].nome[h][k];
      }
      if(l!=0&&m!=0)
      {
         horario[l].tabela[a][k]=materia[l].nome[m][k]; // preenche na tabela o nome das matérias
         horario[l].tabela[b][k]=materia[l].nome[m][k];
         horario[l].tabela[x][k]=materia[l].nome[m][k];
         horario[l].tabela[y][k]=materia[l].nome[m][k];
      }
      if(n!=0&&o!=0)
      {
         horario[n].tabela[a][k]=materia[n].nome[o][k]; // preenche na tabela o nome das matérias
         horario[n].tabela[b][k]=materia[n].nome[o][k];
         horario[n].tabela[x][k]=materia[n].nome[o][k];
         horario[n].tabela[y][k]=materia[n].nome[o][k];
      }

   }

   horario[c].nprof[a]=materia[c].prof[d]; // número do professor responsável pela matéria
   horario[c].nprof[b]=materia[c].prof[d];
   horario[c].nprof[x]=materia[c].prof[d];
   horario[c].nprof[y]=materia[c].prof[d];
   horario[e].nprof[a]=materia[e].prof[f];
   horario[e].nprof[b]=materia[e].prof[f];
   horario[e].nprof[x]=materia[e].prof[f];
   horario[e].nprof[y]=materia[e].prof[f];

   professor[materia[c].prof[d]].restricao[a]=1; // marca posição do professor como ocupado
   professor[materia[c].prof[d]].restricao[b]=1;
```

```
professor[materia[c].prof[d]].restricao[x]=1;
professor[materia[c].prof[d]].restricao[y]=1;
professor[materia[e].prof[f]].restricao[a]=1;
professor[materia[e].prof[f]].restricao[b]=1;
professor[materia[e].prof[f]].restricao[x]=1;
professor[materia[e].prof[f]].restricao[y]=1;

horario[c].hor[a]=1;    // marca posição como utilizada no horário
horario[c].hor[b]=1;
horario[c].hor[x]=1;
horario[c].hor[y]=1;
horario[e].hor[a]=1;
horario[e].hor[b]=1;
horario[e].hor[x]=1;
horario[e].hor[y]=1;

if(g!=0&&h!=0) // mesmo q acima, porém para "combo" com 3 turmas juntas
{
   horario[g].nprof[a]=materia[g].prof[h];
   horario[g].nprof[b]=materia[g].prof[h];
   horario[g].nprof[x]=materia[g].prof[h];
   horario[g].nprof[y]=materia[g].prof[h];

   professor[materia[g].prof[h]].restricao[a]=1;
   professor[materia[g].prof[h]].restricao[b]=1;
   professor[materia[g].prof[h]].restricao[x]=1;
   professor[materia[g].prof[h]].restricao[y]=1;

   horario[g].hor[a]=1;
   horario[g].hor[b]=1;
   horario[g].hor[x]=1;
   horario[g].hor[y]=1;
}
if(l!=0&&m!=0) // mesmo q acima, porém para "combo" com 3 turmas juntas
{
   horario[l].nprof[a]=materia[l].prof[m];
   horario[l].nprof[b]=materia[l].prof[m];
   horario[l].nprof[x]=materia[l].prof[m];
   horario[l].nprof[y]=materia[l].prof[m];

   professor[materia[l].prof[m]].restricao[a]=1;
   professor[materia[l].prof[m]].restricao[b]=1;
   professor[materia[l].prof[m]].restricao[x]=1;
   professor[materia[l].prof[m]].restricao[y]=1;

   horario[l].hor[a]=1;
   horario[l].hor[b]=1;
   horario[l].hor[x]=1;
   horario[l].hor[y]=1;
}
```

```
   if(n!=0&&o!=0) // mesmo q acima, porém para "combo" com 3 turmas juntas
   {
      horario[n].nprof[a]=materia[n].prof[o];
      horario[n].nprof[b]=materia[n].prof[o];
      horario[n].nprof[x]=materia[n].prof[o];
      horario[n].nprof[y]=materia[n].prof[o];

      professor[materia[n].prof[o]].restricao[a]=1;
      professor[materia[n].prof[o]].restricao[b]=1;
      professor[materia[n].prof[o]].restricao[x]=1;
      professor[materia[n].prof[o]].restricao[y]=1;

      horario[n].hor[a]=1;
      horario[n].hor[b]=1;
      horario[n].hor[x]=1;
      horario[n].hor[y]=1;
   }

   materia[c].qaula[d]=materia[c].qaula[d]-4;  //Reduz o número de aulas nas matérias
   materia[e].qaula[f]=materia[e].qaula[f]-4;

   if(g!=0&&h!=0)  materia[g].qaula[h]=materia[g].qaula[h]-4;  //Reduz  o  número  de
aulas nas matérias para "combo" c/ 3 turmas
   if(l!=0&&m!=0)  materia[l].qaula[m]=materia[l].qaula[m]-4;  //Reduz  o  número  de
aulas nas matérias para "combo" c/ 4 turmas
   if(n!=0&&o!=0)  materia[n].qaula[o]=materia[n].qaula[o]-4;  //Reduz  o  número  de
aulas nas matérias para "combo" c/ 5 turmas

   for(i=1;i<=temppos[c][0];i++)
      if(temppos[c][i]==d) j=i;                    // atualiza a variável que armazena os
períodos e as matérias que
   temppos[c][j]=temppos[c][temppos[c][0]];              // serão sorteadas, removendo a
matéria que já foi preenchida
   temppos[c][0]--;

   for(i=1;i<=temppos[e][0];i++)
      if(temppos[e][i]==f) j=i;
   temppos[e][j]=temppos[e][temppos[e][0]];
   temppos[e][0]--;

   if(g!=0 && h!=0)
   {
      for(i=1;i<=temppos[g][0];i++)
         if(temppos[g][i]==h) j=i;
      temppos[g][j]=temppos[g][temppos[g][0]];
      temppos[g][0]--;
   }

   if(l!=0 && m!=0)
   {
```

```
      for(i=1;i<=temppos[l][0];i++)
         if(temppos[l][i]==m) j=i;
      temppos[l][j]=temppos[l][temppos[l][0]];
      temppos[l][0]--;
   }

   if(n!=0 && o!=0)
   {
      for(i=1;i<=temppos[n][0];i++)
         if(temppos[n][i]==o) j=i;
      temppos[n][j]=temppos[n][temppos[n][0]];
      temppos[n][0]--;
   }

   combook++;
   combovet[comboa][11]=1;

   materia[combovet[comboa][1]].tres--;                    //diminui a qtde de aulas já
divididas
   materia[combovet[comboa][3]].tres--;

   if(g!=0 && h!=0) materia[combovet[comboa][5]].tres--;
   if(l!=0 && m!=0) materia[combovet[comboa][7]].tres--;
   if(n!=0 && o!=0) materia[combovet[comboa][9]].tres--;

return 0;

}

//Verifica se horario pode ser preenchido, avaliando a disponibilidade dos professores e
posições livres na tabela, para materias de 2 ou 4(2e2, 1e3. 3e1) aulas
//**********************************************************************
**************************
verifcombo(int a,int b,int c,int d)
{
   int numprof;
   numprof=materia[combovet[comboa][1]].prof[combovet[comboa][2]];

   if(materia[combovet[comboa][1]].qaula[combovet[comboa][2]]>0&&
      professor[numprof].restricao[a]==0&&
      professor[numprof].restricao[b]==0&&
      (professor[numprof].restricao[c]==0||c==0)&&
      (professor[numprof].restricao[d]==0||d==0)&&
      horario[combovet[comboa][1]].hor[a]==0&&
      horario[combovet[comboa][1]].hor[b]==0&&
      (horario[combovet[comboa][1]].hor[c]==0||c==0)&&
      (horario[combovet[comboa][1]].hor[d]==0||d==0)&&
      horario[combovet[comboa][3]].hor[a]==0&&
      horario[combovet[comboa][3]].hor[b]==0&&
      (horario[combovet[comboa][3]].hor[c]==0||c==0)&&
```

```
      (horario[combovet[comboa][3]].hor[d]==0||d==0)&&
      horario[combovet[comboa][5]].hor[a]==0&&
      horario[combovet[comboa][5]].hor[b]==0&&
      (horario[combovet[comboa][5]].hor[c]==0||c==0)&&
      (horario[combovet[comboa][5]].hor[d]==0||d==0)&&
      horario[combovet[comboa][7]].hor[a]==0&&
      horario[combovet[comboa][7]].hor[b]==0&&
      (horario[combovet[comboa][7]].hor[c]==0||c==0)&&
      (horario[combovet[comboa][7]].hor[d]==0||d==0)&&
      horario[combovet[comboa][9]].hor[a]==0&&
      horario[combovet[comboa][9]].hor[b]==0&&
      (horario[combovet[comboa][9]].hor[c]==0||c==0)&&
      (horario[combovet[comboa][9]].hor[d]==0||d==0)
   )
   {
      if(c==0)

preenchecombo(a,b,combovet[comboa][1],combovet[comboa][2],combovet[comboa][3]
,combovet[comboa][4],combovet[comboa][5],combovet[comboa][6],combovet[comboa
][7],combovet[comboa][8],combovet[comboa][9],combovet[comboa][10]);
       else

preenchecombotres(a,b,c,d,combovet[comboa][1],combovet[comboa][2],combovet[com
boa][3],combovet[comboa][4],combovet[comboa][5],combovet[comboa][6],combovet[
comboa][7],combovet[comboa][8],combovet[comboa][9],combovet[comboa][10]);
   }

return 0;
}
// Função responsável em mandar para as outras funções, as possibilidades de se gerar a
tabela de horário enviando possíveis posições
//*****************************************************************************
**************************
processacombo()
{
   if(materia[combovet[comboa][1]].qaula[combovet[comboa][2]]>0)
   {
                  jafoisemana=(rand()%2);              // Sorteia pra decidir se vai verificar
pra preencher no dia da semana ou não
                  if(!jafoisemana) verifcombo(1,2,0,0);
                  if(!jafoisemana) verifcombo(3,4,0,0);
                  jafoisemana=(rand()%2);
                  if(!jafoisemana) verifcombo(5,6,0,0);
                  if(!jafoisemana) verifcombo(7,8,0,0);
                  jafoisemana=(rand()%2);
                  if(!jafoisemana) verifcombo(9,10,0,0);
                  if(!jafoisemana) verifcombo(11,12,0,0);
                  jafoisemana=(rand()%2);
                  if(!jafoisemana) verifcombo(13,14,0,0);
                  if(!jafoisemana) verifcombo(15,16,0,0);
```

```
                jafoisemana=(rand()%2);
                if(!jafoisemana) verifcombo(17,18,0,0);
                if(!jafoisemana) verifcombo(19,20,0,0);

      if(materia[combovet[comboa][1]].qaula[combovet[comboa][2]]>0)  erro=1;  // se depois ainda tiver aula pra preencher, deu erro
   }                                                    // vai zerar pra começar a verificar de novo
return 0;
}
// Função Principal de processamento. Faz todo o processo de geração do horário. Após sair dessa função a tabela é considerada gerada
// Chama as funções responsáveis em efetuar a carga dos dados
//*************************************************************************************************
processador()
{
   int per,pos;
        zerahorarios();
   z=0;
   combook=0;
   tentativas=0;
   erro=0;
   todastabelasok=0;
   qtprof=-1;
   qtjanela=0;
        tempoesgotado=0;

   carregamaterias();       //Chamada de funções para carregar os dados para começar a execução
   carregaprofessores();
   carregacombo();
   fastrand();
   tres();

   time(&inicioparcial);        //Marca o inicio da contagem de tempo

   srand ( time(NULL) );   //inicializa a função para geração de números aleatórios

//O programa ficará em loop nessa função até que todas as tabelas, com todos os períodos e matérias passadas estejam devidamente preenchidas
//*************************************************************************************************
   while((todastabelasok==0))
   {
      if(erro==1)       // Caso ocorra algum erro,  os dados serão retornados às suas posições originais para tentar novamente
      {
         reload();
         zerahorarios();
```

```
        tres();
        fastrand();
        erro=0;
        combook=0;
                }
                if(tempoesgotado==1)
                {
                        return 0;
                }

// Ficará em loop até preencher as matérias que combinam com outras, de outros períodos e/ou cursos
//*************************************************************************
**************************
      while(combook!=combovet[0][0])
      {
        valeu=0;
        while(!valeu) // Sorteará qual combo executar, até que sorteie um que ainda não foi preenchido
        {
          comboa=(rand()%combovet[0][0])+1;
          if(!combovet[comboa][11]) valeu=1;
        }

        if(0)
          preenchecombopicado();
        else
          processacombo();     // Chama a função responsável em verificar e preencher os combos

        if(erro==1)         // Se ocorrer erro enquanto preenchendo os combos, recarregará os dados para tentar novamente
        {
          reload();
          zerahorarios();
          tres();
          fastrand();
          erro=0;
          combook=0;
        }

        if(combook==combovet[0][0])
        {
          j=0;
          temp2[0]=0;

          for(i=1;i<=tempper[0];i++)
          {
            if(temppos[tempper[i]][0]>0)
            {
```

```
                j++;
                temp2[j]=tempper[i];
                temp2[0]++;
             }
          }
          for(i=0;i<=temp2[0];i++)
             tempper[i]=temp2[i];
       }

    }
//Armazena em variáveis, os períodos com as posições das matérias que serão utilizadas para o sorteio
//Isso evitará que a matéria que já foi preenchida seja sorteada novamente, agilizando o sorteio
//*************************************************************************
**************************

    tper=(rand()%tempper[0])+1; //sorteia uma posição na variável que armazena os periodos
    tper2=tempper[tper];
    for(i=1;i<=periodos[0];i++)
       if(tper2==periodos[i]) per=i;   //Identifica o período sorteado
    if(temppos[tper2][0]==1)
    {
       tempper[tper]=tempper[tempper[0]]; //Verifica se tem somente uma matéria no período sorteado,
       tempper[0]--;                      // eliminando o período para que não seja sorteado novamente
    }

    tpos=(rand()%temppos[tper2][0])+1;    //Sorteia a matéria do período sorteado acima
    pos=temppos[tper2][tpos];
    temppos[tper2][tpos]=temppos[tper2][temppos[tper2][0]]; //Elimina a matéria sorteada
    temppos[tper2][0]--;                          //para que não seja sorteada novamente

    if(0)
       preenchepicado(periodos[per],pos);
    else
    {
       if(materia[periodos[per]].qaula[pos]==4)     //Verificações para matérias com 4 aulas na semana
       {
          for(i=2;i<=6;i++) semana[i]=0;  //Variável utilizada para evitar que ocorra 4 aulas num mesmo dia da semana
            //Depois de preencher a qtde estipulada de aulas em 3e1 ou 1e3, preencherá as matérias com 4 aulas em 2e2
             verifsemana(1,2,periodos[per],pos,2);
             verifsemana(3,4,periodos[per],pos,2);
```

```
            verifsemana(5,6,periodos[per],pos,3);
            verifsemana(7,8,periodos[per],pos,3);

            verifsemana(9,10,periodos[per],pos,4);
            verifsemana(11,12,periodos[per],pos,4);

            verifsemana(13,14,periodos[per],pos,5);
            verifsemana(15,16,periodos[per],pos,5);

            verifsemana(17,18,periodos[per],pos,6);
            verifsemana(19,20,periodos[per],pos,6);

            if(materia[periodos[per]].qaula[pos]>0) erro=1; //Se não consegui
preencher a tabela ocorrerá erro, retornará
        }                                   //com dados originais e tentará preencher
novamente

        if((materia[periodos[per]].qaula[pos]==3) && erro!=1) //Verificará pra
preencher as matérias com 3 aulas
        {
           veriftres(1,2,3,0,periodos[per],pos);
           veriftres(3,4,5,0,periodos[per],pos);
           veriftres(6,7,8,0,periodos[per],pos);
           veriftres(8,9,10,0,periodos[per],pos);
           veriftres(11,12,13,0,periodos[per],pos);
           veriftres(13,14,15,0,periodos[per],pos);
           veriftres(16,17,18,0,periodos[per],pos);
           veriftres(18,19,20,0,periodos[per],pos);
           veriftres(21,22,23,0,periodos[per],pos);
           veriftres(23,24,25,0,periodos[per],pos);

           if(materia[periodos[per]].qaula[pos]>0) erro=1; // Se erro retornará com dados
originais e tentará novamente
        }

        if((materia[periodos[per]].qaula[pos]==2) && erro!=1) //Verificações para
matérias com 2 aulas na semana
        {
           for(i=2;i<=6;i++) semana[i]=0;

                                verifsemana(1,2,periodos[per],pos,2);       //Verifica
algumas posições para ver a disponibilidade de horário
                                verifsemana(3,4,periodos[per],pos,2);       //para que
ocorra o preenchimento

                                verifsemana(5,6,periodos[per],pos,3);
                                verifsemana(7,8,periodos[per],pos,3);

                                verifsemana(9,10,periodos[per],pos,4);
```

```
                                verifsemana(11,12,periodos[per],pos,4);

                                verifsemana(13,14,periodos[per],pos,5);
                                verifsemana(15,16,periodos[per],pos,5);

                                verifsemana(17,18,periodos[per],pos,6);
                                verifsemana(19,20,periodos[per],pos,6);

            if(materia[periodos[per]].qaula[pos]>0) erro=1;

         }
      }
//Utiliza as variáveis cont1 e cont2 para verificar se a quantidade de matérias preenchidas é igual a quantidade de matérias
//que foram selecionadas para serem preenchidas
//*************************************************************************
**************************
      cont2=0;
      for(i=1;i<=periodos[0];i++)
      {
         cont1=0;
         for(j=1;j<=materia[periodos[i]].qmat;j++)
         {
            if(materia[periodos[i]].qaula[j]==0) cont1++;
         }
         if(cont1==materia[periodos[i]].qmat) cont2++;
      }
//Se a quantidade for a mesma, então todas as matérias foram preenchidas com sucesso
//Deve-se verificar então se há "buracos" entre as aulas, ou como nós chamamos aqui, janela
//*************************************************************************
**************************
      if(cont2==periodos[0])
      {
         gotoxy(posx2,posy2);

                           if(!cjanela)
         {
            janela=0;
            for(i=1;i<=periodos[0];i++)
               for(j=1;j<20;j++)

if((horario[periodos[i]].tabela[j][0]=='\0')&&(horario[periodos[i]].tabela[j+1][0]!='\0')
&& (j!=4 && j!=8 && j!=12 && j!=16)) janela=1;
         }
         else janela=0;
//Se encontrado janela, ocorrerá erro ou não, dependendo da aceitação no começo da execução
//*************************************************************************
**************************
```

```
        if(janela)
        {
                                erro=1;
           qtjanela++;
           if(qtjanela==1) printf("\nFoi encontrado 1 horario com janela...");
                                else printf("\nForam encontrados %d horarios com
janelas...",qtjanela);
        }
        else    // Caso não encontre nenhuma janela, exibirá um resumo com dados como
a qtde de janelas encontradas, tempo de execução
        {

           todastabelasok=1;
           gotoxy(posx,posy);  //mantem o cursor na mesma posição para que os "printf"
acima substituam um ao outro
           time(&fimparcial); //Marca o termino da contagem de tempo
           fprintf(log,"Tabela %d Gerada com Sucesso\n",qs);

           printf("  %d Tabela(s) concluida(s). Faltando %d tabela(s) !\n",qs,qsaida-qs);

           if((fimparcial-inicioparcial)<1) fprintf(log,"O processo demorou menos de 1
segundo\n");
           else
              {
                 sec = div(fimparcial-inicioparcial,3600); //Faz a divisão do tempo
calculado em segundos, para que seja exibido também em Horas, e minutos
                 horas = sec.quot;
                 sec = div(sec.rem,60);

                 fprintf(log,"O processo demorou %d hora(s), %d minuto(s) e %d
segundo(s)\n", horas, sec.quot, sec.rem);
              }
           fprintf(log,"Precisou recarregar %d X...\n",tentativas);
           if(fimparcial-inicioparcial>1)      fprintf(log,"Media      de      %d      X      /
segundo...\n\n",tentativas/(fimparcial-inicioparcial));
           else fprintf(log,"\n");

           time(&inicio2);
           time(&fim2);
           while(inicio2==fim2) time(&fim2); //espera mudar o seg, para q não gere
tabelas iguais

           if(qs==qsaida)
           {
              printf("\rProcesso concluido !                    \n\n");
              time(&fim);       //Marca o termino da contagem de tempo
              if((fim-inicio)<1) printf("O processo demorou menos de 1 segundo\n\n");
              else
              {
```

```
                sec = div(fim-inicio,3600); //Faz a divisão do tempo calculado em segundos, para que seja exibido também em Horas, e minutos
                horas = sec.quot;
                sec = div(sec.rem,60);
                printf("O processo demorou %d hora(s), %d minuto(s) e %d segundo(s)\n", horas, sec.quot, sec.rem);
            }

            printf("\n\nPROFESSORES\n");

            for(i=1;i<=qtprof;i++)
            {
               printf("\nProfessor%d - ",i);
               for(j=1;j<=20;j++)
                  printf("%d",professor[i].restricao[j]);
            }
                                        printf("\n\n");

                                        system("pause");

            fclose(log);
         }
      }
   }
   return 0;
}
// Função Principal. Aqui será exibido um menu para que o usuário entre com os dados que deseja gerar
//*************************************************************************
**************************
main()
{
   clrscr();

   log=fopen("dados/log.txt","w");

   while(op1!=0)
   {
      clrscr();

      printf("\nMENU PRINCIPAL\n\n");

      printf("1 - GERAR TABELA DE HORARIO\n");
      printf("0 - SAIR\n");

      printf("\n-> ");
      scanf("%d",&op1);

      if(op1==1)
```

```
{
   i=0;
   quais=1;
   while(quais!=0 && quais!=99)
   {
      printf("Gerar tabela de quais periodos? (0 pra parar) (99 pra todos) : ");
      scanf("%d",&quais);

      if(quais!=0&&quais!=99) //Se selecionado somente alguns períodos, a variável "períodos" vai recebendo os mesmos
      {
         i++;
         periodos[i]=quais;
         periodos[0]=i;
      }
      else if(quais==99) //Se selecionado todos os períodos, será verificado quais arquivos possuem tamanho>0, e selecionará todos eles.
      {
         fp = open("dados/1.txt", O_RDONLY); if(filelength(fp)>0) {i++;periodos[i]=1;periodos[0]=i;} close(fp);
         fp = open("dados/2.txt", O_RDONLY); if(filelength(fp)>0) {i++;periodos[i]=2;periodos[0]=i;} close(fp);
         fp = open("dados/3.txt", O_RDONLY); if(filelength(fp)>0) {i++;periodos[i]=3;periodos[0]=i;} close(fp);
         fp = open("dados/4.txt", O_RDONLY); if(filelength(fp)>0) {i++;periodos[i]=4;periodos[0]=i;} close(fp);
         fp = open("dados/5.txt", O_RDONLY); if(filelength(fp)>0) {i++;periodos[i]=5;periodos[0]=i;} close(fp);
         fp = open("dados/6.txt", O_RDONLY); if(filelength(fp)>0) {i++;periodos[i]=6;periodos[0]=i;} close(fp);
         fp = open("dados/7.txt", O_RDONLY); if(filelength(fp)>0) {i++;periodos[i]=7;periodos[0]=i;} close(fp);
         fp = open("dados/8.txt", O_RDONLY); if(filelength(fp)>0) {i++;periodos[i]=8;periodos[0]=i;} close(fp);
         fp = open("dados/9.txt", O_RDONLY); if(filelength(fp)>0) {i++;periodos[i]=9;periodos[0]=i;} close(fp);
         fp = open("dados/10.txt", O_RDONLY); if(filelength(fp)>0) {i++;periodos[i]=10;periodos[0]=i;} close(fp);
         fp = open("dados/11.txt", O_RDONLY); if(filelength(fp)>0) {i++;periodos[i]=11;periodos[0]=i;} close(fp);
         fp = open("dados/12.txt", O_RDONLY); if(filelength(fp)>0) {i++;periodos[i]=12;periodos[0]=i;} close(fp);
         fp = open("dados/13.txt", O_RDONLY); if(filelength(fp)>0) {i++;periodos[i]=13;periodos[0]=i;} close(fp);
         fp = open("dados/14.txt", O_RDONLY); if(filelength(fp)>0) {i++;periodos[i]=14;periodos[0]=i;} close(fp);
         fp = open("dados/15.txt", O_RDONLY); if(filelength(fp)>0) {i++;periodos[i]=15;periodos[0]=i;} close(fp);
         fp = open("dados/16.txt", O_RDONLY); if(filelength(fp)>0) {i++;periodos[i]=16;periodos[0]=i;} close(fp);
```

```
            fp = open("dados/17.txt", O_RDONLY); if(filelength(fp)>0)
{i++;periodos[i]=17;periodos[0]=i;} close(fp);
            fp = open("dados/18.txt", O_RDONLY); if(filelength(fp)>0)
{i++;periodos[i]=18;periodos[0]=i;} close(fp);
            fp = open("dados/19.txt", O_RDONLY); if(filelength(fp)>0)
{i++;periodos[i]=19;periodos[0]=i;} close(fp);
            fp = open("dados/20.txt", O_RDONLY); if(filelength(fp)>0)
{i++;periodos[i]=20;periodos[0]=i;} close(fp);
            fp = open("dados/21.txt", O_RDONLY); if(filelength(fp)>0)
{i++;periodos[i]=21;periodos[0]=i;} close(fp);
            fp = open("dados/22.txt", O_RDONLY); if(filelength(fp)>0)
{i++;periodos[i]=22;periodos[0]=i;} close(fp);
            fp = open("dados/23.txt", O_RDONLY); if(filelength(fp)>0)
{i++;periodos[i]=23;periodos[0]=i;} close(fp);
            fp = open("dados/24.txt", O_RDONLY); if(filelength(fp)>0)
{i++;periodos[i]=24;periodos[0]=i;} close(fp);
            fp = open("dados/25.txt", O_RDONLY); if(filelength(fp)>0)
{i++;periodos[i]=25;periodos[0]=i;} close(fp);
            fp = open("dados/26.txt", O_RDONLY); if(filelength(fp)>0)
{i++;periodos[i]=26;periodos[0]=i;} close(fp);
            fp = open("dados/27.txt", O_RDONLY); if(filelength(fp)>0)
{i++;periodos[i]=27;periodos[0]=i;} close(fp);
            fp = open("dados/28.txt", O_RDONLY); if(filelength(fp)>0)
{i++;periodos[i]=28;periodos[0]=i;} close(fp);
            fp = open("dados/29.txt", O_RDONLY); if(filelength(fp)>0)
{i++;periodos[i]=29;periodos[0]=i;} close(fp);
            fp = open("dados/30.txt", O_RDONLY); if(filelength(fp)>0)
{i++;periodos[i]=30;periodos[0]=i;} close(fp);
                        fp = open("dados/31.txt", O_RDONLY);
if(filelength(fp)>0) {i++;periodos[i]=31;periodos[0]=i;} close(fp);
                        fp = open("dados/32.txt", O_RDONLY);
if(filelength(fp)>0) {i++;periodos[i]=32;periodos[0]=i;} close(fp);
                        fp = open("dados/33.txt", O_RDONLY);
if(filelength(fp)>0) {i++;periodos[i]=33;periodos[0]=i;} close(fp);
                        fp = open("dados/34.txt", O_RDONLY);
if(filelength(fp)>0) {i++;periodos[i]=34;periodos[0]=i;} close(fp);
                        fp = open("dados/35.txt", O_RDONLY);
if(filelength(fp)>0) {i++;periodos[i]=35;periodos[0]=i;} close(fp);
                        fp = open("dados/36.txt", O_RDONLY);
if(filelength(fp)>0) {i++;periodos[i]=36;periodos[0]=i;} close(fp);
                        fp = open("dados/37.txt", O_RDONLY);
if(filelength(fp)>0) {i++;periodos[i]=37;periodos[0]=i;} close(fp);
                        fp = open("dados/38.txt", O_RDONLY);
if(filelength(fp)>0) {i++;periodos[i]=38;periodos[0]=i;} close(fp);
                        fp = open("dados/39.txt", O_RDONLY);
if(filelength(fp)>0) {i++;periodos[i]=39;periodos[0]=i;} close(fp);
                        }
        }
        printf("\nDeseja gerar quantas tabelas? (Maximo 10 Tabelas) : ");
        scanf("%d",&qsaida);
```

```
        printf("\nAceitar horario com janela ? (1-Sim 0-Nao) : ");
        scanf("%d",&cjanela);
        printf("\n\nProcessando... ");
        posx=wherex();              //marca a posição do cursor
        posy=wherey();

                        printf("\n\n\n");
                        posx2=wherex();
                        posy2=wherey();

        time(&inicio);

                        regressivoinicial=inicio+300; //5 Minutos

        for(qs=1;qs<=qsaida;qs++)
        {
           if(periodos[0]) processador();  //Se a variavel periodos conter dados, chama-se
a função de processamento principal
                                if(todastabelasok==0) break;
                                qualsaida=qs;
           if(periodos[0]) imprime();       //Logo após, será chamada a função imprime,
responsável em imprimir a saída final
        }
      }
   }

   return 0;
}
```