Nome: Luiz Fernando Pereira de Carvalho
Matrícula: 2018020616
Nome: Yurih Sales Coelho
Matrícula: 2018093171

Trabalho Prático 2

Considerações iniciais

O objetivo do TP2, denominado Calendário, é que o programa possa imprimir um calendário, em modo de lista, após receber do usuário o mês,o ano,o idioma e os eventos que serão lidos a partir de um documento txt. Além disso, outras funções específicas,como uma pesquisa a um determinado dia.

Esse trabalho prático é importante para demonstrar a utilização de TADS, tipos abstratos de dados, que foram tratados em sala de aula.

Antes de pensar no trabalho em si, e aprofundando no que foi proposto, é necessário destacar que o usuário deverá entrar:

- com o mês e o ano escolhido: valor entre 1900 e 2999.
- o idioma: português ou inglês.
- os eventos que devem estar em seu calendário, assim como o tipo desses eventos, que foram classificados como A para aniversários, F para feriados, V para viagem e O para outros.
- nome do arquivo que conterá os dados, que deverá ser um arquivo de extensão txt

A bibliotecas adicionadas para esse trabalho foram:

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
```

A inclusão da biblioteca "calendario.h" se faz necessária pelas definições de funções que serão utilizadas posteriormente

Dessa maneira, foi importante identificar, como primeiro passo, as possíveis funcionalidades que o programa deveria ter, pois elas que iriam formar as TADS. A partir daí, foi possível perceber que a primeira função deveria ser a língua que o usuário deseja para o programa, sendo elas, "Português",selecionada quando o usuário digitar 0, e "Inglês",selecionada quando o usuário digitar 1. É importante ressaltar que, na especificação disponibilizada pelo professor no Moodle, foi indicado utilizar 0 e 1 para os idiomas, respectivamente.

Após a escolha do idioma, o usuário já pode escolher a função que deseja realizar, na linguagem escolhida anteriormente. A ideia do programa poder realizar diferentes funcionalidades levou a criação de um Menu. Tratando sobre a implementação do menu, a

resposta do usuário será levada para uma função denominada "acao" que, possuindo um SWITCH, sua principal função é direcionar a opção do usuário à função correspondente.

A princípio, após receber do usuário o mês,o ano,o idioma e os eventos, o programa deveria imprimir, em modo lista, o calendário correspondente. Porém, pareceu interessante, também, que fosse possível a impressão de um calendário em outro modo, como é visto no dia a dia, um modo que aproximaria o nosso programa da realidade. Assim, é possível não só imprimir o calendário como uma lista, mas também como uma matriz. Junto a isso, foi possível perceber também que, o modo de implementação para a impressão dos eventos no modo lista poderia ser subdividida, já que a criação do calendário nada mais é que a união de vários dias. Desse modo, foi possível criar a função "pesquisa por dia". Essas foram as primeiras funções do programa. O Menu, nesse contexto, é válido pois é a ligação da vontade do usuário com as funcionalidades do programa. Futuramente foi adicionada às funções "Mudar idioma" e "sair", que serão tratadas depois.

A primeira opção do Menu é a impressão do calendário modo 1 (calendário em matriz). A segunda opção do Menu é a impressão do calendário modo 2 (calendário em lista). A terceira opção do Menu é a pesquisa de um determinado dia.

Após essas funções, vieram outras funcionalidades ao programa: a função mudar "idioma", "pesquisa por tipo" e a "sair". A primeira surgiu com o pensamento de que fosse possível ao usuário não só mudar de idioma no início do programa. A segunda, que,na verdade, deveria ser implementada no próximo trabalho prático, assim como a pesquisa por data, foi adiantada para esse trabalho. Isso porque pareceu viável e interessante ter um programa ainda mais útil. A terceira função "sair", era extremamente necessária, pois o usuário ainda não podia deixar o programa quando desejava.

O programa possui essas seis funções anteriormente listadas. Após a utilização de uma determinada função, o programa mostra a opção ao usuário de ver novamente o menu e assim realizar uma ação, escolhido caso o usuário entre com o número zero, realizar outra ação diretamente,sem acesso ao menu, escolhida caso o usuário entre com o número um,e a opção de deixar o programa, escolhida caso o usuário entre com outro número diferente de zero e um.

Logo, conhecendo o programa, será tratado sobre a implementação dos códigos dessas funções e o que cada uma representa na prática, assim como outras estruturas que ajudam a formar a TAD. Abaixo listado, em ordem, as principais funções e estruturas do programa:

- **struct evento:**

Essa struct está presente em todas as funções que o usuário pode escolher. Ela é responsável por identificar as variáveis entregues no arquivo txt e "trazer" para o programa. A struct teve variáveis que foram iniciadas como:

ab.tipo_s: tipo do evento que está no arquivo. Os tipos de eventos podem ser A, F, V, O.

ab.dia_s: dia dos eventos do arquivo.

ab.mes_s: meses dos eventos dos arquivo.

ab.ano_s: ano dos eventos do arquivo.

ab.evento:evento a ser impresso no calendário.

Por exemplo, caso a primeira linha do arquivo txt fosse "V 2 03 2019 fazer x", o programa irá reconhecer essas variáveis, em ordem, como:

ab.tipo_s: V, ou seja, viagem.

ab.dia_s: dia 2.

ab.mes_s: mes 3, ou seja, março.

ab.ano_s: ano de 2019.

ab.evento: fazer x.

```
struct evento
{
    char tipo_s;
    int dia_s;
    int mes_s;
    int ano_s;
    char evento[70];
}ab;
```

Vale destacar a variável char evento[70]. Foi escolhido esse tamanho para a string por se considerar não existir evento com mais caracteres que 70.

É importante citar também que essas variáveis são utilizadas junto com a função fscanf e gets. A fscanf guardará na variável o tipo,dia,mês e ano que foi lido no arquivo txt. A função gets está encarregada de ler o evento do arquivo txt e guardar na variável ab.evento.

```
or (j=0; j<=limite; j++)

   while(fscanf(arquivo, "%c %d %d %d", &ab.tipo_s, &ab.dia_s, &ab.mes_s, &ab.ano_s) != EOF)
   {
       fgets(ab.evento, 70,arquivo);
       if (ab.ano_s == 0)
       {
```

Essas variáveis podem vir seguidas da função fseek para a posição de início. Isso porque é necessário que o programa sempre faça retornar o arquivo ao seu início para uma nova leitura e com diferentes critérios.

```
fflush(stdin);
fseek(arquivo, SEEK_SET, 0);
```

Outra importante função utilizada é a fflush, encarregada de descarregar os buffers.

- **void pesquisa3(int var1, char *nome_arquivo, char *tipo)**

A função pesquisa3 é a função que cumpre o papel de realizar a pesquisa por tipo de evento. Ela recebe como parâmetro as variáveis int var1, que é responsável por dizer a linguagem que o usuário está utilizando(se var1==1, português e se var1==2, inglês), char *nome_arquivo, que trás o nome do arquivo txt que o usuário deseja abrir e char *tipo, que representa o tipo do evento escolhido pelo usuário.

As primeiras linhas da função traz a abertura do arquivo. O programa, após abrir o arquivo txt, realiza uma contagem para encontrar o menor e o maior ano que está neste arquivo pelas variáveis minano e maxano. Foi estabelecido um ano de critério de comparação que, no caso, foi o ano 2000. O while colocado está encarregado de salvar o ano lido na variável ab.ano_s e, caso esta variável seja menor que minano, esse ano será então será atribuído a minano, caso esta variável seja maior que maxano, esse ano será então atribuído a maxano. Para a variável minano, o ano zero não será atribuído.

```
int minano, maxano;
minano=2000;
maxano=2000;
while(fscanf(arquivo, "%c %d %d %d", &ab.tipo_s, &ab.dia_s, &ab.mes_s, &ab.ano_s) != EOF)
{
    if(ab.ano_s<minano & ab.ano_s != 0)minano=ab.ano_s;
    if(ab.ano_s>maxano)maxano=ab.ano_s;
}
```

Após isso, o programa entrará em um processo de busca no arquivo para encontrar a variável entregue pelo usuário. Como existe um critério de impressão, que segue a ordem ano>mes>dia>tipo, foi necessário criar algumas estruturas de for para que essa condição fosse respeitada.

```
for (k=0; k<=maxano; k++)
{
    for (i=0; i<13; i++)
    {
        for (l=0; l<32; l++)
        {
            for (j=0; j<4; j++)
            {
```

É estabelecido, também, a condição para que, caso a variável escolhida pelo usuário for igual a variável lida pelo programa, no caso, tipo[j], será impresso o tipo,ano,mês,ano e o evento. A variável tipo[j], pela estrutura de repetição for, fará uma busca pelo tipo que a pessoa escolheu em ordem de prioridade. Caso o usuário deseja ver os tipos "A" e "O", nessa ordem, o programa guardará "A" em tipo[0] e "O" em tipo[1]. A comparação abaixo garante que só será impresso o tipo que for igual a entrada do usuário.

```
if(tipo[j] == ab.tipo_s && k == ab.ano_s && i == ab.mes_s && l == ab.dia_s){
```

Como foi dito acima, é preciso que tenha uma ordem de impressão, assim, somente será impresso o ano de acordo com a ordem estabelecida pela estrutura do for. O programa então entregará o tipo, dia, mês e ano correspondente e em ordem. Foi acrescentado,

também, uma condição para números menores que 10, para que eles possam aparecer sempre com o zero antes. O arquivo então é fechado e a função acaba.

- **void pesquisa2(int var1, char *nome_arquivo)**

A função pesquisa2 é a função que cumpre o papel de realizar a pesquisa por evento. Ela recebe como parâmetro as variáveis int var1 e char *nome_arquivo, variáveis já tratadas acima.

Logo de início, é pedido ao usuário entrar com o evento desejado. Esse evento será guardado na variável nome_evento. Foi pensado aqui em colocar uma ferramenta que auxiliasse a leitura do evento escolhido pelo usuário. Para isso, foi adicionado ao fim de cada evento a terminação "zzzz". Com isso, foi criado um while com um contador, e,quando os quatros z's forem encontrado pela leitura do while, o contador identifica quantas palavras possuem o evento digitado pelo usuário. Sabendo a quantidade de letras do evento, o programa então irá procurar o evento correspondente no arquivo que possui a mesma quantidade de letras. Encontrado, para que se confirme que o evento encontrado realmente é o evento certo, existe uma comparação de cada elemento do evento encontrado no txt com o evento digitado pelo usuário. Estando correta essa comparação, o evento será impresso para o usuário.

```
while (q < 4)
{
    if (nome_evento[a]== hau[0])
    {
        q++;
    }
    else
    {
        q--;
        if (q<0) {q=0;}
    }
    a++;
}
a = a - 4;
q = 0;
int u=0;


fscanf(arquivo,"%[^\n]s",ab.evento);
for (i=0; i<a; i++)
{
    int j;
    j = i + 1;
    if (ab.evento[j] == nome_evento[i]) q++;
}
if(q<a) q=0;
else
```

Além disso, foi criado um identificador para os casos em que aparecem o zero para o dia, mes ou ano. Por exemplo, quando é lido que o dia é igual a zero do evento escolhido pelo usuário, a mensagem "o evento está marcado para todos os dias" aparece ao usuário.

```
{
    if (ab.dia_s==0)
    {
        if(var1==1)printf("o evento esta marcado para todos os dias\n");
        if(var1==2)printf("this event happens everyday \n");
    }
    if (ab.mes_s==0 & ab.dia_s != 0)
    {
        if(var1==1)printf("o evento acontece todos os meses no dia %d\n", ab.dia_s);
        if(var1==2)printf("this event happens every month on the %dth\n", ab.dia_s);
    }
    if (ab.ano_s == 0 & ab.dia_s != 0 & ab.mes_s != 0)
    {
        if(var1==1)printf("o evento acontece todos os anos em %d/%d\n", ab.dia_s, ab.mes_s);
        if(var1==2)printf("this event happens every year on %d/%d\n", ab.mes_s, ab.dia_s);
    }
    if (ab.ano_s != 0 & ab.dia_s != 0 & ab.mes_s != 0)
    {
        if(var1==1)printf("o evento esta marcado para %d/%d/%d\n", ab.dia_s, ab.mes_s, ab.ano_s);
        if(var1==2)printf("this event happens %d/%d/%d\n", ab.mes_s, ab.dia_s, ab.ano_s);
    }
    q=0;
    u=1;
```

Caso não seja encontrado o evento, a mensagem "o evento nao existe" irá ser impressa

```
    if (u==0)
    {
        if (var1==1)printf("este evento nao existe");
        if (var1==2) printf("this event doesnt exist");
    }
    fflush(stdin);
    fclose(arquivo);
```

O programa então fecha o arquivo e termina essa função.

- **int zero(int var1)**

A função zero, que recebe como parâmetro a linguagem que o usuário está utilizando, é responsável por encerrar o programa através do comando exit(0); Essa função, diferente das outras, foi iniciada como uma função de inteiros, e não como void. Isso, porque era necessário que, quando essa função fosse chamada, o programa finalizasse imediatamente, ao invés de ainda retornar. Essa função é chamada pelo menu através da escolha de número 7.O programa então fecha o arquivo e termina essa função.

```
int zero(int var1){

    if(var1==1){printf("..Fim..");}
    if(var1==2){printf("..Bye..");}
    exit(0);
}
```

- **void pesquisa(int var1, char *nome_arquivo, int ano, int mes,int limite)**

A função pesquisa é a função que cumpre o papel de realizar a pesquisa de um dia, imprimindo os eventos que existem nele. Ela recebe como parâmetro as mesmas variáveis da função pesquisa2 com acréscimo do limite, que fala qual o limite de dias de um certo

mês, int ano, que representa o ano escolhido pelo usuário e int mes, que representa o mês escolhido pelo usuário.

A função começa recebendo do usuário qual o dia ele deseja pesquisar e guarda em uma variável denominada "diaD". Essa função não recebe do usuário dias que estão fora do limite daquele mês.

```
if(var1==1){
    printf("qual dia pesquisar? ");
}
if(var1==2){
    printf("which day do you want? ");
}
scanf("%d",&diaD);
if(diaD>limite && var1==1){printf("dia nao existente nesse mes\n");ijk=1;}
else if(diaD>limite && var1==2){printf("the day doesnt exist on month\n");ijk=1;}
```

Após realizar a abertura do arquivo, a função pesquisa realiza do mesmo artifício que a função pesquisa3: criar critérios para que o programa venha imprimir somente o dia escolhido. Porém, a função pesquisa utiliza de um critério a mais para dias,meses e anos que foram escritos como zero no arquivo txt: fazendo a utilização das funções já descritas, fscanf e gets, até encontrar o fim do arquivo, o programa iguala essas variáveis, que estavam iguais a zero, ao dia ou mês ou ano escolhido, dependendo de qual delas foi escrita como zero. Desse modo, não mais iguais a zero, essas variáveis serão reconhecidas pelo programa como aquelas que não tivessem o número zero.

Logo depois, o programa determina a condição para que seja impresso somente os eventos de um certo tipo, seguindo a ordem AFVO, e, caso o dia escolhido pelo usuário seja igual ao dia que o programa guardou para a variável ab.dia_s.

Foi utilizado, também, um contador para todos os eventos que podem existir de uma certa data.O programa então fecha o arquivo e termina essa função.

```
if(diaD == ab.dia_s)
{
    if(var1==1)
    {
        printf("%d-evento do dia %d: %c", contador,diaD,ab.tipo_s);
        contador++;
    }

    if(var1==2)
    {
        printf("%d-event on the %dth: %c", contador, diaD,ab.tipo_s);
        contador++;
    }
    puts(ab.evento);
}
```

- **void imprime1(int mes,int ano,int var1,int limite,char *nome_arquivo, int a, char *tipo)**

A função imprime1 é a função que cumpre o papel de imprimir os eventos calendário no modo matriz. Ela recebe como parâmetro algumas variáveis já tratadas acima, além da variável "a" e a "tipode". Essas variáveis surgiram da necessidade do usuário poder imprimir, no calendário, todo tipo de evento ou somente um tipo específico de evento e, caso escolha um tipo de evento, qual é esse tipo. Como nesse modo de

impressão de calendário os números aparecem distantes e separados dos eventos, foi conveniente destacar qual era a classificação para cada tipo de variável na tela.

```
int i;
if(var1==1)printf(" tipo/dia/mes/ano/evento\n");
if(var1==2)printf(" type/day/mon/year/event\n");
for(i=1; i<=limite; i++)
```

Indo mais além sobre essa função, existe um for rodando nela que é responsável pela impressão dos dias do mês, de 1 até o limite daquele mês. Junto a isso, ela avalia se o mês e ano corresponde à entrada do usuário é igual ao mês e ano lido e guardado nas variáveis tipo_mes e tipo_anos, caso seja igual, o programa passará para uma próxima avaliação. Para essa avaliação, foi definida uma variável char tipo_evento[4]. Essa variável é responsável pela prioridade (AFVO) da impressão dos eventos. Assim, apostar abstrair as variáveis pela fscanf e gets já citadas, o programa irá estabelecer a condição: o tipo de evento lido é igual a primeira prioridade(A)? Caso sim, ela é igual ao dia em que o for está rodando? Caso sim, será impresso para o usuário o tipo,dia,ano,mês e evento do arquivo txt. Logo após essa primeira avaliação, o programa fará isso para as próximas prioridades até chegar ao fim.

Para esclarecer ainda mais essa situação, se no arquivo estiver escrito: "V 2 03 2019 fazer x", esta descrição só seria impressa se o usuário escolhesse o ano de 2019 e o mês 03. Além disso, ela só seria impressa quando o for estivesse rodando para o dia 2. Ainda assim, seguindo a regra de prioridade, ela só seria impressa logo após existisse um evento no mesmo ano,mês e dia do tipo A ou F.

Essa função realiza o mesmo processo para todos os tipos. Caso o usuário escolha imprimir um tipo específico de evento no calendário(a==2), a função começa avaliar e imprimir então somente aquele tipo, desconsiderando os outros tipos. Caso o usuário não escolha um tipo específico de tipo de evento(a==1), será impresso todos os eventos dos tipos AFVO.

O programa então fecha o arquivo e termina essa função. Vale destacar que essa função não é responsável por imprimir o calendário em si, mas somente os eventos.

- **void imprime2(int mes,int ano,int i,int var1,int limite,char *nome_arquivo, int a, char *tipo)**

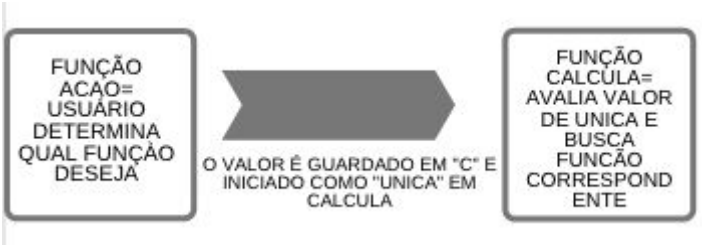
A função imprime2 é a função que cumpre o papel de imprimir os eventos calendário no modo lista. Ela recebe como parâmetro as mesmas variáveis da função imprime1 acrescida da variável int i, que é responsável por determinar o dia em que o calendário em modo lista está rodando. Essa função é muito próxima da função imprime1. Indo mais além, utiliza dos mesmos critérios para imprimir um certo tipo de evento. Então, porque se fez necessário a criação dessa função? Como já foi dito, ela se diferencia da função imprime1 pelo modo de calendário que cada uma está responsável de imprimir seus eventos. Assim, se outrora era necessário que os eventos fossem impressos separados dos dias, agora, o programa deve imprimir cada evento para seu próprio dia. Fazendo uma breve analogia, é como se na função imprime 1 estivessem muitos carrinhos de brinquedos, fora de uma caixa, ordenados somente pelo material(representando os dias) e cor(representando os tipos) e, na função imprime2, além de ordenadas, os brinquedos foram colocadas em caixas diferentes para cada tipo de material. Ou seja, era necessário, para o tipo calendário em lista, uma estrutura for que separasse os eventos, de vários tipos, pelos dias.

Para esse papel, a variável "i" foi primordial para determinar qual era o dia em que o calendário estava rodando.

É importante citar que esssa função também não é responsável por imprimir os dias do calendário em si, mas somente os eventos. Porém, ela deve trabalhar junto com a função que imprime os dias(função calcula).

- **void calcula(int unica,int var1, char *nome_arquivo){}:**

A função calcula é a função que cumpre o papel de realizar todos os cálculos para a montagem do calendário de acordo com o ano e mês escolhido pelo usuário. Além disso, ela também é responsável por imprimir o calendário que será utilizado na função imprime1 e na imprime2, fazer a chamada da pesquisa e pesquisa3. Ela recebe como parâmetro as variáveis int unica,int var1, char *nome_arquivo. Como a função calcula realiza várias funções e está ligada diretamente a várias outras, fez-se necessário a criação de uma variável que diferenciasse qual seria a utilização desse cálculo. Por exemplo, como já foi tratado, a função imprime1 e imprime2 utilizam da função calcula, porém, o programa diferencia a função escolhida pelo usuário através de uma variável, denominada variável única. Essa variável trabalha junto a função acao pois, é nela que o usuário entra com a função desejada.



A função calcula inicia pedindo o usuário o ano e o mês desejado. Eles serão importantes para o cálculo dos dias do calendário. O cálculo é feito da seguinte forma: foi criado a variável "dia" que varia de 1 até o dia limite do mês. Além disso, foi criada a referência "pjan" que representa o dia primeiro de janeiro de 1901. Existe também o cálculo de quantos anos bissextos existiram entre o ano escolhido e 1901, que é guardado na variável "bs". Junto a isso, tendo a referência pjan, é preciso saber a diferença entre o ano do usuário e o ano de referência (1901), representado pela variável cont_ano. Dessa maneira, com todas essas variáveis calculadas, é possível determinar o primeiro dia de cada mês, que é a soma de bs+cont_ano+pjan. Esse resultado é guardado na variável cont_dia.

Foi criado também uma verificação para verificar se o ano escolhido pelo usuário é bissexto. Os anos bissextos, no intervalo determinado pelo professor de ano, são múltiplos de 4. Os anos múltiplos de 100, para serem bissextos, precisam também ser múltiplos de 400. Anos múltiplos de 4 e não múltiplos de 100 também são bissextos. Essa verificação é importante para se somar ao dia de cada mês.

```
}
dia = 1;
pjan = 3;
bs = (ano - 1901)/4;
if (bs < 0)bs = 0;
cont_ano = ano - 1901;
cont_dia = bs + cont_ano + pjan;
if((ano - 1900)%4 == 0) {
    if(ano%100 == 0){
        if(ano%400 == 0)f=1;
        else f=0;
    }else f=1;
}
else f=0;
char d1[15];
```

Junto a isso, é estabelecido um contador de dia e um limite para cada mês que o usuário entra. Logo após, caso a função escolhida pelo usuário seja a função pesquisa, ela já pode ser chamada com seus devidos parâmetros. Para as outras funções, o usuário deve responder se deseja algum tipo específico de tipo sendo impresso (escolhido pelo número 1), ou se não deseja ver somente tipos específicos(escolhido pelo número 2). Caso o usuário deseja ver somente algum tipo sendo impresso, é perguntado qual tipo ele deseja e então é guardado na variável "tipo".

```
if(var!=3 && var!=4 && var!=6){
if(var1==1)printf("\ndeseja um tipo de evento especifico? (1)sim (2)nao: ");
if(var1==2) printf("\nsearch a specific type of event? (1)yes (2)no: ");
scanf(" %d", &a);
if (a==1)
    {
```

Caso a função escolhida pelo usuário seja a função pesquisa3, ela já pode ser chamada com seus devidos parâmetros.

```
if(var==5) pesquisa3(var1,nome_arquivo,tipo);
```

Para a função imprime1 e imprime2 o programa irá imprimir o mês e ano na tela, variando o mês de acordo com o idioma escolhido. Essa função foi feita através da strcpy, que, de acordo com o mês escolhido, copia a string correspondente dentro de uma variável "d1".

```
if(var!=3 && var!=5 && var!=6){
    char jan1 []="janeiro";char jan2 []="january";
    char fev1 []="feveiro";char fev2 []="february";
    char mar1 []="marco";char mar2 []="march";
    char abr1 []="abril";char abr2 []="april";
    char mai1 []="maio";char mai2 []="may";
    char jun1 []="junho";char jun2 []="june";
    char jul1 []="julho";char jul2 []="july";
    char ago1 []="agosto";char ago2 []="august";
    char set1 []="setembro";char set2 []="september";
    char out1 []="outubro";char out2 []="october";
    char nov1 []="novembro";char nov2 []="november";
    char dez1 []="dezembro";char dez2 []="december";
```

Finalmente, para a função imprime1 poderá ser impresso os números do calendário. Antes, é impresso os dias da semana, de acordo com o idioma escolhido e a matriz de números é feita através de um for que criar linhas e colunas do calendário, como uma matriz, imprime pontos para os dias anteriores ao dia 1 daquele mês, pula linhas após o dia de sábado e imprime os números até o limite daquele mês.

imprimir os dias da semana:

```
if(var1==1)printf(" dom seg ter qua qui sex sab");
if(var1==2)printf(" sun mon tue wed thu fri sat");
```

imprimir pontos para dias de outros meses:

```
if (j + 1 <cont_dia & i==0)printf("  . ");
```

Para variáveis abaixo de 10, imprimir 0+número:

```
if (dia <= limite){
    if (dia <10)printf(" 0%d ", dia);
    else printf(" %d ", dia);
    dia++;
}
```

Já para a função imprime2, ao invés de imprimir somente uma vez os dias da semana, ela possui um contador que, de 1 a 7 faz cópia de um determinado dia da semana para a variável "d", de acordo com o idioma escolhido.

```
char seg[]="segunda\0";char seg1[]="monday\0";
char t[]="terca\0";char t1[]="tuesday\0";
char qa[]="quarta\0";char qa1[]="wednesday\0";
char qui[]="quinta\0";char qui1[]="thursday\0";
char sex[]="sexta\0";char sex1[]="friday\0";
char sab[]="sabado\0";char sab1[]="saturday\0";
char dom[]="domingo\0";char dom1[]="sunday\0";
```

Foi definida a condição para que, quando esse contador chegasse no dia da semana corresponde ao número 8, ele reiniciasse e voltasse para o número 1.

```
cont_dia++;
if (cont_dia==8) cont_dia=1;
```

A função imprime 2, então, primeiro faz a contagem do dia, inicia ele como 1 até limite, através de uma estrutura de repetição for, imprime o dia da semana correspondente e, somente depois disso, faz a chamada da função imprime2, já explicitada anteriormente.

```
imprime2(mes, ano, i,var1,limite,nome_arquivo,a,tipo);
```

- **void menu1(int var1){}:**

A função menu1 é extremamente básica: ela avalia a linguagem escolhida pelo usuário e, então, imprime o menu correspondente. O menu apresenta as opções:

1-visualizar o calendário e eventos modo 1
2-visualizar o calendário e eventos modo 2
3-pesquisar data de evento
4-.modo inglês
5-.pesquisar tipo de evento
6-.pesquisar evento
7-.sair

- **void acao(int a,int var1,int b,char *nome_arquivo):**

A função acao é a responsável por receber do usuário o número da função desejada e direcionar para essa função. Ela recebe como parâmetro as variáveis "a", responsável pela função que o usuário deseja realizar do menu, a variável "b", que é importante a realização de sucessivas funções e as já conhecidas variáveis "var1" e "nome_arquivo". É então criada a variável "c", que possui o mesmo valor de "a".

É importante ressaltar a opção do switch para c==4, mudar de idioma. Como já foi dito, foi resolvido criar uma funcionalidade para que usuário pudesse modificar o idioma

quando desejasse. Para isso, caso o usuário escolhesse a o valor de var1==1, que corresponde ao idioma português, para a mudança de idioma nesse caso, a variável var1 então será igualada a 2 e o menu é chamado novamente. O valor digitado no menu é recebido pela variável "a", que só aceita valores dentro do menu, e então a função acao é chamada novamente. Esse laço se repete, para que o usuário possa, a qualquer amostra do menu, trocar a linguagem, até que, pelo próprio menu, o usuário escolha a opção 7(sair) ou escolha sair após a realização de cada ação, quando lhe é perguntado "Voltar para menu(0) tentar novamente(1) Sair(outro numero) ". O procedimento caso o usuário estivesse escolhido var1==2 seria o mesmo, porém, com a troca de valor de var1 sendo igual a 1. Porém, a ideia seria a mesma.

- **void portugueseIngles(int var1, char *nome_arquivo)**

Essa função é responsável por fazer a chamada do menu e da função acao, com seus devidos parâmetros. Ela receberá do usuário qual o tipo de opção do menu desejada e, após a realização, caso não seja chamada a troca de idioma, dá a possibilidade ao usuário de voltar ao menu, continuar realizando ações sem o menu ou sair do programa.

- **main()**

A função main é a função que dá início ao programa. Ela recebe, inicialmente, a linguagem desejada pelo usuário e transmite para as demais funções. Além disso, recebe o nome do arquivo txt que será aberto. Após a chamada de todas as funções, caso a saída imediata não seja escolhida, a própria main finalizará o programa.

```
printf(".Portugues(0)\n.English(1)\n");
printf("opção/option: ");
scanf("%d",&a);
a=a+1;
char nome_arquivo[50];
if(a==1){
    printf("digite o nome do arquivo:");
    scanf("%s", nome_arquivo);
    strcat(nome_arquivo,".txt");
    portugueseIngles(a,nome_arquivo);
    c=0;
    printf("\n..fim..");
```

Possíveis caminhos do programa

português>imprimir modo 1>imprimir modo 2>sair

português>imprimir modo 2>imprimir modo 2>imprimir modo 1>sair

português>pesquisa por data>sair

português>pesquisa por tipo>pesquisa por evento>imprimir pdf>sair

português>pesquisa por evento>sair

português>imprimir pdf>sair

portuguese>sair

ingles>imprimir modo 1>sair

inglês>imprimir modo 2>imprime modo 2>sair

inglês>pesquisa por data>imprime modo 1>imprime modo 2>sair

inglês>pesquisa por tipo>sair

inglês>pesquisa por evento>sair

inglês>imprimir pdf>pesquisa por tipo>pesquisa por data>sair

inglês>sair

Considerações finais

A escrita desse programa foi bastante importante para a ampliação dos conhecimentos da dupla participante para a criação de TADS, que, anteriormente, ainda não eram utilizadas na criação de programas. Além disso, aumentou as habilidades dos estudante em relação a leitura de arquivos txt.

Documentação específica do Trabalho Prático 3

A partir de agora, será tratado algumas questões específicas do trabalho prático 3. Apesar do fato que somente um documento e programa será enviado, pareceu importante tratar especificamente de algumas funções para o trabalho prático 3. Isso não significa que uma abordagem exclui a outra, pois, apesar de, por exemplo, a função de criar pdf, à princípio, não fazer parte do trabalho prático 2, foi autorizado pelo monitor a entrega de somente um documento, com a restrição de que seja enviada duas vezes. A ideia dessa parte do trabalho é destacar o quão diferente foi a proposta de criar um arquivo pdf.

À princípio, a ideia era a utilização de alguma outra linguagem, como HTML, que facilitaria a criação do pdf. Não era de conhecimento que existia uma biblioteca em c para isso. Com pesquisas e discussões com outros colegas de classe, descobriu-se uma biblioteca "PDFGEN", encontrada no GITHUB-AndreRenaud-, que faz a criação de arquivos em pdf, além de adicionar imagem, linhas, formas geométricas e textos aos arquivos pdf.

O objetivo do TP3, é que o programa possa imprimir um calendário em pdf, junto com uma imagem e ícones para os tipos de eventos. Além disso, outras funções específicas do tp3, que foram utilizadas no tp2, são: consulta de eventos por tipo e por data.

As bibliotecas utilizadas foram:

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "pdfgen.h"
```

A inclusão da biblioteca "pdfgen.h" se faz necessária pelas condições estabelecidas pelo próprio criador da função gerar pdf.

Para o menu foi adicionada o tipo modo pdf, correspondente ao número 7 (a função sair mudou para 8).

Abaixo listado, em ordem, as principais funções e estruturas do programa:

- **void callpdf (int var1, char *nome_arquivo)**

Essa função é responsável por receber do usuário o mês e ano, assim como a opção de visualizar comente o calendário, calendário+mês posterior+mês anterior,

calendário+mês posterior. Ela recebe como parâmetro as variáveis int var1 e char *nome_arquivo, variáveis já tratadas anteriormente. Após receber essas informações do usuário, ela faz a chamada da função que realizará a impressão do calendário.

- **void pdf(int mes, int ano, int v, char *nome_arquivo, int var1)**

Essa função é responsável por gerar o pdf, modificar o pdf e salvar o pdf. Ela recebe como parâmetro as variáveis int var1" , char *nome_arquivo, int ano, int mes, int var1 e int v. As últimas quatro variáveis foram recebidas da função callpdf e fazem relação, respectivamente, à: ano escolhido pelo usuário, mês escolhido pelo usuário, idioma escolhido pelo usuário (var1==1 português, var1==2 inglês) e modo de impressão do pdf.

A primeira estrutura que aparece é uma estrutura original, definida pelo criador da biblioteca pdfgen. Elá é responsável por trazer as informações do arquivo pdf criado.

```
struct pdf_info info =
{
    .creator = "My software",
    .producer = "My software",
    .title = "My document",
    .author = "My name",
    .subject = "My subject",
    .date = "Today"
};
```

Logo após, é criado o pdf e as funções de adicionar texto e imagem são chamadas.

```
struct pdf_doc *pdf = pdf_create(PDF_A4_WIDTH, PDF_A4_HEIGHT, &info);
pdf_set_font(pdf, "Helvetica-Bold");
pdf_append_page(pdf);
pdf_add_jpeg(pdf,NULL,0,0,600,900,"caled.jpg");
```

A fonte do texto, escolhida como "Helvetica-Bolt", já foi definida em "pdfgen.h" e "pdfgen.c". A imagem adicionada está junto a pasta do programa.

O próximo passo foi definir coordenadas x,igual a 25, e y, igual a 660, para imprimir, no pdf, alguns valores desejados. Logo após, há uma série de contas semelhante a função calcula, já tratada, que trará o primeiro dia e o limite de dias de um certo mês. Dependendo da entrada de mês que o usuário escolha e o idioma, será impresso o mês no pdf. Além disso, o ano escolhido pelo usuário também está sendo impresso, porém, com a utilização da função sprintf. A função sprintf foi usada nesse programa para receber do leitor a variável de ano e transformar em uma string. Isso foi necessário pois, para imprimir no pdf, é preciso que a variável esteja em formato de texto, ou seja, uma string.

```
sprintf(s_ano, "%i", ano);
pdf_add_text(pdf, NULL, s_ano, 40, 450, 650, PDF_WHITE);
```

Depois disso, é então impresso no pdf os dias da semana. Indo mais além, faltava determinar uma série de equações e estruturas para que, no pdf, os dias de um certo mês fossem impressos. Para isso, foi usado a mesma lógica de impressão do calendário em modo 1, que é utilizar de uma estrutura de repetição for para criar linhas e colunas do calendário. Porém, onde lá determinava imprimir na tela do usuário, aqui, será impresso no pdf. Para o primeiro for, para o número de semanas, foi definida uma condição para que, após terminar a leitura de uma linha, a coordenada y diminuísse um certo valor.

```
for (i=0; i<6; i++)
{
    if (i==0)
        y = y - 80;
    else
        y = y - 93;
```

O segundo for, o número de dias, foi definido também um valor para se somar na variável x para a mudança de dias. Para os dias anteriores ao dia 1 de um certo mês, será impresso espaço(" "). Para os dias do mês escolhido pelo usuário, será impresso no pdf os dias da semana, mudando a variável x para a mudança de dia e a variável y para a mudança de semana.\

```
x = 25;
for (j=0; j<7; j++)
{

    if ((j + 1 <cont_dia) & (i==0))
    {
        pdf_add_text(pdf, NULL, "     ", 15, x, y, PDF_WHITE);

        x = x + 81;

    }

    else


if(j==0||j==6)
    pdf_add_text(pdf, NULL, b, 15, x, y, PDF_BLUE);
else
    pdf_add_text(pdf, NULL, b, 15, x, y, PDF_WHITE);
x = x + 81;
```

A função sprintf ainda não era conhecida, então, a conversão de inteiro para string foi feita pela soma de valores utilizando a tabela ASCii. O primeiro dia de um certo mês foi somado a 48 e transformado em string, que corresponde ao valor do dia na tabela ASCii.

```
diapdf = dia + 48;
w=diapdf;
b[0]=w;
matriz[0][dia] = x;
matriz[1][dia] = y;
if (dia <10)
```

Para os dias com dois algarismos, foi necessário separar esses dois algarismos, através do resto e divisão por 10, e imprimir no pdf como

```
e = (dia/10)+48;
u = (dia%10)+48;
w=e;
c[0]=w;
w=u;
c[1]=w;
```

As coordenadas x e y foram salvas nas variáveis "matriz ", nas posições 0 e 1 respectivamente. As próximas condições avaliam se aquele dia é sábado ou domingo, para imprimir em cor azul, e os restos dos dias em branco.

```
if(j==0||j==6)
    pdf_add_text(pdf, NULL, c, 15, x, y, PDF_BLUE);
else
    pdf_add_text(pdf, NULL, c, 15, x, y, PDF_WHITE);
x = x + 81;
```

Após isso, é necessário imprimir os eventos no pdf. Existe então a opção para o usuário escolher um tipo específico de evento do arquivo txt. Abrindo o arquivo pelas funções fscanf e gets, ele irá imprimir em ordem de prioridade de tipo, AFVO, os eventos relacionados a cada dia do mês que está representado pela letra i. Com a estrutura for rodando de 1 até o limite de um certo mês, é possível passar por todos dias da semana. Como cada coordenada x e y foi salva em uma matriz, agora, essas coordenadas são acionadas para adicionar um ícone a cada tipo de evento. As imagens correspondentes aos ícones estão na pasta do arquivo.

```
if (j==0)
{
    pdf_add_jpeg(pdf,NULL,x,y-1,8,8,"bolo.jpg");
    pdf_add_text(pdf, NULL, ab.evento, 7, x+7, y, PDF_BLUE);
}
if (j==1)
{
    pdf_add_jpeg(pdf,NULL,x,y-1,8,8,"feriado.jpg");
    pdf_add_text(pdf, NULL, ab.evento, 7, x+7, y, PDF_RED);
}
if (j==2)
{
    pdf_add_jpeg(pdf,NULL,x,y-1,8,8,"aviao.jpg");
    pdf_add_text(pdf, NULL, ab.evento, 7, x+7, y, PDF_GREEN);
}
if(j==3)
    pdf_add_text(pdf, NULL, ab.evento, 7, x+7, y, PDF_WHITE);
y=y-10;
```

O calendário principal então está pronto.

Era necessário, então, montar equações para caso o usuário quisesse imprimir o mês anterior e posterior no pdf. Para isso, era preciso calcular o primeiro dia do mês anterior e posterior ao escolhido pelo usuário. O acont_dia é responsável pelo primeiro dia do mês anterior e o pcont_dia responsável pelo primeiro dia do mês posteior. O mês posterior, pmes, e o mês anterior, ames, foi obtido simplesmente somando ou subtraindo um.

```
if (limite%10==0)
    acont = cont_dia +4;
else
    acont= cont_dia + 5;
if (mes==2)
    acont=cont_dia+4;
if (mes==8) acont=cont_dia + 4;
if (mes==3)
    acont= cont_dia +7 -f;


pmes = mes + 1;
ames = mes - 1;
```

Caso o mês escolhido pelo usuário fosse dezembro, foi criado a condição para que o mês posterior seja janeiro. Caso o mês escolhido pelo usuário fosse janeiro,oi criado a condição para que o mês posterior seja dezembro.

```
if (pmes>12)
{
    pmes = 1;
    pcont = cont_dia+3;
}
if (ames == 0)
{
    ames = 12;
    acont = cont_dia+4;
}
```

Calculado o mês anterior e posterior. Para o caso em que o usuário quiser somente o mês posterior, igualou-se a variável de escolha do modo "v" a zero. O programa reconhecerá que não existe mês igual a zero e não irá imprimir.

```
if(v==2)
    ames=0;
```

Redefinindo as variáveis x e y, o processo para printar o outros meses foi exatamente igual ao do mês escolhido pelo usuário. Somente a soma de valores a x e y mudaram, pelo tamanho do mini calendário que deveria ser impresso

```
x=450;
y=780;
```

O ano, os meses e os dias das semanas são impressos com suas coordenadas já ajustadas e, por fim, os números dos dias da semana, que é a mesma função para com o calendário cujo mês foi escolhido pelo usuário,mas com as coordenadas também ajustadas

Observação: o mês posterior está 430 posições à frente do mês anterior.

```
if (ames==1)
    n=430;
else
    n=0;
```

```
if (i==0)
    y = y - 7;
else
    y = y - 12;
x = 450;
a=x-430;
for (j=0; j<7; j++)
{
```

O pdf então é salvo e a função acaba.

Considerações finais

A escrita dessa parte específica do programa, além de trabalhosa, foi bastante importante para a ampliação dos conhecimentos da dupla participante em relação à utilização de novas bibliotecas, assim como a criação de um arquivo em pdf. Vale destacar que uma ótima opção seria, também, aprender outra linguagem que realizasse esse mesmo serviço. A escolha da biblioteca utilizada e da linguagem c pela dupla se deu pela facilidade e pelo conhecimento já adquirido dessa linguagem, que é mais familiar.