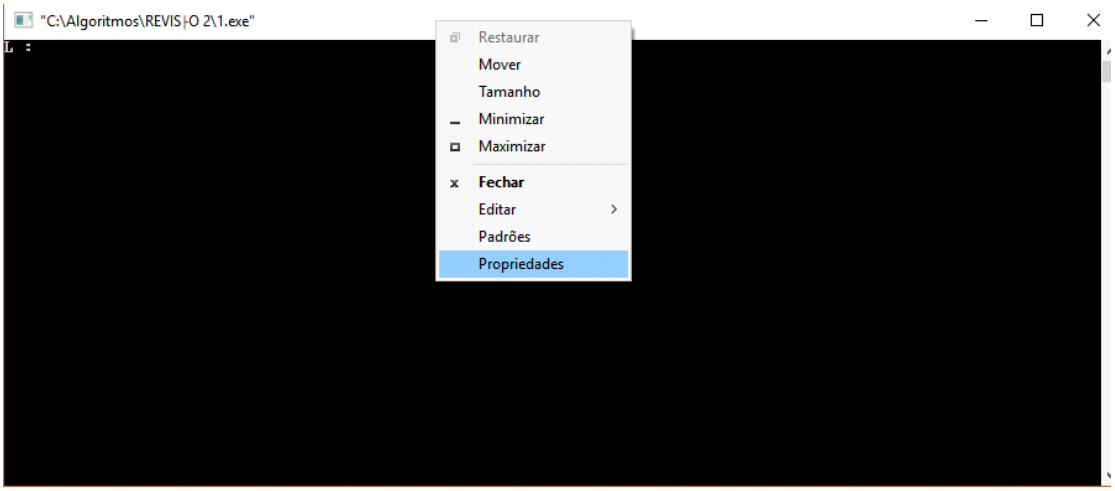
RELATÓRIO - APS ALGORITMOS 2

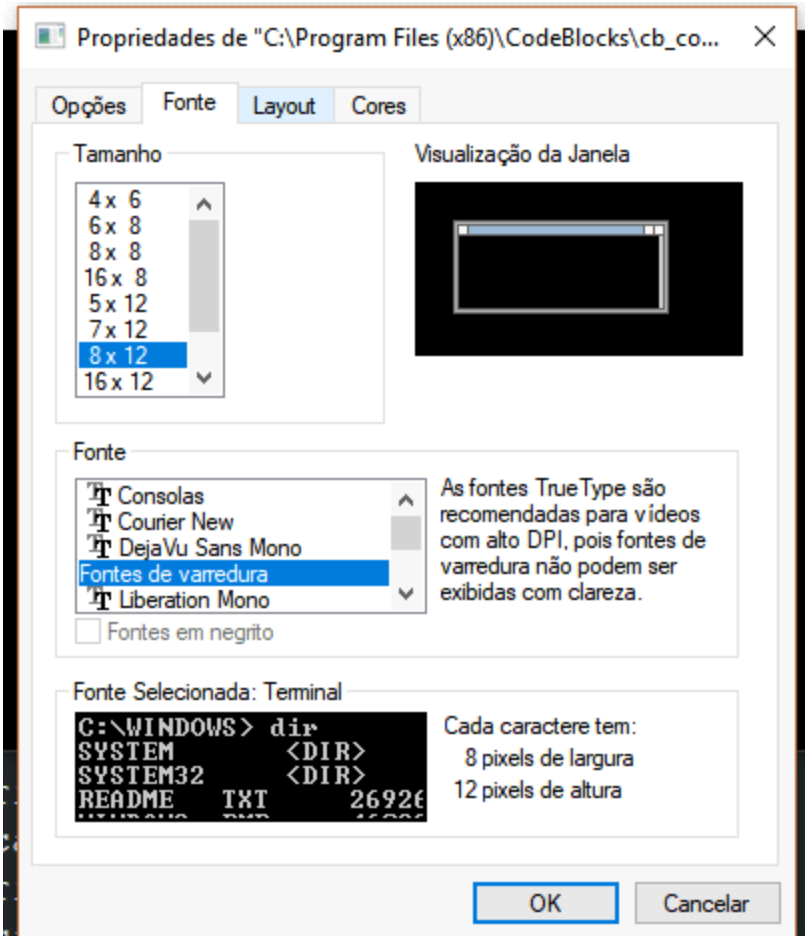
Einheit - Last Chapter

Alunos: José Otávio Bremm - 2003295;
Lucas Trizzini - 2002450;

A APS é um jogo de texto em C, no qual uma história é contata e o jogador tem como função o rumo da historia, contendo decisões onde cada escolha gera uma consequencia no decorrer da história. O jogo é totalmente interativo, basta executalo e irá explicar por sí só. *Einheit* em alemão significa unidade, união, compania, o que é apresentado no jogo na forma de conjunto dos personagens.

Para o jogo ser executado com exito, se faz necessário a mudança da fonte do executavel do CodeBlocks.





Utilizando a fonte "Fontes de varredura(8x 12)" os ASCII presentes no jogo seram lidos corretamente. Quando o jogo iniciar, espere a tela ficar fullscreen, após isso , aperte qualquer tecla para iniciar o jogo.

Lucas Trizzini - História e desenvolvimento criativo do jogo.
José Otávio Bremm - Desenvolvimento do jogo em C.

Conteúdo da disciplina usado dentro do jogo:

- Arquivos;
- Funções;
- Ponteiros;
- Passagem de parametro;
- Automatos;

O codigo é implementado e funciona totalmente em cima de automatos e arquivos, os arquivos servem para contar a história de forma grafica, como os textos e artes em ASCII. Você navega entre as telas de história e graficas com a função Getch, usando os caracteres corretos para cara tela para navegar. Também pode se encontrar variadas funções, que puxam e buscam uma as outras, a maioria é feita para a organização do código e divisão das partes que possuem a história. Em momentos é gerado números aleatórios para fazer decisões randomicas entre as telas, por exemplo: "Você vai conseguir?, se o rand

for 1, sim, se for 2, não". A dificuldade do jogo é regulada da forma que você joga, dependendo de suas decisões a dificuldade aumenta, ou se mantem. Alguns trechos importantes do código serão exemplificados a seguir.

Automato:

Parte de código que declara o funcionamento de um automato presente no código, especificamente o automato da cidade do jogo, possibilitanto a locomoção do personagem entre o código.
O automato funciona com vetor de funções, você define funções para cada comodo listado acima com "ENUM {};"

```
int f_alquimista(void);
int f_castelo(void);
int f_ferreiro(void);
int f_city(void);

enum cidadela {alquimista, castelo, ferreiro, city};

int cidade(void)
{
	setlocale(LC_ALL, "");
	int (*cidadela[4])(void);

	enum cidadela estado = city;

	cidadela[alquimista] = f_alquimista;
	cidadela[castelo] = f_castelo;
	cidadela[ferreiro] = f_ferreiro;
	cidadela[city] = f_city;

	do
	{
		estado = cidadela[estado]();
	}
	while(estado != 'city');

	return 0;
```

Arquivos:

Neste trecho do código é aberto um dos vários arquivos do jogo, você define um ponteiro FILE para o arquivo, e imprime char por char do arquivo presente no computador até que o char seja diferente de EOF(end of file).

```
FILE *taverna1;
	char a;
```

```
char state;
setlocale(LC_ALL, "");
do
{
	system("cls");
	vod();
	taverna1 = fopen("historia\\taverna.txt","r");
	while(fscanf(taverna1,"%c",&a) != EOF)
	{
		printf("%c",a);
	}
	state = getch();
```

Passagem de parametros:

Neste trecho do código está sendo representada uma das passagem de parametros utilizada no código, o qual a função manda para a função batalha a vida do monstro, o dano do monstro e o nome do monstro, o qual será utilizada para complementar e funcionalizar a batalha. Os pontos antes e depois de cada algoritmo servem para mostrar que são algoritmos diferentes, e que existem mais coisas acima deles, ou abaixo, não estão completos, mas servem para exemplificar a situação.

```
.
.
.
while(fscanf(go,"%c",&a) != EOF)
 {
	printf("%c",a);
 }
 getch();
 batalha(vmonstro,dmonstro,monstro);

}
.
.
.
void batalha (int vmonstro, int dmonstro, char monstro[])
{
	extern int vida;
	extern int dano;
	extern int ouro;
	int x, y,z;
	float t;
	static int v=0;
	srand(time(NULL));
	if(v == 0)
	{
```

```
system("cls");
.
.
```

Também se encontra no código váriaveis globais estáticas, que são usadas por algumas funções do código, essas variaveis são responsaveis por armazenar o ouro, a vida e o dano dos personagens.

```
extern int vida;
extern int dano;
extern int ouro;
```

O jogador começa com 25 de ouro, 50 de dano e 100 de vida.

Ponteiros:

```
FILE *go;
```

Ponteiros são utilizar juntamente dos arquivos, quando se cria um arquivo utilizando FILE, você cria um ponteiro de FILE, para utilizar o arquivo que será usado.

Funções:

O código abaixo é uma das funções utilizada no jogo, ela é responsavel por mostrar a vida, o ouro e o dano do personagem durante a execução de cada tela do jogo, ela é apenas utilizada de forma grafica dentro do jogo

```
void vod (void)
{
    //USADO PARA MOSTRAR A VIDA, OURO E DANO DO PERSONAGEM NA PARTE
SUPERIOR DIREITA DA TELA
    extern int dano;
    extern int ouro;
    extern int vida;
    printf(ANSI_COLOR_LIGHT_MAGENTA "\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t/-------------/\n"
ANSI_COLOR_RESET);
    printf(ANSI_COLOR_RED "\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t %c " ANSI_COLOR_RESET, 3);
    printf("%d /",vida);
    printf(ANSI_COLOR_BLUE " %c " ANSI_COLOR_RESET, 6);
    printf("%d\n", dano);
    printf(ANSI_COLOR_YELLOW "\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t %c " ANSI_COLOR_RESET,
4);
    printf("%d \n", ouro);
    printf(ANSI_COLOR_LIGHT_MAGENTA "\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t/-------------/\n"
ANSI_COLOR_RESET);
}
```

Em algumas partes do código pode ser encontrado números randomicos, para se selecionar opções randomicas, como o acontecimento de uma ação em uma batalha, ou o dano acima de alguma coisa, como no código abaixo, o qual é necessario abrir uma fechadura, a fechadura é aberta se a vida da fechadura é reduzida a 0.

```
do
    {
        if(lockpick > 0)
        {
        x = 30 + rand() % (40 - 30 + 1);
        vida = (vida - x);
        printf("\t\t\nUtiliza de um de seus lockpicks para destrancar a porta!\n");
        getch();

        if(vida > 0 && lockpick > 0)
        {
            lockpick--;
            printf("Uma das lockpicks quebrou, restam %d!\n",lockpick);
        getch();
        }
        else if (vida <= 0 && lockpick > 0)
        {
            printf("A porta foi destrancada com sucesso!");
              getch();
            return fuga_prisao();
        }
        }
        else if(lockpick <= 0 && vida > 0)
  {
            setlocale(LC_ALL, "");
            printf("\nSariel nao conseguiu destrancar a porta. Os lockpicks que carregava
consigo eram gastos e velhos.");
            printf("\nThoradir vai precisar usar a forca para quebrar a porta.\n");
            getch();
            return dano_sair_prisao();
        }
              state = getch();

    }
    while(state == '1');
```

Referências à conteúdos externos:

No código também podem ser encontradas partes de código não presentes na disciplina, como a implementação de cor em C, som em C, a mudança da cor do console, limpar a tela, e a maximização do executavel na tela do computador.

Cor:

```
#define ANSI_COLOR_BLUE                 "\x1B[94m"
#define ANSI_COLOR_RESET                "\x1b[0m"
                        .
                        .
                        .
  while(fscanf(algdom,"%c",&b) != EOF)
  {
        printf(ANSI_COLOR_BLUE "%c" ANSI_COLOR_RESET, b);
  }
```

Que pode ser encontrado no link a seguir :
https://github.com/shiena/ansicolor/blob/master/README.md

Som:

```
Beep(600,200);
Beep(900,150);
Beep(1000,150);
Beep(600,150);
```

Que pode ser encontrado no link a seguir:
https://cboard.cprogramming.com/cplusplus-programming/31905-beep.html

Mudança de cor no console:

```
SetConsoleTextAttribute(GetStdHandle(STD_OUTPUT_HANDLE),3);
```

Que pode ser encontrado no link a seguir:
https://stackoverflow.com/questions/30645675/how-to-change-colour-of-a-specific-output-on-console-in-c

Abrir executavel em tela cheia:

```
keybd_event(VK_MENU,0x36,0,0);
keybd_event(VK_RETURN,0x1C,0,0);
keybd_event(VK_RETURN,0x1C,KEYEVENTF_KEYUP,0);
keybd_event(VK_MENU,0x38,KEYEVENTF_KEYUP,0);
system("pause > nul");
```

Que pode ser encontrado no link a seguir:
https://www.clubedohardware.com.br/forums/topic/966166-abrie-em-excutável-em-tela-cheia/

Limpar a tela:

```
system("cls");
```

essa função está contida na biblioteca <windows.h>.
http://personalizandoc.blogspot.com/2012/11/comando-systemcls-limpando-tela.html

Imprimir arquivo de acordo com as coordenadas que você deseja:

SetConsoleCursorPosition(GetStdHandle(STD_OUTPUT_HANDLE),(COORD){j,i});

Que pode ser encontrado no link a seguir:
https://stackoverflow.com/questions/15770853/how-to-use-setconsolecursorposition-func