

**USP - ICMC - SSC
SSC 0501 - 1o. Semestre 2010**

**Disciplina de
Introdução à Ciência da Computação
ICC 1 - Teoria**

Prof. Fernando Santos Osório

Email: fosorio [at] { icmc. usp. br , gmail. com }

Página Pessoal: <http://www.icmc.usp.br/~fosorio/>

Web - WIKI ICMC: <http://wiki.icmc.usp.br/index.php/SSC-501>

PAE: Gustavo Pessin (Dout. CCMC) / [Http://pessin.googlepages.com/](http://pessin.googlepages.com/)

Email: pessin [at] { icmc.usp.br , gmail.com }

Monitor: Matheus Lin Alvarenga (EC) / [Http://matheuslin.wordpress.com/](http://matheuslin.wordpress.com/)

Email: matheus.lin [at] gmail.com

Aula 14s

1

Maio 2010

Linguagem de Programação “C”

Agenda:

- **Modularização de Programas:**
 - Usando Bibliotecas de Funções
 - > Rotinas Gráficas: WinBGI
 - > Rotinas de Pilha (Stack)
 - > Rotinas de Lista Encadeada

High Effective Language Learning

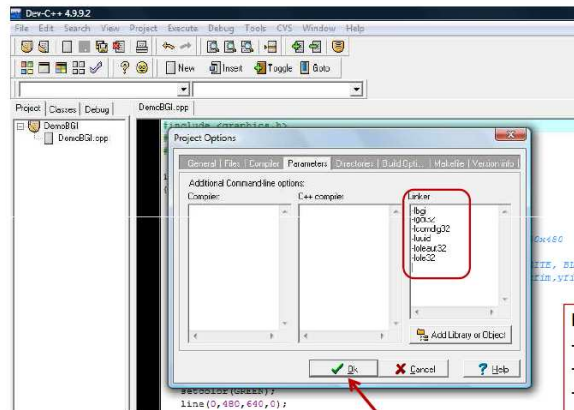
2

Maio 2010

Biblioteca de Funções: Rotinas Gráficas

Rotinas Gráficas: WinBGI

Biblioteca do tipo .DLL/.LIB/.A

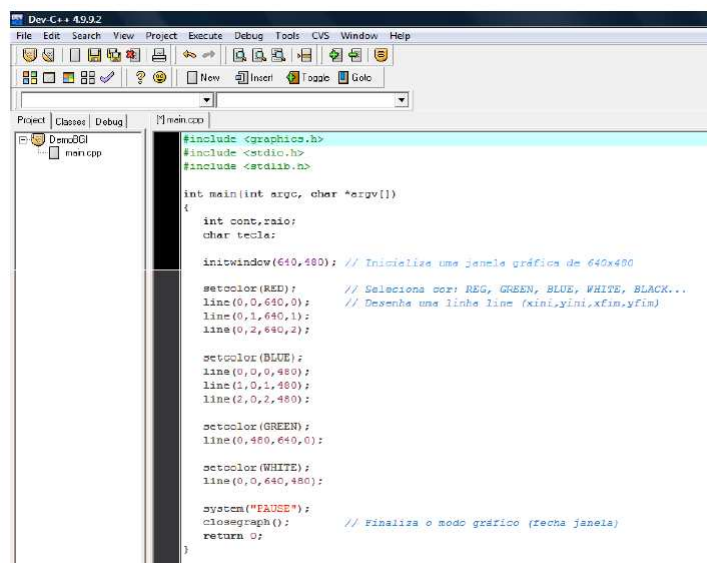


Bibliotecas:
-bgi
-lgdi32
-lcomdlg32
-luuid
-oleaut32
-ole32

Ver Arquivo: Demo-WINBGI-Passo-a-Passo.pdf

Biblioteca de Funções: Rotinas Gráficas

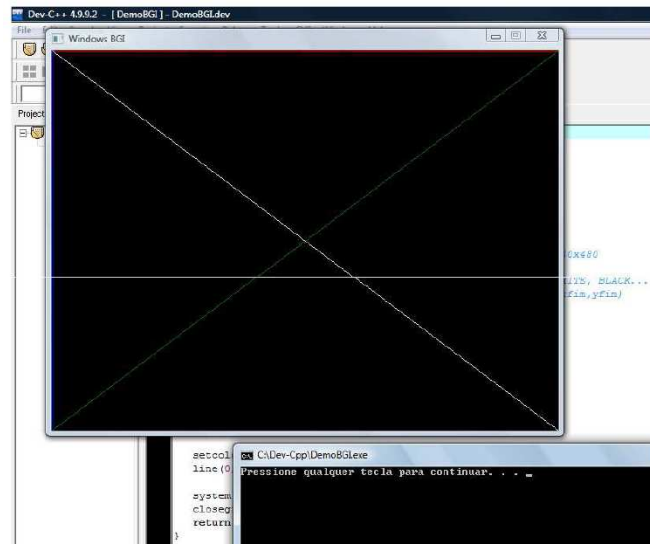
Rotinas Gráficas WinBGI Funções



Biblioteca de Funções: Rotinas Gráficas

Rotinas Gráficas WinBGI

Run!



FIM!

Biblioteca de Funções: Rotinas Gráficas

Rotinas Gráficas: WinBGI

HELP!

Borland Graphics Interface (BGI) for Windows

Version 6.0, August 9, 2004

The following functions are mostly from the original Borland Graphics Interface for DOS programs. The BGI graphics functions may also be used with Windows programs created by the Borland 5.0 compiler, the free [GNU C++ compiler](#), and possibly other compilers. Extra Windows functions are also available, described in www.cs.colorado.edu/~main/cs1300/doc/bgi/bgi.html. These extra functions are indicated below by **WIN**. Also, any of the functions that use colors can use [RGB colors](#) in addition to the 16-color BGI palette.

Functions:

```
int kbhit (void); WIN

void line (int x1, int y1, int x2, int y2);
void linerel (int dx, int dy);
void lineto (int x, int y);

int mousex (void); WIN
int mousey (void); WIN
void moverel (int dx, int dy);
void moveto (int x, int y);

void outtext (char *textstring);
void outtextxy (int x, int y, char *textstring);
```

Biblioteca de Funções: Rotinas Gráficas

Rotinas Gráficas: WinBGI HELP!

line

Syntax

```
#include <graphics.h>

void line(int x1, int y1, int x2, int y2);
```

Description

line draws a line in the current color, using the current line style and thickness between the two points specified, (x1,y1) and (x2,y2), without updating the current position (CP).

Return Value

None.

See also

[getlinesettings](#)
[linerc1](#)
[lineto](#)
[setcolor](#)
[setlinestyle](#)
[setwritemode](#)

Example

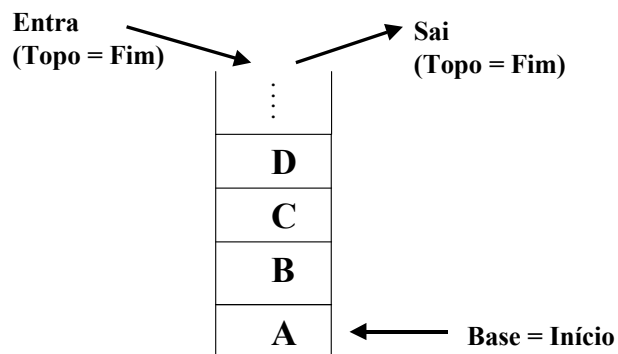
```
/* line example */

#include <graphics.h>
```

Biblioteca de Funções: Pilha de Dados

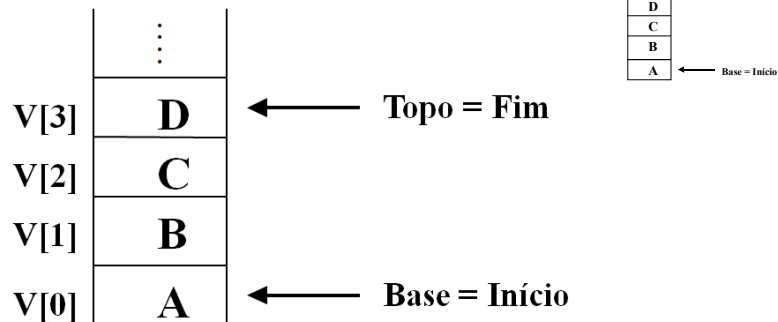
Pilha de Dados:

Podemos criar estruturas de dados em vetores (com alocação estática ou dinâmica) para armazenar uma pilha de dados, também conhecido por **STACK (LIFO: Last In, First Out)**



Pilha de Dados:

Podemos criar estruturas de dados em vetores (com alocação estática ou dinâmica) para armazenar uma pilha de dados, também conhecido por **STACK (LIFO: Last In, First Out)**



Pilha de Dados:

Podemos ter o seguinte tipo de estrutura de dados para descrever uma pilha:

```
#define FALSO      0
#define VERDADEIRO 1

#define OK        1
#define ERRO      0

typedef int Tipo_Dado;

typedef struct
{
    Tipo_Dado *Dado;
    int Base, Topo;
} Tipo_Pilha;

int MAX_PILHA=0;
```

```
void cria_pilha (Tipo_Pilha *, int );
int insere_pilha (Tipo_Pilha *, Tipo_Dado );
int retira_pilha (Tipo_Pilha *, Tipo_Dado *);
void exhibe_pilha (Tipo_Pilha );
int quantidade_pilha (Tipo_Pilha );
int cheia_pilha (Tipo_Pilha );
int vazia_pilha (Tipo_Pilha );
int esvazia_pilha (Tipo_Pilha *);
```

Pilha de Dados:

Podemos ter o seguinte tipo de estrutura de dados para descrever uma pilha:

```
#define FALSO      0
#define VERDADEIRO 1

#define OK        1
#define ERRO      0

typedef int Tipo_Dado;

typedef struct
{
    Tipo_Dado *Dado;
    int Base, Topo;
} Tipo_Pilha;

int MAX_PILHA=0;
```

```
void cria_pilha(P, NElem)
Tipo_Pilha *P;
int NElem;
{
    P->Base=0;
    P->Topo=0;
    P->Dado= (Tipo_Dado *) calloc
              (NElem, sizeof(Tipo_Dado));
    MAX_PILHA=NElem;
}
```

Pilha de Dados:

Observe a notação que foi usada para acesso aos dados do registro (struct) através de ponteiros:

Acessar o campo “Base”
da Struct Tipo_Pilha através do ponteiro P

```
P->Base=0;
```

É o mesmo que

```
(*P).Base = 0;
```

Biblioteca de Funções: Pilha de Dados

Pilha de Dados:

Podemos ter o seguinte tipo de estrutura de dados para descrever uma pilha:

```
#define FALSO      0
#define VERDADEIRO 1

#define OK        1
#define ERRO      0

typedef int Tipo_Dado;

typedef struct
{
    Tipo_Dado *Dado;
    int Base, Topo;
} Tipo_Pilha;

int MAX_PILHA=0;
```

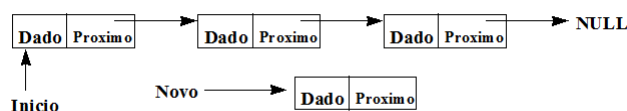
```
int insere_pilha (P, Dado)
Tipo_Pilha *P;
Tipo_Dado Dado;
{
    /* Vetor nao esta cheio ? */
    if (P->Topo < MAX_PILHA)
    {
        P->Dado[P->Topo]=Dado;
        (P->Topo)++;
        return(OK);
    }
    else
        return(ERRO);
}
```

13
Maio 2010

Biblioteca de Funções: Lista Encadeada

Lista Encadeada:

Estruturas criadas com registros (typedef e structs) podem ser encadeadas através do uso de ponteiros, onde estes blocos estão espalhados pela memória sendo conectados por ligações (ponteiros) e resultando em estruturas denominadas de “*listas encadeadas*” de alocação dinâmica.



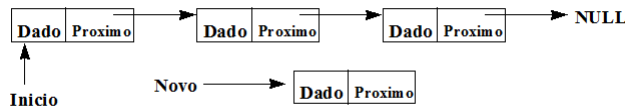
A figura acima mostra um exemplo de lista encadeada, baseada na estrutura de dados descrita logo a seguir.

14
Maio 2010

Lista Encadeada:

Podemos ter o seguinte tipo de estrutura de dados para descrever uma lista encadeada:

```
typedef int Tipo_Dado;  
  
typedef struct bloco {  
    Tipo_Dado Dado;  
    struct bloco *Proximo;  
} T_Nodo;
```



15

Maio 2010

Lista Encadeada:

```
void inicializa_lista (Nodo **N);  
int insere_inicio_lista (Nodo **N, Tipo_Dado Dado);  
int insere_fim_lista (Nodo **N, Tipo_Dado Dado);  
int insere_ordenando_lista (Nodo **N, Tipo_Dado Dado);  
int remove_inicio_lista (Nodo **N, Tipo_Dado *Dado);  
int remove_fim_lista (Nodo **N, Tipo_Dado *Dado);  
int remove_elemento_lista (Nodo **N, Tipo_Dado Dado);  
int quantidade_lista (Nodo *N);  
void exibe_lista (Nodo *N);  
int pesquisa_lista (Nodo **N, Tipo_Dado Dado);  
int percorre_lista (Nodo **N, Tipo_Dado *Dado);  
void apaga_lista (Nodo **N);
```

16

Maio 2010

Lista Encadeada: Como foi feita...

```
void inicializa_lista(N)
Nodo **N;
{
    *N = NULL;
}

int insere_inicio_lista(N, Dado)
Nodo **N;
Tipo_Dado Dado;
{
    Nodo *novo;

    novo = (Nodo *) calloc ( 1, sizeof (Nodo) );
    if ( novo == NULL )
        return (ERRO); /* Nao conseguiu alocar espaço na memória */
    novo -> Dado = Dado;
    novo -> Proximo = *N;
    *N = novo;
    return (OK);
}
```

17

Maio 2010

Lista Encadeada: Como foi feita... VISÃO DE UM USUÁRIO

```
void inicializa_lista(N)
Nodo **N;
{
    
}

int insere_inicio_lista(N, Dado)
Nodo **N;
Tipo_Dado Dado;
{
    
}

Código
```

18

Maio 2010

Lista Encadeada: VISÃO DE UM USUÁRIO

```
main()
{
    Nodo *Lista;
    int Valor;

    printf("\n>>> Rotinas de manipulacao de listas simples encadeadas <<<\n\n");

    inicializa_lista(&Lista);

    if (insere_inicio_lista(&Lista,1) ) printf("=> Inserido dado: 1\n");
    else printf(">>> Erro de insercao na lista!\n");

    if (insere_inicio_lista(&Lista,2) ) printf("=> Inserido dado: 2\n");
    else printf(">>> Erro de insercao na lista!\n");

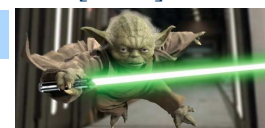
    if (insere_inicio_lista(&Lista,3) ) printf("=> Inserido dado: 3\n");
    else printf(">>> Erro de insercao na lista!\n");

    exibe_lista(Lista);

    printf("Pressione uma tecla para terminar o programa...\n");
    getch();
}
```

“Fear is the path to the dark side” [Yoda]

May the force be with you



```
int insere_antes_ld (LD, Dado)
Nodo_ID = *LD;
Tipo_Dado Dado;
{
    Nodo_ID *novo, *aux;

    novo=(Nodo_ID *)calloc(1, sizeof(Nodo_ID));
    if (novo == NULL)
        return (ERRO);

    novo->Dado = Dado;

    if (*LD == NULL)
    {
        *LD = novo;
        novo->Anterior = NULL;
        novo->Proximo = NULL;
    }
    else
    {
        aux=(*LD)->Anterior;
        (*LD)->Anterior=novo;
        novo->Proximo=(*LD);
        novo->Anterior=aux;
        if (aux != NULL)
            aux->Proximo=novo;
    }
    return (OK);
}
```



INFORMAÇÕES SOBRE A DISCIPLINA

USP - Universidade de São Paulo - São Carlos, SP
ICMC - Instituto de Ciências Matemáticas e de Computação
SSC - Departamento de Sistemas de Computação

Prof. Fernando Santos OSÓRIO

Web institucional: <http://www.icmc.usp.br/ssc/>

Página pessoal: <http://www.icmc.usp.br/~fosorio/>

E-mail: fosorio [at] icmc. usp. br ou fosorio [at] gmail. com

PAE Gustavo Pessin – E-mail: pessin [at] gmail .com

Monitor Matheus Lin – E-mail: matheus.lin [at] gmail.com

Disciplina de Introdução a Ciência da Computação

Web disciplina: Wiki ICMC - [Http://wiki.icmc.usp.br](http://wiki.icmc.usp.br)

> Programa, Material de Aulas, Critérios de Avaliação,

> Trabalhos Práticos, Datas das Provas, Notas