/\*

\* Arquivo: pilha.h

\* Biblicoteca de operações para pilha

\* implementada com vetor.

\*/

#define VERDADEIRO 1

#define FALSO 0

struct tipo\_pilha {

elemento vet[TAM];

int topo;

};

typedef struct tipo\_pilha pilha;

//Cria pilha.

void CriarPilha(pilha \*p) {

(\*p).topo = 0;

}

//Verifica se a pilha está vazia.

int PilhaVazia(pilha \*p) {

return((\*p).topo == 0);

}

//Verifica se a pilha está cheia.

int PilhaCheia(pilha \*p) {

return((\*p).topo == TAM);

}

//Insere um elemento na pilha.

void Empilhar(pilha \*p, elemento e) {

(\*p).vet[(\*p).topo] = e;

(\*p).topo++;

}

//Retira um elemento na pilha.

elemento Desempilhar(pilha \*p) {

(\*p).topo--;

return (\*p).vet[(\*p).topo];

}

void DesempilharArquivo(pilha \*p, FILE \*pont\_arq) {

int num = 0;

printf("\nNumeros informados: ");

while(PilhaVazia(p) == FALSO) {

num = Desempilhar(p);

fprintf(pont\_arq,"\n%d", num);

}

}

void EmpilharArquivo(pilha \*p, int c[]) {

for (int e = 0; e < TAM;e++){

(\*p).vet[(\*p).topo] = c[e];

(\*p).topo++;

}

}

void ImprimePilha(pilha \*p)

{

int num = 0;

printf("\nNumeros informados: ");

while(PilhaVazia(p) == FALSO) {

num = Desempilhar(p);

printf("\n%d", num);

}

}