Unoeste
Campus I Rua José Bongiovani, 700
Cidade Universitária CEP 19050 900
Fone (18) 3229 2003

Faculdade de Informática de Presidente Prudente
Fone (18) 3229 1060
www.unoeste.br/fipp
e-mail: fipp@fipp.unoeste.br

FIPP FACULDADE DE INFORMÁTICA DE PRESIDENTE PRUDENTE CAMPUS I

Estruturas de Dados II

Prof. Francisco Assis da Silva

– Simulação de programas escritos em linguagem C –

Seu trabalho é fazer uma simulação de execução de um pequeno programa na linguagem C. A Figura 1 mostra em (a) o conteúdo da memória RAM (pilha de variáveis) e em (b) a tela do seu simulador, com a execução do programa Exemplo 1 em andamento.

Exemplo 1:

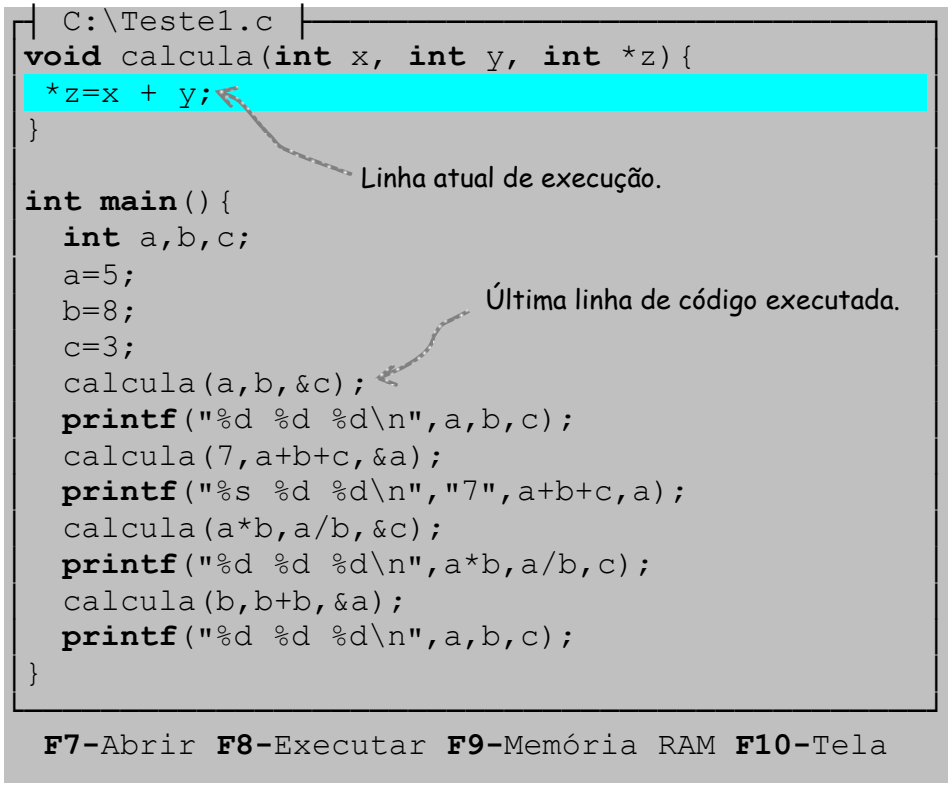
```
#include<stdio.h>

void calcula(int x, int y, int *z)
{
  *z=x + y;
}

int main()
{
  int a,b,c;
  a=5;
  b=8;
  c=3;
  calcula(a,b,&c);
  printf("%d %d %d\n",a,b,c);
  calcula(7,a+b+c,&a);
  printf("%s %d %d\n","7",a+b+c,a);
  calcula(a*b,a/b,&c);
  printf("%d %d %d\n",a*b,a/b,c);
  calcula(b,b+b,&a);
  printf("%d %d %d\n",a,b,c);
}
```

&		
100	a	5
104	b	8
108	c	3
112	x	5
116	y	8
120	z	[108]

(a)



(b)

Figura 1. Simulação da execução de um programa na linguagem C. (a) conteúdo da pilha de variáveis. (b) tela do simulador, com a simulação de um pequeno programa em execução.

O simulador deve conseguir lidar com os seguintes comandos e estruturas:

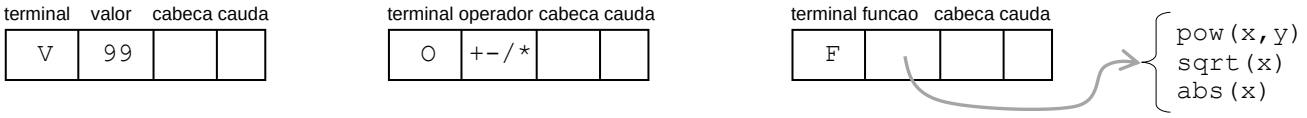
a) Fazer chamadas de funções com passagem de parâmetro por valor, por referência e por `return`;
b) Fazer chamadas de funções dentro de outra função, com tratamento de parâmetros;
c) Resolver equações aritméticas e algumas funções matemáticas: potência, raiz quadrada e módulo (valor absoluto);
d) Declarações e atribuição de valores a variáveis (apenas do tipo `int` e `float`);
e) Executar a estrutura de repetição `while`;
f) Executar o comando de exibição `printf`.

Seu programa deve permitir ao usuário abrir um arquivo fonte na linguagem C (F7), executar o programa passo a passo (F8), mostrar o conteúdo da memória RAM no momento atual da execução (F9) e mostrar a tela (`printf` dos resultados) (F10). Para a execução passo a passo, deve-se dar ENTER para executar a linha atual (linha com fundo demarcado) e passar para a próxima.

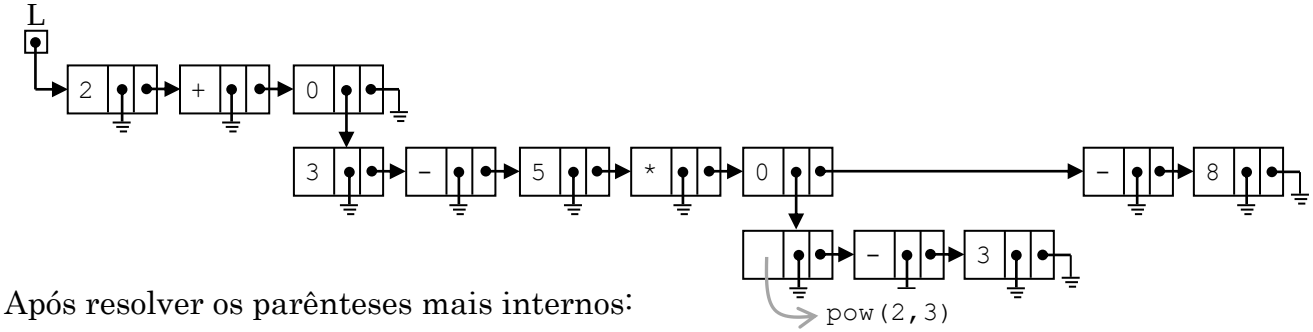
Você deve utilizar apenas estruturas de listas para construir seu programa simulador:

a) Deverá ter uma lista para representar a pilha de variáveis;
b) Deverá ter uma lista de listas para guardar as linhas de comandos do programa em C que abriu. Cada nodo da lista principal (considere uma lista vertical) terá uma lista vinculada a ela para guardar em cada nodo os termos da linha de comandos (considere uma lista horizontal vinculada ao nodo vertical).
   Exemplo: Se a linha de comandos fosse "**`void`** `calcula(`**`int`** `x,` **`int`** `y,` **`int`** `*z)`", você teria, na lista horizontal, um nodo para "`void`", outro para "`calcula`", outro para "`(`", outro para "`int x`", outro para "`int y`", outro para "`int *z`" e outro para "`)`";

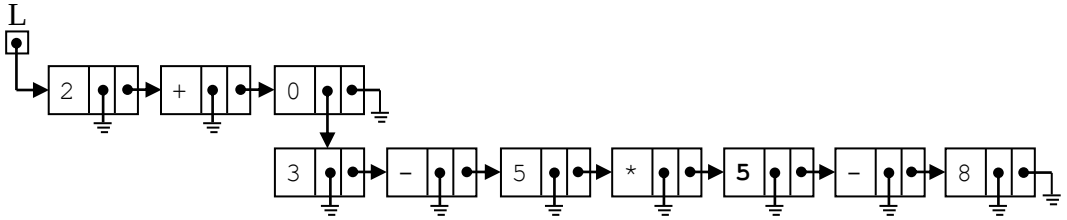
Para resolver as expressões matemáticas, você deverá utilizar uma lista generalizada, diferente do modelo conceitual das aulas. A partir de uma **string** (char [ ]) passada por parâmetro para uma de suas funções, crie um algoritmo para construir uma lista generalizada segundo as prioridades e resolver a expressão "podando" os nodos da lista generalizada até sobrar um único nodo com o resultado, cujo valor deverá ser retornado. A medida que cada linha de operação for resolvida, os nodos da lista generalizada devem ser removidos e sobrar apenas um nodo com o resultado. O nodo da lista deverá ser construído com o uso de **union**, uma vez que o nodo poderá ser um valor (**float**), um operador (**char**) ou uma função (**pow**, **sqrt** ou **abs**).
Tipos de nodos:



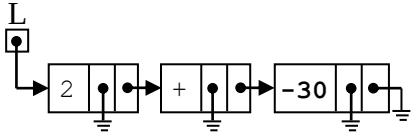
Exemplo de expressão aritmética a ser resolvido: "`2+(3-5*(pow(2,3)-3)-8)`"
A lista construída será:



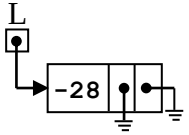
Após resolver os parênteses mais internos:



Após resolver o segundo nível de parênteses:



Após resolver toda a expressão:



Observações importantes:
Não utilize funções prontas da linguagem C, como por exemplo, strtok para separar string, filas, pilhas etc.
Para resolver cada lista de expressões, você pode utilizar duas pilhas (criadas por você), uma de valor e outra de operador.

Exemplos de códigos em C contendo comandos e estruturas que seu programa deverá ser capaz simular a execução:

Exemplo 2:

```
#include<stdio.h>

void calcula1(int *I, int *J, int K)
{
  *I=*I+3;
  *J=*I+K;
  K=K+1;
}

void calcula2(int I, int J, int *K)
{
  I=I+2;
  J=J+*K;
  *K=*K+1;
}

void executa(int *p1, int *p2, int *p3)
{
  calcula1(&*p3, &*p1,*p2);
  calcula2(*p2, *p2, &*p2);
}

int main()
{
  int A=3,B=5,C=7;
  executa(&A,&B,&C);
  printf("Primeiro valor modificado = %d\n",A);
  printf("Segundo valor modificado = %d\n",B);
  printf("Terceiro valor modificado = %d\n",C);
}
```

Exemplo 3:

```
#include<stdio.h>

int faz_uma_repeticao(int X, int n)
{
  int i, result;
  i=0;
  result=1;
  while (i<n)
  {
    result=result * X;
    i=i+1;
  }
  return result;
}

int main()
{
  int s;
  s = faz_uma_repeticao(2,4);
  printf("Resultado = %d\n",s);
}
```

Exemplo 4:

```
#include<stdio.h>
#include<math.h>

float juros_composto(float P, float i, int n)
{
  float M;
  M = P*pow(1 + i/100,n);
  return M;
}

int main()
{
  float res = juros_composto(2000, 1.5, 10);
  printf("Resultado = %f\n",res);
}
```

Importante!!!

Data da apresentação: 28/03/2023

O que deve enviar no Aprender no dia da apresentação:

a) Todos os códigos-fonte;
b) Os programas em C usados para testar o simulador. Tem que ser diferentes dos Exemplos 1 e 2;
c) Um .docx com o desenho das estruturas (`structs` das listas) usadas.