Lista de exercícios (GDB)

Instalação e Confuguração:

- Instale o GCC em seu sistema operacional;
- Instale o GDB em seu sistema operacional;
- Compile um código simples em linguagem C com depuração (usando a flag -g);
- Coloque um breakpoint em uma determinada linha de código e execute o programa até esse ponto.
- Execute o programa no GDB usando o comando *run*;

Código 1 - Fibonacci

```c
#include<stdio.h>
int fibonacci(int n) {
    if (n <= 1) {
        return n;
    }
    else {
        return fibonacci(n - 1) + fibonacci(n - 2);
    }
}
int main(){
    fibonacci(15);
}
```

a) Depure o código para descobrir quantas vezes o **fibonacci(5)** é calculado (requisitado).

b) Descubra através da depuração, qual a maior quantidade de níveis (*frames*) esse código alcança.

c) Depure o código até o momento que o primeiro fibonacci(14) é calculado e o executável irá começar a calcular o fibonacci(13).

Código 2 - Operação Matemática

```c
#include <stdio.h>

// Funcao para calcular algo
int operacaoMatematica(int a, int b) {
    int r;

    while (b != 0) {
        r = a % b;
        a = b;
        b = r;
    }

    return a;
}

int main() {
    int num1, num2;

    printf("Digite o primeiro numero: ");
    scanf("%d", &num1);

    printf("Digite o segundo numero: ");
    scanf("%d", &num2);

    int res = operacaoMatematica(num1, num2);

    printf("A operacao de %d e %d e: %d\n", num1, num2, res);

    return 0;
}
```

a) Execute o algoritmo em modo de depuração.

b) Descubra através da depuração, qual a maior profundidade de (*frames*) esse código alcança.

c) Descubra o que ocorre quando um parâmetro é 0.

d) Descubra o que ocorre quando os dois parâmetros apresentados são iguais e maiores que 1.

Código 3 - Números primos

```c
#include <stdio.h>

// Funcao recursiva para verificar se um numero e primo
int ehPrimoRecursivo(int n, int i) {
    if (n <= 2) {
        return (n == 2);
    }

    if (n % i == 0) {
        return 0;
    }

    if (i * i > n) {
        return 1;
    }

    return ehPrimoRecursivo(n, i + 1);
}

// Funcao para imprimir numeros primos em um intervalo
void imprimirPrimosIntervalo(int inicio, int fim) {
    if (inicio > fim) {
        return;
    }

    if (ehPrimoRecursivo(inicio, 2)) {
        printf("%d ", inicio);
    }

    imprimirPrimosIntervalo(inicio + 1, fim);
}

int main() {
    int inicio, fim;

    printf("Digite o inicio do intervalo: ");
    scanf("%d", &inicio);

    printf("Digite o fim do intervalo: ");
    scanf("%d", &fim);

    printf("Numeros primos entre %d e %d: ", inicio, fim);
    imprimirPrimosIntervalo(inicio, fim);
    printf("\n");

    return 0;
}
```

a) Através da depuração, compreenda como o código funciona.

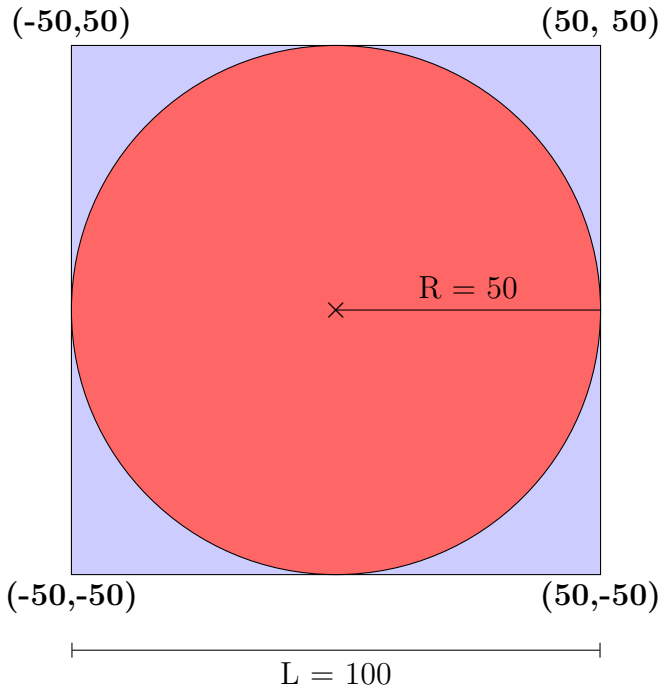
b) Desenhe em formato de uma árvore de recursão os passos para o cálculo da procura dos números primos de 10 a 20 (usando a depuração).

c) Conte a quantidade de chamadas recursivas para realizar o resultado de b).

d) Depure o algoritmo com os valores de entrada de 1000 e 10000. Utilize o depurador para interromper a depuração quando o 240º número primo aparecer.

Código 4 - Função aleatória

A figura a seguir representa um círculo inscrito em uma circunferência. A origem (0,0) do centro de coordenadas coincide com a origem do círculo e o centro do quadrado. Foi desenvolvido um algoritmo para gerar pontos aleatórios que aparecerão nos limites deste quadrado. O objetivo é selecionar apenas os pontos que estão dentro do círculo. A seguir, apresentamos o código e algumas questões relacionadas ao mesmo.




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

int randomInRange(int min, int max) {
    return min + rand() % (max - min + 1);
}

void gerarNumerosAleatorios() {
    int i;
    for (i = 0; i < 100; i++) {
        int px = randomInRange(0, 100);
        int py = randomInRange(0, 100);
        printf("P = (%d, %d)\n ", px, py);
    }
    printf("\n");
}

int main() {
    srand(42);
    gerarNumerosAleatorios();
}
```

a) Através da depuração, realize a interrupção do algoritmo toda vez que o número aleatório é maior que 0.

b) Utilize a depuração para realizar interrupções no algoritmo somente quando os pontos estiver na região externa ao círculo.

c) Utilize a depuração para realizar interrupções no algoritmo somente quando os pontos estiver na região interna ao círculo.