Programação em C++

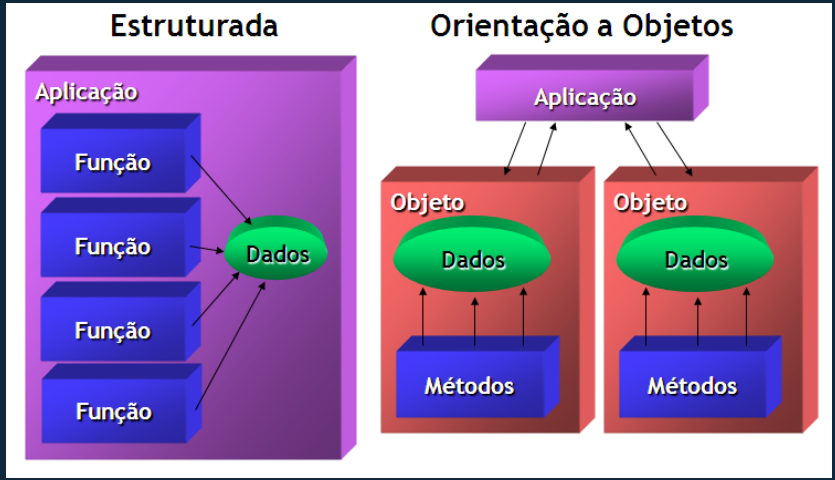
Jeferson W. D. Fernandes

1

Noções básicas de C++

Começar a programar em linguagem C++

Primeiro programa em C++



Classe

Representação de um conjunto de objetos com características afins. Definição do comportamento dos objetos (métodos) e seus atributos.

Objeto

Instância de uma classe.

Armazenamento de estados através de seus atributos e reação a mensagens enviadas por outros objetos.

Herança

Mecanismo pela qual uma classe (sub-classe) pode estender outra classe (super-classe), estendendo seus comportamentos e atributos.

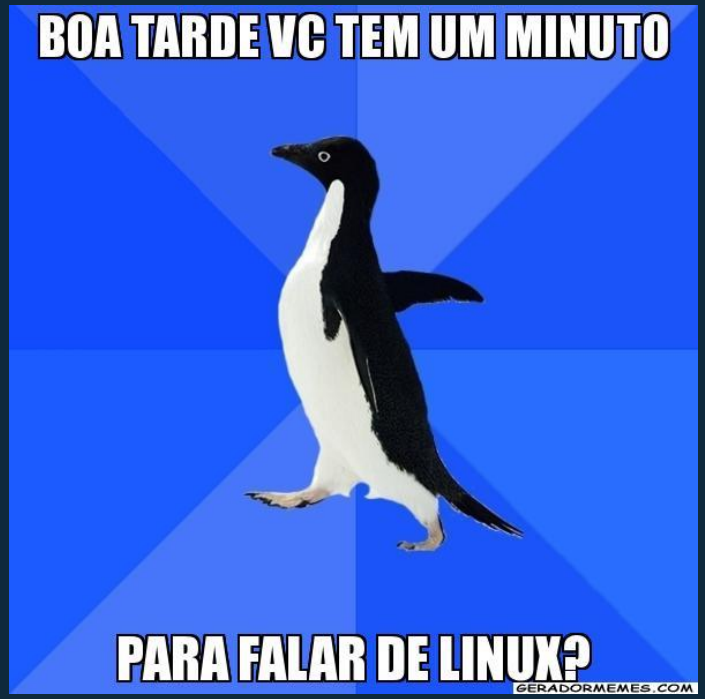
Polimorfismo

Princípio pelo qual as instâncias de duas classes ou mais classes derivadas de uma mesma super-classe podem invocar métodos com a mesma assinatura, mas com comportamentos distintos.

Encapsulamento

Proibição do acesso direto ao estado de um objeto, disponibilizando apenas métodos que alterem esses estados na interface pública.

Primeiro programa em C++



Bitvise SSH Client

Host:	p2
Port:	22
Username:	curso
Senha:	curso2018

Primeiro programa em C++

Hello, World!

```
#include <iostream>
//using namespace std;

int main() {
    std::cout << "Hello, World!“ << std::endl;
    return 0;
}
```

Compilar

```
g++ hello.cpp –o test
```

Executar

```
./test
```

Data types

```
#include <iostream>
using namespace std;

int main() {
    cout << "Hello, World!“ << endl;

    char   charVariable[5] = “C++”;

    cout << charVariable << endl;

    int    iNumber = 5;
    float  fNumber = 7.5;
    double dNumber = 3. / 5.;
    bool   bVariable = true;

    cout << iNumber << fNumber << dNumber <<
bVariable << endl;

    return 0;
}
```

Primeiro programa em C++

Loops

```
#include <iostream>
using namespace std;

int main() {
    int a;

    for (int i=0; i<5; i++){
        a *= 2;
    };

    if (a < 100){
        while (a < 100){
            a++;
        };
    }else{
        cout << “Valor de a = ” << a << endl;
    };
    (a < 10) ? (a = 10) : (a = 11);
}
```

Operadores lógicos

<	Menor que
>	Maior que
<=	Menor ou igual a
>=	Maior ou igual a
==	Igual a
!=	Diferente de
&&	e
||	ou
false	falso
true	verdadeiro

Ponteiros

```
#include <iostream>

int main() {

    int var = 1.;
    int *pontVar;

    pontVar = &var;

    std::cout << "Ponteiro " << var << " "
              << *pontVar << std::endl;

    var = 5;

    std::cout << "Ponteiro " << var << " "
              << *pontVar << std::endl;

    return 0;
};
```

Primeiro programa em C++

Classes

```cpp
#include <iostream>

using namespace std;

#include “Person.hpp“
#include “GradeFreq.hpp"

int main() {
    Person *person1, *person2;
    GradeAndFrequency grade1, grade2;

    person1 = new Person(52,100,1.95);

    person1 -> setAge(25);
    grade1.setGrade(9.2);

    double g1 = grade1.getGrade();

    int age1 = person1 -> getAge();
    cout << "Pages " << age1 << " " << g1 << endl;

    return 0;
};
```

Person.hpp

```cpp
#include <iostream>
using namespace std;

// Class Person
class Person{
private:
    int      age_;
    double   weight_;
    double   height_;

public:
    Person(int a, double w, double h){
        age_ = a;
        weight_ = w;
        height_ = h;
    };
    void setAge(int a){
        age_ = a;
    };
    int getAge(){
        return age_;
    };
};
```

GradeFreq.hpp

```cpp
#include <iostream>
using namespace std;

//Class Grade and Frequency
class GradeAndFrequency{

private:
    double   grade_;
    double   frequency_;

public:
    void setGrade(double g){
        grade_ = g;
    };

    double getGrade(){
        return grade_;
    };

};
```

Primeiro programa em C++

Matrizes e vetores – biblioteca boost

```
#include <iostream>
#include <boost/numeric/ublas/vector.hpp>
#include <boost/numeric/ublas/matrix.hpp>

using namespace boost::numeric;

int main() {

    ublas::bounded_vector<double, 10>       myVector;
    ublas::bounded_matrix<double, 10,10>    myMatrix;

    myVector(0) = 5.;
    myMatrix(2,2) = 1.;

    myVector = prod(myMatrix,myVector);
    myMatrix.clear();

    ublas::identity_matrix<double>          ident(10);

    std::cout << "My Vector " << myVector(3) << std::endl;

    return 0;
};
```

Pilhas e Pares

```
#include <iostream>
#include <vector>
#include <cmath>

int main() {
    std::vector<double>      myVector;
    double pi = M_PI;

    myVector.push_back(4.2);
    myVector.push_back(1./3.);
    myVector.push_back(pi);

    std::cout << "My Vector " << myVector[0] << " "
              << myVector[1] << " " << myVector[2] << " "
              << myVector.size() << std::endl;

    std::pair<double, int> myPair = std::make_pair(pi, 2);

    std::cout << "My Pair " << myPair.first << " "
              << myPair.second << std::endl;

    return 0;
};
```

Primeiro programa em C++

Templates

```
#include <iostream>
#include <string>

using namespace std;

template <typename T>
T max(T &val1, T &val2) {
    return (val1 > val2 ? val1 : val2);
}

template <typename T>
T min(T &val1, T &val2) {
    return (val1 < val2 ? val1 : val2);
}
```

```
int main() {
    int i1, i2;
    float f1, f2;
    string s1, s2;
    cout << "Introduza dois valores inteiros:" << endl;
    cin >> i1;
    cin >> i2;
    cout << "O inteiro maior e: " << max(i1, i2) << endl;
    cout << "O inteiro menor e: " << min(i1, i2) << endl;
    cout << "Introduza dois valores do tipo float:" << endl;
    cin >> f1;
    cin >> f2;
    cout << "O float maior e: " << max(f1, f2) << endl;
    cout << "O float menor e: " << min(f1, f2) << endl;
    cout << "Introduza 2 strings:" << endl;
    cin >> s1;
    cin >> s2;
    cout << "A string maior e: " << max(s1, s2) << endl;
    cout << "A string menor e: " << min(s1, s2) << endl;
}
```

Paralelização em MPI

Ganhar velocidade de processamento

Primeiro programa em MPI

```cpp
#include <iostream>
#include <mpi.h>

int main(int argc, char **argv){

    int nProcessos, id;

    MPI_Init(&argc, &argv);

    MPI_Comm_size(MPI_COMM_WORLD, &nProcessos);
    MPI_Comm_rank(MPI_COMM_WORLD, &id);

    std::cout << " Processo " << id << " de " << nProcessos << std::endl;

    MPI_Finalize();
};
```

Compilar

```
mpic++.openmpi hello.cpp -o test
```

Executar

```
mpirun.openmpi -np N test
```

Data Types em MPI

Tipo C	Identificador MPI
char	MPI_CHAR
short	MPI_SHORT
int	MPI_INT
long	MPI_LONG
unsigned char	MPI_UNSIGNED_CHAR
unsigned short	MPI_UNSIGNED_SHORT
unsigned int	MPI_UNSIGNED_INT
unsigned long	MPI_UNSIGNED_LONG
float	MPI_FLOAT
double	MPI_DOUBLE
long double	MPI_LONG_DOUBLE

Primeiro programa em MPI

Principais comandos

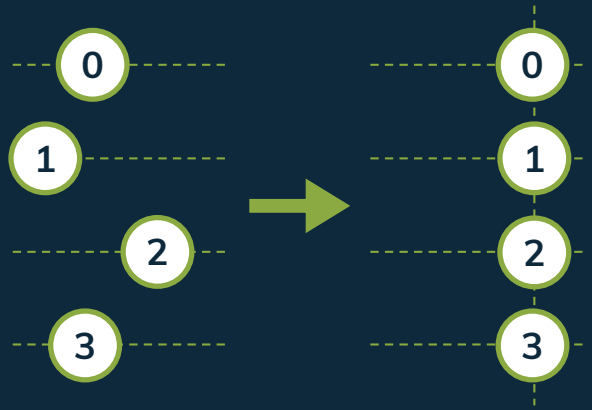
```
MPI_Send(void* data, int count, MPI_Datatype datatype, int destination, int tag, MPI_Comm communicator)

MPI_Recv(void* data, int count, MPI_Datatype datatype, int source, int tag, MPI_Comm communicator, MPI_Status* status)

MPI_Bcast(void* data, int count, MPI_Datatype datatype, int root, MPI_Comm communicator)

MPI_Barrier(MPI_Comm communicator)

MPI_Scatter(void* send_data, int send_count, MPI_Datatype send_datatype, void* recv_data, int recv_count,
            MPI_Datatype recv_datatype, int root, MPI_Comm communicator)

MPI_Gather(void* send_data, int send_count, MPI_Datatype send_datatype, void* recv_data, int recv_count,
           MPI_Datatype recv_datatype, int root, MPI_Comm communicator)
```

Send

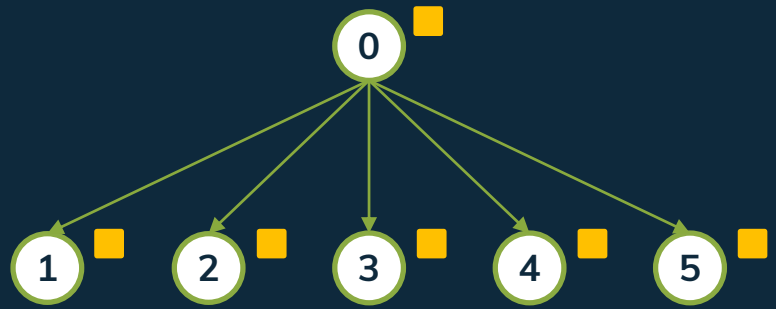
Receive

Primeiro programa em MPI

Barier

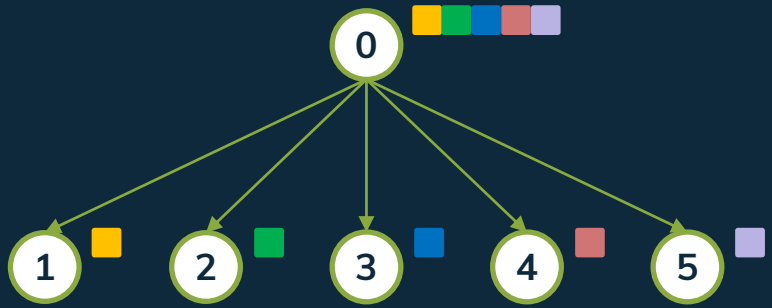


Broadcast

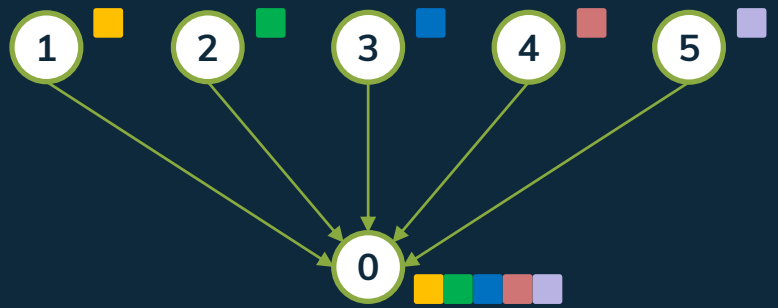


Primeiro programa em MPI

Scatter



Gather



MAIS INFORMAÇÕES?

https://wikimechanica.eesc.usp.br/mpi

Makefile

Compilar e executar meu programa de forma eficiente

Makefile

O que é?

O makefile é um arquivo para configuração da compilação e execução de um programa.

Por que usar?

O makefile ajuda a dar mais produtividade e permite organizar melhor o código.

Como fazer um makefile?

Existem diversos tutoriais na internet que podem ser úteis para a criação de makefiles.

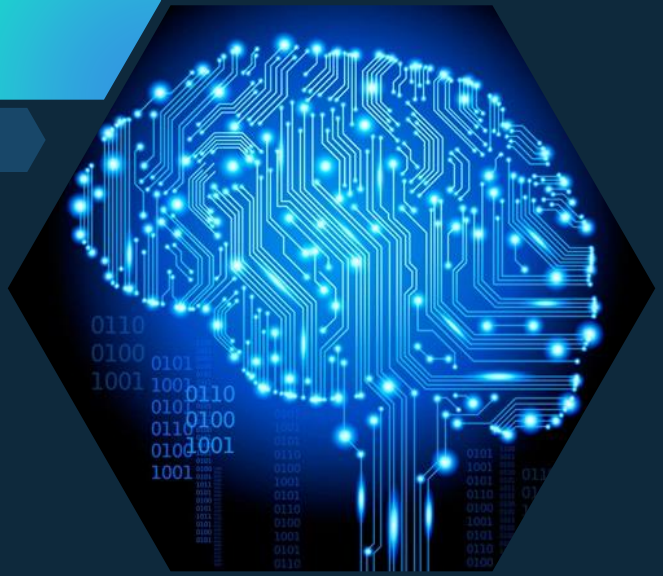
MAIS INFORMAÇÕES?

https://wikimechanica.eesc.usp.br/makefile

4

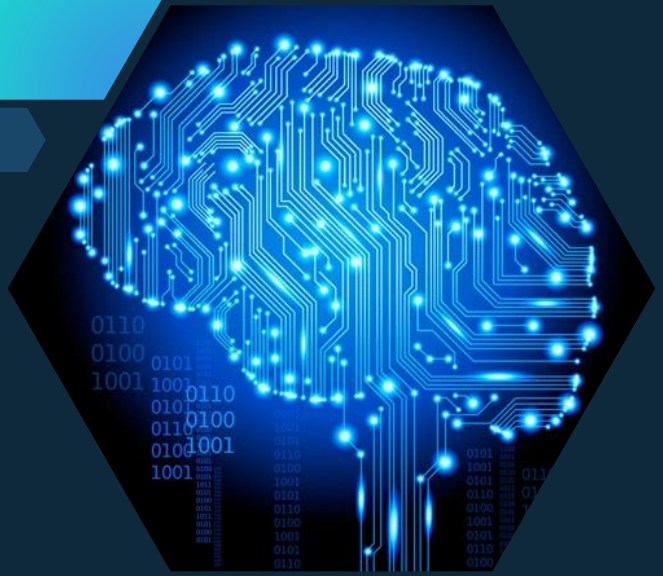
PETSc

Manipular matrizes e vetores e resolver sistemas lineares utilizando uma biblioteca robusta



O que é o PETSc?

PETSc é um conjunto de bibliotecas escritas em linguagem C, mas possíveis de serem utilizadas em aplicações escritas em **C, C++, Fortran, Python e Matlab**. Tem como objetivo a aplicação em projetos de **larga escala**. Possui interface simples para usuários iniciantes, porém seu design cuidadoso permite que usuários avançados tenham controle detalhado sobre o processo de solução. O PETSc inclui um grande conjunto de *solvers* de **equações lineares, não lineares e integradores para ODE's**. O PETSc ainda possui muitos dos mecanismos necessários dentro de códigos de **aplicação paralelos**, como rotinas simples de matriz paralela e de montagem de vetores que permitem a sobreposição de comunicação e computação. Além disso, o PETSc inclui suporte para matrizes distribuídas paralelas úteis para métodos de diferenças finitas.

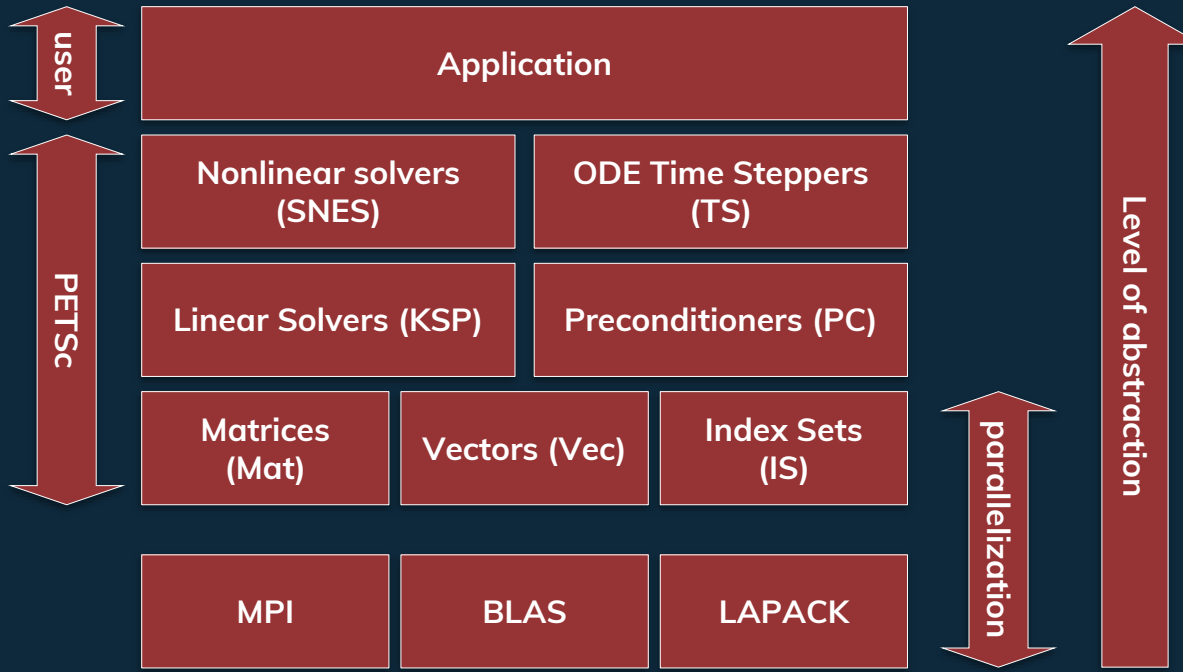


O que é o PETSc?

O Pacote inclui:

- Vetores paralelos;
- Matrizes paralelas;
- Pré-condicionadores paralelos;
- Métodos de Krylov;
- Solvers não lineares com base no método de Newton
- Integradores temporais paralelospara ODE's;
- Suporte para placas Nvidia GPU;
- Documentação completa;
- Perfis automáticos de ponto flutuante e uso de memória;
- Interface de usuário consistente;
- Verificação de erros intensiva;
- Portátil para UNIX e Windows;
- Mais de cem exemplos;
- O PETSc é suportado e será ativamente aprimorado por muitos anos

Hierarquia dos componentes



Fonte: http://www.training.prace-ri.eu/uploads/tx_pracetmo/petsc_1_intro.pdf

Exemplo de aplicação

1° PASSO: Toda aplicação deve ser realizada entre os comandos:

```
static char help[] = "Solves my problem";

int main(int argc, char **args) {

     PetscInitialize(&argc, &args, (char*)0, help);
     .
     .
     .
     PetscFinalize();
     return 0;
}
```

2° PASSO: A biblioteca do PETSc deve ser incluída na classe em que será utilizada:

```
#include <petsc/finclude/petscsys.h>
#include <petsc/finclude/petscvec.h>
#include <petsc/finclude/petscmat.h>
#include <petsc/finclude/petscpc.h>
#include <petsc/finclude/petscksp.h>
#include <petsc/finclude/petscviewer.h>
Entre outras...
```

3° PASSO: Declaração das variáveis do tipo PETSc:

```
Mat                A,F;
Vec                b, u, All;
PetscErrorCode     ierr;
PetscInt           Istart, Iend, iterations;
KSP                ksp;
PC                 pc;
VecScatter         ctx;
PetscScalar        val;
```

Exemplo de aplicação

4° PASSO: Declaração dos vetores e matrizes

```
// Preallocates the matrix
ierr = MatCreateAIJ(PETSC_COMM_WORLD, PETSC_DECIDE, PETSC_DECIDE,
                    sysSize, sysSize, 120, NULL, 500, NULL, &A);
CHKERRQ(ierr);

// Divides the matrix between the processes
ierr = MatGetOwnershipRange(A, &Istart, &Iend);CHKERRQ(ierr);

//Create PETSc vectors
ierr = VecCreate(PETSC_COMM_WORLD, &b); CHKERRQ(ierr);
ierr = VecSetSizes(b, PETSC_DECIDE, sysSize); CHKERRQ(ierr);
ierr = VecSetFromOptions(b); CHKERRQ(ierr);
ierr = VecDuplicate(b, &u); CHKERRQ(ierr);
ierr = VecDuplicate(b, &All); CHKERRQ(ierr);
```

5° PASSO: Insere valores nas matrizes e vetores

INSERT_VALUES

```
ierr = MatSetValues(A,1,&dof_i,1,&dof_j,&Ajac(2*i  ,2*j  ),ADD_VALUES);
ierr = VecSetValues(b,1,&dof_i,&Rhs(2*i  ),ADD_VALUES);
```

Exemplo de aplicação

6° PASSO: Finalização da montagem das matrizes e vetores

```
//Assemble matrices and vectors
ierr = MatAssemblyEnd(A,MAT_FINAL_ASSEMBLY);CHKERRQ(ierr);
ierr = MatAssemblyBegin(A,MAT_FINAL_ASSEMBLY);CHKERRQ(ierr);

ierr = VecAssemblyBegin(b);CHKERRQ(ierr);
ierr = VecAssemblyEnd(b);CHKERRQ(ierr);
```

7° PASSO: Resolvendo o sistema linear

```
ierr = KSPCreate(PETSC_COMM_WORLD,&ksp);

ierr = KSPSetOperators(ksp,A,A);CHKERRQ(ierr);
ierr = KSPSetTolerances(ksp,1.e-7,1.e-10,PETSC_DEFAULT,1000);
ierr = KSPGMRESSetRestart(ksp, 10);

ierr = KSPGetPC(ksp,&pc);
ierr = PCSetType(pc,PCNONE);
ierr = KSPSetPCSide(ksp, PC_RIGHT);

ierr = KSPSetType(ksp,KSPLSQR);
ierr = KSPView(ksp,PETSC_VIEWER_STDOUT_WORLD);
ierr = KSPSetFromOptions(ksp);

ierr = KSPSolve(ksp,b,u);
```

Exemplo de aplicação

8° PASSO: Reunir e compartilhar a solução do sistema com todos os processadores:

```
//Gathers the solution vector to the master process
ierr = VecScatterCreateToAll(u, &ctx, &All);CHKERRQ(ierr);

ierr = VecScatterBegin(ctx, u, All, INSERT_VALUES, SCATTER_FORWARD);
ierr = VecScatterEnd(ctx, u, All, INSERT_VALUES, SCATTER_FORWARD);

ierr = VecScatterDestroy(&ctx);CHKERRQ(ierr);
```

9° PASSO: Desaloca todos os objetos criados pelo PETSc

```
ierr = KSPDestroy(&ksp); CHKERRQ(ierr);
ierr = VecDestroy(&b); CHKERRQ(ierr);
ierr = VecDestroy(&u); CHKERRQ(ierr);
ierr = VecDestroy(&All); CHKERRQ(ierr);
ierr = MatDestroy(&A); CHKERRQ(ierr);
```

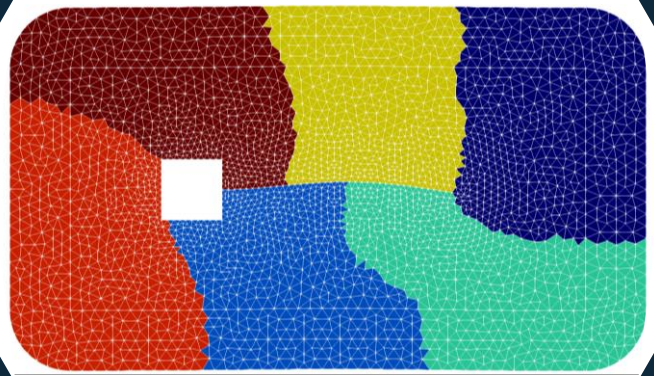
CONFIGURAÇÃO E INSTALAÇÃO DO PETSC

https://wikimechanica.eesc.usp.br/petsc

METIS

Particionar do domínio de elementos finitos para paralelização do código

O que é o METIS/ParMETIS?



O ParMETIS é uma biblioteca paralela desenvolvida em MPI que contém uma variedade de algoritmos para particionamento de gráficos não estruturados, malhas e para o cálculo de ordenações e redução de preenchimento de matrizes esparsas. O ParMETIS estende a funcionalidade fornecida pelo METIS e inclui rotinas que são especialmente adequadas para cálculos paralelos de AMR e simulações numéricas de grande escala. Os algoritmos implementados no ParMETIS são baseados nos esquemas de particionamento de grafos *k-way* em vários níveis paralelos, particionamento adaptativo e esquemas de particionamento multi-restritos paralelos.

GitHub

Desenvolver um código em equipe e compartilhar informações

Desenvolvimento compartilhado

GitHub é uma plataforma de hospedagem de código-fonte com controle de versão usando o Git. Ele permite que programadores, utilitários ou qualquer usuário cadastrado na plataforma contribuam em projetos privados e/ou Open Source de qualquer lugar do mundo.

Comandos importantes

```
git status

git add <file>

git commit -am “comentario”

git push

git pull

git checkout <branch>
```

MAIS INFORMAÇÕES?

https://wikimechanica.eesc.usp.br/github

7

Doxygen

Documentar meu código

Documentando meu código

O Doxygen é uma ferramenta para gerar documentação de códigos comentados em C++, que também suporta outras linguagens de programação populares como C, C#, PHP, Java, Python, Fortran, etc.

Estilos de comentários

```
/**
* ... text
...
*/
```

```
/*!
* ... text ...
*/
```

```
/// ... text ...
/// ... text ...
/// ... text ...
```

```
//!... text ...
//!... text ...
//!... text ...
```

MAIS INFORMAÇÕES?

https://wikimechanica.eesc.usp.br/doxygen