M342 Álgebra Computacional

Christian Lomp

FCUP

12 de setembro de 2011

1. Introdução

Aspectos computacionais de álgebra:

1. Introdução

Aspectos computacionais de álgebra:

- Representação de estruturas algébricas no computador;

1. Introdução

Aspectos computacionais de álgebra:

- Representação de estruturas algébricas no computador;
- Aritmética de inteiros, corpos racionais, polinómios numa indeterminada;

1. Introdução

Aspectos computacionais de álgebra:

- Representação de estruturas algébricas no computador;
- Aritmética de inteiros, corpos racionais, polinómios numa indeterminada;
- Aritmética de matrizes e vectores;

1. Introdução

Aspectos computacionais de álgebra:

- Representação de estruturas algébricas no computador;
- Aritmética de inteiros, corpos racionais, polinómios numa indeterminada;
- Aritmética de matrizes e vectores;
- Aritmética de polinómios em múltiplas indeterminadas;

1. Introdução

Aspectos computacionais de álgebra:

- Representação de estruturas algébricas no computador;
- Aritmética de inteiros, corpos racionais, polinómios numa indeterminada;
- Aritmética de matrizes e vectores;
- Aritmética de polinómios em múltiplas indeterminadas;
- Problemas de geometria algébrica (Bases de Gröbner).

1. Introdução

Aspectos computacionais de álgebra:

- Representação de estruturas algébricas no computador;
- Aritmética de inteiros, corpos racionais, polinómios numa indeterminada;
- Aritmética de matrizes e vectores;
- Aritmética de polinómios em múltiplas indeterminadas;
- Problemas de geometria algébrica (Bases de Gröbner).

Programação na linguagem C++.

C++ versus Python

C++ é compilada:

C++ é "strongly typed":

- variáveis têm de ser declaradas antes do primeiro uso e seu típo têm de ser especificado
- o valor da função tem ser declarado

Compiladores e IDE

Compiladores e IDE

- *Gnu CC*: compilador para todas as plataformas http:/gcc.gnu.org

Compiladores e IDE

- *Gnu CC*: compilador para todas as plataformas http:/gcc.gnu.org
- *Code::Blocks* IDE para todas as plataformas http://www.codeblocks.org/

Compiladores e IDE

- *Gnu CC*: compilador para todas as plataformas http:/gcc.gnu.org
- *Code::Blocks* IDE para todas as plataformas http://www.codeblocks.org/
- *Microsoft Visual C++* IDE para Windows; gratuito para alunos da UP https://sigarra.up.pt/up/WEB_BASE.GERA_PAGINA?P_pagina=1001525

Compiladores e IDE

- *Gnu CC*: compilador para todas as plataformas http:/gcc.gnu.org
- *Code::Blocks* IDE para todas as plataformas http://www.codeblocks.org/
- *Microsoft Visual C++* IDE para Windows; gratuito para alunos da UP https://sigarra.up.pt/up/WEB_BASE.GERA_PAGINA?P_pagina=1001525
- *Eclipse*: IDE para todas as plataformas;

Compiladores e IDE

- *Gnu CC*: compilador para todas as plataformas http:/gcc.gnu.org
- *Code::Blocks* IDE para todas as plataformas http://www.codeblocks.org/
- *Microsoft Visual C++* IDE para Windows; gratuito para alunos da UP https://sigarra.up.pt/up/WEB_BASE.GERA_PAGINA?P_pagina=1001525
- *Eclipse*: IDE para todas as plataformas;
- *Xcode*: IDE para Mac OS X

Variáveis

Variáveis

Espaço de memória reservado para armazenar tipos de dados, com um nome para referenciar seu conteúdo.

Observações importantes

- Todas as variáveis devem ser declaradas antes de serem usadas.
- Mais de uma variável do mesmo tipo: separam-se seus nomes por vírgulas.

Memória

Memória

- *1 bit* = menor unidade de informação, dois valores $\{0, 1\}$.
- *1 byte* = 8 bit $\boxed{a_0}\boxed{a_1}\boxed{a_2}\boxed{a_3}\boxed{a_4}\boxed{a_5}\boxed{a_6}\boxed{a_7}$; representa o número

$$0 \leq \sum_{i=0}^{7} a_i 2^i < 2^8 = 256$$

Tipos de dados em C++

C++ conhece os seguintes tipos de variáveis:

tipo		nº bytes	valores	
char	caracter	1 byte	$-128 \cdots 127$	(signed)
			$0 \cdots 255$	(unsigned)
short	inteiro	2 bytes	$-32768 \cdots 32767$	(signed)
			$0 \cdots 65535$	(unsigned)
int	inteiro	4 bytes	$-2.147.483.648 \cdots 2.147.483.647$	(signed)
			$0 \cdots 4.294.967.295 = 2^{32} - 1$	(unsigned)
long long	inteiro	8 bytes	$-9.223.372.036.854.775.808 \cdots 9.223.372.036.854.775.807$	(signed)
			$0 \cdots 18.446.744.073.709.551.615 = 2^{64} - 1$	(unsigned)
bool	v/f	1 byte	*true* ou *false*	
float	"real"	4 bytes	$\pm 3.4e^{\pm 38}$ (ca. 7 digitos)	
double	"real"	8 bytes	$\pm 1.7e^{\pm 308}$ (ca. 15 digits)	

Exemplo

Python

```
a=3
b=a*a+2
print a,b
```

C++

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

int main()
{
    int a=3;
    int b=a*a+2;
    cout << a << b;
}
```

Ciclo 'for'

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

int main()
{
    int n=10, factorial=1;
    cin >> n;
    for (int i=1; i<=n; i++)
      factorial=factorial*i;
    cout << factorial;
}
```

Ciclo 'while'

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

int main()
{
    int n=10, factorial=1, i=1;
    cin >> n;
    while(i<=n) {
       i++;
       factorial=factorial*i;
    }
    cout << factorial;
}
```

Ciclo 'do while'

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

int main()
{
    int n=10, factorial=1, i=1;
    cin >> n;
    do {
       i++;
       factorial=factorial*i;
    } while (i<n);
    cout << factorial;
}
```

Funções

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

int factorial ( int n )
{
   if (n==0)
     return 1;
   else
     return n*factorial(n-1);
};

int main();
{
    int n=10;
    cin >> n;
    cout factorial(n);
}
```

STL - Standard Template Library

1. Funções do input/output (io): manipulação de ficheiros, teclado, etc. (e.g. *cout, cin, ...*).
2. Funções para manipular de *strings*.
3. Classes para tratar de *listas*, *vectores*, *conjuntos*, *funções*, etc...

Classes

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

class inteiro
{
  unsigned int n;
  public:
    inteiro (unsigned int numero) { n=numero; };

    unsigned int factorial() {
      unsigned int fact=1;
      for (int i=1; i<=n; i++)   fact=fact*i;
      return fact;
    }
};

main() {
  unsigned int n;
  cin >> n;
  inteiro numero(n);
  cout << numero.factorial() << endl;
}
```

Classes

```
class triangulo {

   int a,b,c ;
   public:

   triangulo (int lado1, int lado2, int lado3) {
      a=lado1;  b=lado2;  c=lado3;
   }

   double Area() { ...  };
};

main() {
   triangulo t(3,4,5);
   cout << t.Area()<< endl;
}
```