Ferramentas Computacionais

Carlos Baquero

Grupo de Sistemas Distribuídos
Departamento de Informática
Universidade do Minho

2005/2006

Objectivos

- Homogenização de conhecimentos de informática.
  - Conceitos de hardware e infra-estructura.
  - Evolução histórica dos computadores.
- Aprendizagem e treino de uma linguagem de programação imperativa: Basic.
  - Manipulação de estado.
  - Estruturas de controlo.
  - Programação recursiva.
  - Noções de complexidade.
  - Programação como meio de exploração científica.
- Programação de textos matemáticos em LaTeX

Avaliação

A avaliação é composta por um exame e um trabalho prático opcional. O exame decorre no calendário de exames e o trabalho prático é desenvolvido na fase final do semestre, uma vez dadas as bases necessárias de programação.
Sendo a nota de exame E e a nota de trabalho T, calcula-se a nota final por:

$$max(E, E * 0.9 + T * 0.1)$$

Contactos

- Página do grupo: http://gsd.di.uminho.pt/
- Página pessoal: http://gsd.di.uminho.pt/cbm
- Email: cbm@di.uminho.pt
- Atendimento: Sexta ao início da tarde.

Conceitos e terminologia básica

Computador Máquina com capacidade de realizar rapidamente conjuntos de operações repetitivas e complexas.

Programa Sequência de operações necessárias para realizar um tarefa, i.e. obter os resultados pretendidos.

Hardware Todos os componentes físicos de um computador.

Software Todos os programas que permitem o funcionamentos dos mecanismos electrónicos que constituem o computador.

Componentes de um computador (Processador)

Processador (CPU) Responsável por executar as operações especificadas por um programa. São normalmente simples mas executadas com grande rapidez. Pode dividir-se em três unidades lógicas:

- **Unidade de Controlo (UC)** Controla as operações de processamento, ou seja, é responsável pela transmissão de sinais, que comandam o funcionamento do computador.
- **Unidade aritmética e lógica (ALU)** É responsável pelas operações fundamentais. Efectua todas as operações aritméticas e lógicas.
- **Registos internos** Memórias utilizadas para guardar resultados temporários e informação de controlo. Alta velocidade de acesso.

Componentes de um computador (Memória)

Memória Espaço de armazenamento da informação sobre a qual são efectuadas as operações do processador.

Principal, volátil, RAM Memória acessível com grande rapidez onde são guardados os dados e o programa que está a ser executado pelo processador.

Permanente (semi-permanente), ROM Só permite acesso de leitura. Por exemplo, o programa de arranque de um computador.

Componentes de um computador (Periféricos)

Periféricos (I/O Devices) Dispositivos utilizados para introduzir e extrair informação de um computador.

Entrada Teclados, leitores de cartões magnéticos, perfurados, scanners, sensores, rato, leitores de códigos de barras, ...

Saída monitor, terminais, plotter, impressora, . . .

Memória Secundária Memória com capacidade de armazenamento várias vezes superior à memória principal, mas com tempos de acesso várias vezes inferior.
Estão organizados por unidades lógicas, normalmente ficheiros.
Os dispositivos mais comuns são os discos magnéticos, tapes, discos ópticos, disketes, memórias *flash* USB, ...

Representação da Informação

BCD Binary Coded Decimal. Faz corresponder a cada algarismo decimal 4 bits.

EBCDIC Extended binary Coded Decimal Interchange Code.

ASCII American Standard Code for Information Interchange. utiliza um byte para representar um caracter. É o código mais utilizado para representação de informação (texto).

Dec	Caracter
48	0
65	A
97	a

Representação da Informação

Os operandos aritméticos são normalmente representados na base 10 (decimal). Esta base não é, no entanto, adequada à representação de valores numéricos em computadores digitais (binários).
Em computadores digitais estes costumam ser representados em base 2, ou para serem mais facilmente perceptíveis pelas pessoas em base 8 (octal) ou 16 (hexadecimal).

Decimal	Binário	Hexadecimal	Octal
0	0	0	0
1	1	1	1
2	10	2	2
3	11	3	3
4	100	4	4
5	101	5	5
6	110	6	6
7	111	7	7

Representação da Informação

Decimal	Binário	Hexadecimal	Octal
8	1000	8	10
9	1001	9	11
10	1010	A	12
11	1011	B	13
12	1100	C	14
13	1101	D	15
14	1110	E	16
15	1111	F	17
16	10000	10	20
17	10001	11	21
18	10010	12	22
...	...	...	...

Representação da Informação

Um número de *n* dígitos numa base *base* pode ser decomposto em:

$$valor_1 \times base^{n-1} + valor_{n-1} \times base^{n-2} + \cdots + valor_{n-1} \times base^1 + valor_n \times base^0$$

por exemplo 123 na base 10 pode ser decomposto em:

$$\begin{aligned} 3 \rightarrow & 3 \times 10^0 = 3 \times 1 = & 3 \\ 2 \rightarrow & 2 \times 10^{2-1} = 2 \times 10 = & 20 \\ 1 \rightarrow & 2 \times 10^{3-1} = 1 \times 100 = & +\frac{100}{123} \end{aligned}$$

Mudança de base

O número 35 na base 10 é convertido na base 2 em 100011 que é convertido na base 16

bin	10	0011
hex	2	3

em 23 e na base 8

bin	100	011
oct	4	3

em 43

Unidades de informação

bit é a unidade mais pequena de informação usada pelos computadores. A linguagem binária (base 2) é composta por uns e zeros $(1, 0)$ e é a utilizada pelos computadores.

byte Espaço necessário para guardar um caracter. 8 bits.

kilobyte (KB) 2^{10} ou 1024 bytes.

megabyte (MB) 2^{10}KB ou 2^{20} bytes.

gigabyte (GB) 2^{10}MB ou 2^{30} bytes.

terabyte (TB) 2^{10}GB ou 2^{40} bytes.

Primórdios

3000 a.c. Ábaco Chinês. Realização de operações elementares de cálculo.

1617 Tábuas de Napier. Transformam multiplicações em adições, recorrendo a tabelas de logaritmos e antilogaritmos.

1643 Calculadora Mecânica de Pascal. Realização de adições e outras operações baseadas em adições.

1673 Calculadora mecânica de Leibnitz. Realização de multiplicações e divisões sob a forma de adições e subtracções.

Primórdios

1804 Tear de Jacquard. Movimentação automática de teares através de uma série de furos feitos em cartões. Um cartão diferente para cada padrão a tecer.

1822 Engenho diferencial de Babbage. Máquina para calcular e imprimir tabelas numéricas.

1890 Tabulador estatístico de Hollerith. Foi construído nos EUA, em 1890, com o objectivo de processar os resultados do recenseamento. Registava a informação respeitante a cada pessoa sob a forma de furos feitos em cartões que eram depois contados electricamente.

Esforço de Guerra

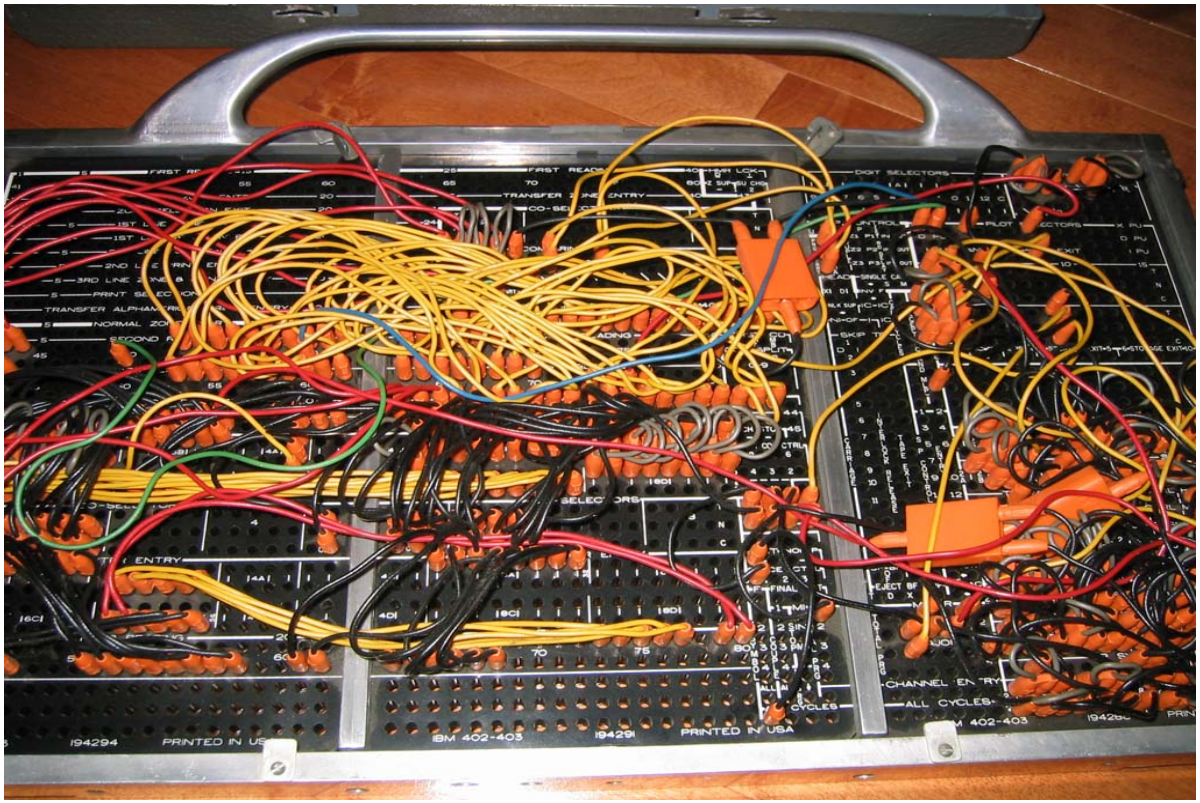
1943 Mark I de Howard Aiken. Desenvolvido com o apoio da Marinha dos EUA e IBM, apresentava uma arquitectura semelhante ao engenho analítico de Babbage. Utilizava cartões perfurados e sistema decimal. Tinha 750000 componentes ligados por 800 Km de fios condutores. Considerado obsoleto antes de estar terminado.

1943 Colossus. Máquina específica para a descodificação de mensagens. Processava 25000 caracteres por segundo, permitindo descodificar vários milhares de mensagens por dia, lidas de um fita perfurada.

1945-1955 Sistemas de Plugboard

- Eram máquinas de cálculo monstruosas muito mais lentas que qualquer computador hoje existente. Constituídos por vários milhares de tubos de vácuo.
- Eram projectados, construídos, programados e operados pelas mesmas pessoas. A programação era feita directamente na máquina muitas vezes utilizando plugboards para controlar as funções básicas das máquinas.
- Cada utilizador reservava a máquina para um determinado intervalo de tempo, inseria os plugboards necessários para realizar as tarefas pretendidas e espera algumas horas até obter um resultado, isto se nenhum dos tubos de vácuo avariasse entretanto.

Plugboard



1955-1965 Sistemas de Batch

Com o aparecimento dos transístores os computadores tornaram-se mais fiáveis, e foi possível estabelecer uma separação entre projectistas, construtores, programadores e operadores/administradores. Os programadores deixaram de ter e querer controlo absoluto sobre o computador.

1955-1965 Sistemas de Batch

Os utilizadores começavam por escrever o programa em FORTRAN ou ASSEMBLY que depois perfuravam em cartões. Em seguida entregavam os cartões aos operadores que se encarregavam de o por a correr quando a máquina estivesse livre. Este processo envolvia a leitura de cartões com por exemplo o compilador de Fortran e o próprio programa. Após estes passos o programa executava e enviava o resultado para uma impressora onde os operadores recolhiam resultado que devolviam ao programador.

1955-1965 Sistemas de Batch

Como este processo era muito demorado o passo seguinte consistiu em pre-processar os cartões guardando vários programas numa banda magnética que eram depois lidos pelo computador que efectuava as tarefas sequencialmente. Os resultados eram novamente armazenados numa banda magnética que era depois colocada num computador específico para imprimir a informação armazenada na banda magnética.

1965-1980 Sistemas de multi-tarefa/partilha de tempo

- A leitura de cartões ou de uma banda magnética era uma operação bastante demorada.
- Aplicações interactivas têm taxas de ocupação de processador da ordem dos 10 a 20% .
- Durante este período o processador não realiza qualquer tarefa.
- Um CPU parado tem um custo elevado.

Solução: Arranjar ocupação para o CPU enquanto decorrem operações de I/O $\Longrightarrow$ Partilha de CPU por vários programas. $\Longrightarrow$ **Multiprogramação**.

1980-1990 Computadores Pessoais

- Integração em larga escala. i.e, milhões de transístores num cm^2 de silício.
- Redução de custos.
- Fabrico em larga escala.
- Custos suportáveis por utilizadores domésticos $\Longrightarrow$ **Computador pessoal**.
- Capacidades de cálculo equivalentes ou superiores aos minicomputadores da geração anterior.

1980-1990 Computadores Pessoais

- Sistema Operativo:
  - MS-DOS Sistema mono-programado; mono-utilizador.
  - UNIX Sistema multi-programado; multi-utilizador.
- O utilizador é cada vez menos perito. Desenvolvimento de *interfaces* gráficas cada vez mais complexas para facilitar a interacção com utilizadores menos bem preparados.
- Os computadores são interligados (rede local) e trocam informação entre si.
- Partilha de recursos (impressoras, scanners, discos, . . .).

1990-· · · Internet

- Redes locais são ligadas entre si para formarem redes empresariais.
- Usando a rede telefónica redes de diversas empresas são interligadas formando uma rede mundial $\Longrightarrow$ **Internet**
- Para se “entenderem” os computadores têm de falar a mesma “linguagem” (protocolo).
- As aplicações começam a estar centradas na *Internet*.
- O protocolo por excelência é o `HTTP` (*hyper text tranfer protocol*).
- A os textos do WWW são codificados em `HTML` (*hyper text markup language*).
- Informação multimédia (som, imagem. . . .)

Hoje na Universidade do Minho

- Redes de computadores pessoais, em que cada um corre um determinado sistema operativo (Windows XP, Linux, MacOS). Uns desempenham o papel de servidores e outros de clientes. São bastante parecidos funcionalmente mas diferem na capacidade.
- As redes locais da Universidade estão ligadas à Internet em Portugal e no Mundo, mas nem sempre os computadores são visíveis no exterior.

Introdução à Programação

- Como indicar ao computador o que pretendemos que este realize? Como comunicar com o computador?
- Linguagem Natural? $\longrightarrow$ Linguagem Máquina? [0011 1101 10101]

Linguagem Natural

- Ambígua
- Computacionalmente complexa

Linguagem Máquina

- Computacionalmente simples.
- Pouco inteligível pelos utilizadores.
- Textos longos, logo sujeitos a erros.

Solução?

Introdução à Programação

- Como indicar ao computador o que pretendemos que este realize? Como comunicar com o computador?
- Linguagem Natural? $\longrightarrow$ Linguagem Máquina? [0011 1101 10101]

Linguagem Natural

- Ambígua
- Computacionalmente complexa

Linguagem Máquina

- Computacionalmente simples.
- Pouco inteligível pelos utilizadores.
- Textos longos, logo sujeitos a erros.

Solução? Linguagens de programação de alto nível.

Fases da resolução

Problema Alvo a automatizar.

Análise Compreensão do problema e documentação.

Especificação Síntese dos requisitos e propriedades da solução.

Algoritmo Esquema da solução.

Programa Codificação da solução.

Código Máquina Transposição para a linguagem máquina.

Execução Solução.

Algoritmo (matemática)

$$\text{Algoritmo} \stackrel{\text{def}}{=} \begin{array}{l}\text{forma da geração dos números;}\\ \text{processo de cálculo em que um certo número de regras formais resolvem, na generalidade e sem excepções, problemas da mesma natureza;}\\ \text{qualquer procedimento que permita mecanizar a obtenção de resultados de tipo determinado, podendo um resultado ser obtido por mais do que um algoritmo;}\end{array}$$

$$\text{Algoritmo} = \begin{cases}\text{Processo matemático}\\ \text{Processo numérico}\\ \text{Processo matemático mecanizável}\end{cases}$$

Algoritmo (genérico)

Algoritmo $\stackrel{\text{def}}{=}$ Método geral de resolução de uma dada classe de problemas;
Conjunto específico de passos que, seguidos exactamente, garantem sucesso na obtenção de um resultado.

Algoritmo estratégia para:

- Executar uma tarefa
- Resolver um problema
- Obter um resultado

Manuais de um equipamento; receitas culinárias; . . .

Exemplo

Problema Considerar que se pretende: Confeccionar um pudim de laranja tendo como ingredientes disponíveis: $1l$ de leite ; $1/2kg$ de açucar ; 6 ovos ; 1 laranja

Especificação

Dados

- $1l$ de leite
- $1/2kg$ de açucar
- 6 ovos
- 1 laranja

Transformação

- Raspar a laranja
- Separar as claras das gemas
- Bater as claras
- ...

Requisitos a satisfazer por um algoritmo

Ser finito

Possuindo uma representação textual, os elementos básicos deste designam-se por passos.
A execução de um algoritmo deve garantir que este termina, isto é, possui um número finito de passos.

Ser efectivo

Cada um dos passos possíveis deve ser realizável

Ser eficiente

Necessitar do menor esforço possível para a sua realização, qualquer que seja a medida adoptada, designadamente, tempo, número de passos, espaço em memória, etc.

Decomposição

O cálculo sequêncial de

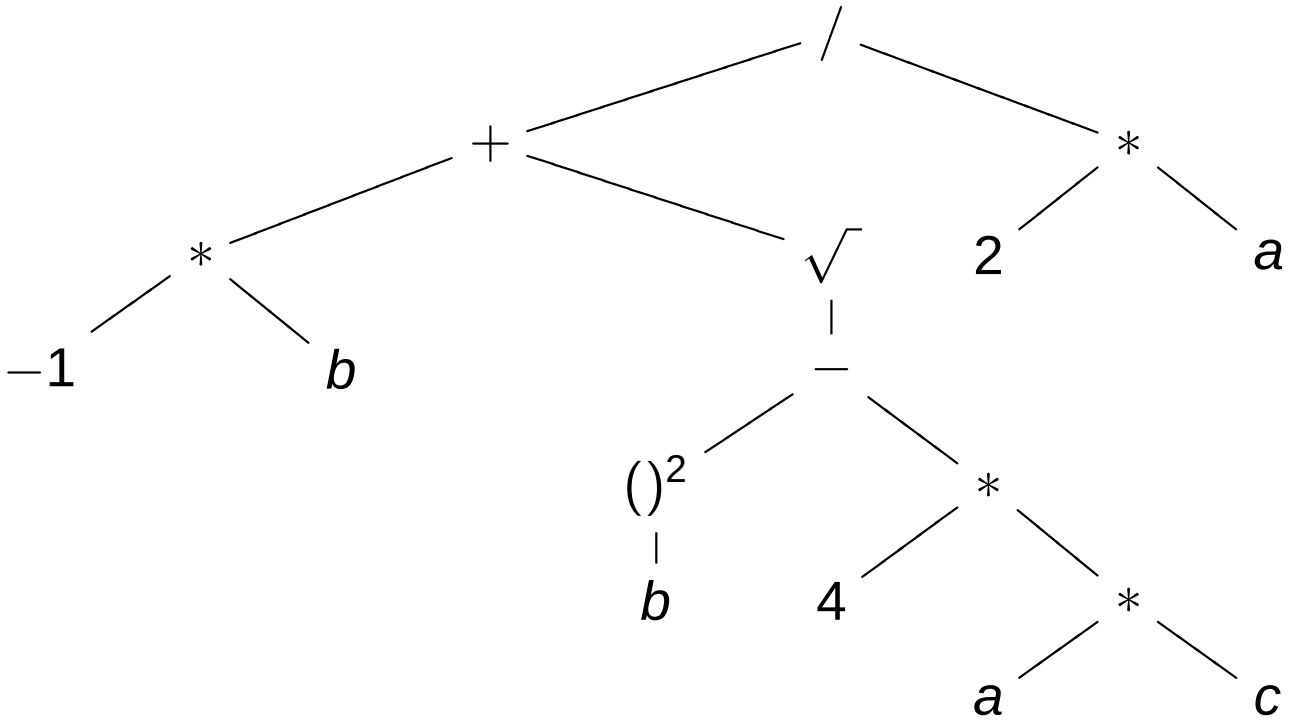
$$\frac{-b + \sqrt{b^2 - 4ac}}{2a}$$

pode ser decompostos em sub-tarefas mais simples.
Existem muitas formas diferentes de fazer esta decomposição . . . .

Uma divisão possível

$$\frac{-b+\sqrt{b^2-4ac}}{2a} \equiv calcular(-b+\sqrt{b^2-4ac});$$
$$calcular(2a);$$
$$dividir.$$

Estructura



Observação e Atribuição

Atribuição Transformação de um objecto forçando-o a possuir novos valores para os seus atributos observáveis. Pode-se representar a atribuição numa expressão

$$\alpha \leftarrow v$$

, que se lê "o atributo α passa a possuir o valor v"

Observação Consulta dos valores dos atributos de um objecto. Pode-se representar uma observação na expressão

$$\alpha = v \ , \ v \in \mathcal{V}$$

em que α é um atributo e $\mathcal{V}$ o conjunto dos seus valores possíveis.

Nota

Observação é passiva enquanto que Atribuição é activa. Nas linguagens de programação a simbologia é variável.

Atribuição e observação em BASIC

```
a=10
b=5
print b
print a
print b*a

c=b*a
print c
```

Mudança de estado ao longo do tempo

```
a=10
print a

a=5
print a

b=a-1
print b

b=b-1
print b

b=b-1
print b
```

Exercício

Coloque nas variáveis $\{a, b, c, d, e, f\}$ respectivamente os factoriais de $\{2, 3, 4, 5, 6, 7\}$.

Exercício

```
a=2
b=3*2
c=4*3*2
d=5*4*3*2
e=6*5*4*3*2
f=7*6*5*4*3*2
```

Exercício

```
a=2
b=3*a
c=4*b
d=5*c
e=6*d
f=7*e
```

Execução condicional

if-then

Se é verdadeira uma determinada condição então executa-se um bloco de instrucções.

if-then-else

Se é verdadeira uma determinada condição então executa-se um bloco de instrucções, caso contrário executa-se um outro bloco de instrucções.

Execução condicional em Basic

```
if condtrue then
  bloctrue
fi

if condtrue then
  bloctrue
else
  blocfalse
fi
```

Execução condicional em Basic

```
a=10
if a>0 then
  print "positivo"
else
  print "negativo"
fi
```

Exercício

Dado um número na variável a indicar se este é negativo, nulo ou positivo.

Exercício

```
a=...
if a<0 then print "negativo" fi
if a=0 then print "nulo" fi
if a>0 then print "positivo" fi
```

Exercício

```
a=...
if a<0 then
  print "negativo"
else
  if a=0 then
    print "nulo"
  else
    print "positivo"
  fi
fi
```

Exercícios

1. Defina um algoritmo que permita definir a acidez/basicidade de uma solução baseada no seu valor de *pH*.

	ácido	neutro	alcalino
pH	$1 \leq pH < 7$	$pH = 7$	$7 < pH \leq 14$

2. Defina um algoritmo para o cálculo dos zeros de uma equação do segundo grau.

$$\frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

   Nota: cuidado com as raízes quadradas de números negativos, e as divisões por zero.

3. Defina o algoritmo que permite saber qual a velocidade a que se pode circular numa estrada, dependendo do seu tipo.

Input / Output

Aquisição de dados Atribuição

Notação

`input` *x*

Significado

O valor de *x* é obtido a partir de uma operação de aquisição de dados (*input* via teclado).

Produção de resultados Observação

Notação

`print` *expressao*

Significado

O valor resultante do cálculo de *expressao* é comunicado para o exterior (*output* para o écran).

Iteração. While

Permite repetir um bloco de acções enquanto se verifica uma dada condição.

```
while(condtrue)
...
wend
```

Pelo menos uma das variáveis na condição terá de ser mudada no corpo do ciclo. Doutro modo o ciclo poderá não parar.

Exemplo

Calcular a soma dos números inteiros inferiores a n:
$\sum_{i=1}^{n} i$
Soma_até_n≡

```
n=...
soma=0
i=1
while (i<=n)
  soma=soma+i
  i=i+1
wend
print soma
```

Exercício

Tentar fazer resolver o exemplo anterior apenas com as variáveis `soma` e `n`.

Exercício

```
n=...
soma=0
while (n>0)
  soma=soma+n
  n=n-1
wend
print soma
```

Iteração. Repeat-Until

Permite repetir um bloco de acções até que se verifica uma dada condição.

```
repeat
...
until (condtrue)
```

Enquanto a condição se mantem falsa o ciclo continua.

Exemplo

Calcular a sequência $\frac{1}{1} - \frac{1}{2} + \frac{1}{3} - \frac{1}{4} + \ldots$

```
c=1
s=1
repeat
  l=c
  s=-(s+sig(s))
  c=c+1/s
  print c
until(abs(l-c)<0.001)
```

Exercício

Calcule a seguinte progressão, usando a forma repeat-until:

$$\sum_{n=1}^{10} \frac{1}{2^n}$$

Iteração. For-Next

Permite repetir um bloco de acções um determinado número de vezes e a uma dada cadência.

```
for var=... to ... step ...
...
next var
```

Parâmetros

`for var` = início `to` fim `step` passo.

Exemplos

Soma_até_n ≡

```
n=...
s=0
for i=1 to n
  s=s+i
next i
```

Produto_até_n ≡

```
n=...
p=1
for i=1 to n
  p=p*i
next i
```

Exercícios

1. Calcular o Quociente da divisão de dois números inteiros.
2. Calcular o factorial de um número.
3. Calcular os factoriais de todos os inteiros inferiores a um dado número.
4. Calcular e imprimir a tabuada da multiplicação.

Soluções - Quociente

```
input x
input y
q=0
while (x>=y)
  x=x-y
  q=q+1
wend
print "O Quociente é:", q
```

Soluções - Factorial

```
input n
if n<0 then print "Não existe"
elseif n=0 then print "O factorial de 0 é 1"
else
  i=1
  fact=1
  while(i<=n)
    fact=fact*i
    i=i+1
  wend
  print "O factorial de ",n," é ",fact
fi
```

Soluções - Factorial

```
input n
fact=1
for i=1 to n
  fact=fact*i
next i
print "O factorial de ",n," é ",fact
```

Soluções - Factorial_n

```
input n
j=1
while (j<=n)
  i=1
  fact=1
  while (i<=j)
    fact=fact*i
    i=i+1
  wend
  print "O factorial de ",j," é ",fact
  j=j+1
wend
```

Soluções - Factorial_n

```
input n
for j=1 to n
  fact=1
  for i=1 to j
    fact=fact*i
  next i
  print "O factorial de ",j," é ",fact
next j
```

Soluções - Factorial_n

```
input n
fact=1
for i=1 to n
  fact=fact*i
  print "O factorial de ",i," é ",fact
next i
```

Vectores/Arrays e estruturas n-dimensionais

Alguns dos seguintes problemas requerem (ou têm a sua resolução facilitada) forma de manipular grupos (indexáveis) de valores:

- Como calcular a média das idades de todos os presentes?
- Como determinar qual a pessoa mais velha, e a mais nova de todos os presentes?
- Como calcular as notas de todos os alunos sabendo que um teste é composto por 20 perguntas?
- Como calcular a média das notas?
- Como saber quantos alunos foram aprovados? e quantos não foram aprovados?
- Como somar dois vectores? como somar e multiplicar matrizes?

Uma solução sem vector

```
maxim=0
repeat
  input idade
  if (idade > maxim) then maxim=idade fi
until (idade=0)
print maxim
```

Solução com um vector

```
input n
dim idade(n)
maxim=0
for i=1 to n
  input idade(i)
next i
for i=1 to n
  if (idade(i) > maxim) then maxim=idade(i) fi
next i
print maxim
```

Cálculo da cota mínima num terreno

```
dim grelha(5,6)
for i=1 to 5
  for j=1 to 6
    input grelha(i,j)
  next j
next i
cotamin=grelha(1,1)
for i=1 to 5
  for j=1 to 6
    if (grelha(i,j) < cotamin) then
      cotamin=grelha(i,j)
    fi
  next j
next i
print cotamin
```

Exercícios

1. Calcular as notas do teste com 10 perguntas de uma turma de 20 alunos.
2. Obter as temperaturas mensais de uma localidade e calcular:
   1. A temperatura máxima;
   2. Número de ocorrências da temperatura máxima;
   3. A temperatura mínima;
   4. A média das temperaturas.
3. Calcular o bónus a atribuir a cada vendedor sabendo que:
   - o que tiver maior volume de vendas tem um bónus de 20% do seu total de vendas.
   - Vendedores que tiverem mais de 75% do valor do melhor têm um bónus de 10% do seu total de vendas
   - Vendedores que tiverem de 50% a 75% do valor do melhor têm um bónus de 5% do seu total de vendas.
   - Os restantes não têm qualquer bónus.

Ordenação de Arrays

Rearranjo de um conjunto de dados de acordo com um critério (ordem) específica.

1. Ordenação por selecção directa;
2. Ordenação por permutação (Bubblesort).

Ordenação por selecção directa

O algoritmo consiste em trocar o menor elemento do array pelo elemento que ocupa a sua posição

```
for i=1 to n-1
  k=i
  item=a(i)
  for j=i+1 to n
    if a(j) < item then
      k=j
      item=a(k)
    fi
  next
  a(k)=a(i)
  a(i)=item
next
```

Ordenação por permutação (Bubblesort)

O algoritmo consiste em percorrer sucessivamente o array deslocando os menores elementos para o início do array.

```
for i=2 to n
  for j=n to i step -1
    if a(j-1) > a(j) then
      item=a(j-1)
      a(j-1)=a(j)
      a(j)=item
    fi
  next
next
```

Funções

Exemplos (matemática)

- $+: \mathbb{N} \times \mathbb{N} \longrightarrow \mathbb{N}$
- $-: \mathbb{N} \times \mathbb{N} \longrightarrow \mathbb{Z}$
- $\times: \mathbb{N} \times \mathbb{N} \longrightarrow \mathbb{N}$
- $\div: \mathbb{N} \times \mathbb{N} \longrightarrow \mathbb{Q}$
- $\sqrt{x}: \mathbb{R}_0^+ \longrightarrow \mathbb{R}_0^+$
- $x^2 : \mathbb{R} \longrightarrow \mathbb{R}_0^+$
- $x^2 + y^2 : \mathbb{R} \times \mathbb{R} \longrightarrow \mathbb{R}_0^+$

Funções - Exemplos

Exemplos

- Ler_array: $\phi \longrightarrow array(\ldots)$
- Maximo_array: $array(\ldots) \longrightarrow \mathbb{N}$
- Minimo_array: $array(\ldots) \longrightarrow \mathbb{N}$
- Ordenar_array: $array(\ldots) \longrightarrow array(\ldots)$
- Calculo_notas: $array(\ldots, \ldots) \longrightarrow array(\ldots)$

Concretizando em YaBasic

O seguinte exemplo, mostra quer a declaração de uma sub-rotina/função, quer o uso de variáveis locais.

```
print multiply$("Hello",3)

sub multiply$(a$,a)
 local b$,b
 for b=1 to a
  b$=b$+a$
 next b
 return b$
end sub
```

Resultado: HelloHelloHello

Scope – Visibilidade de variáveis

```
a=2
print a

sub b()
local a
a=3
print a
end sub

b()

print a
```

Resultado: 2 3 2

Scope – Visibilidade de variáveis

```
a=2
print a

sub b()
a=3
print a
end sub

b()

print a
```

Resultado: 2 3 3

Factorial recursivo

```
sub fact(n)
   if (n=1) then return n
   else return n*fact(n-1)
   endif
end sub

print fact(10)
```

Resultado: 3628800

Calculo da constante *e*

$$e = 1 + \frac{1}{1!} + \frac{1}{2!} + \frac{1}{3!} + \frac{1}{4!} + \cdots$$

```
e=1
for n=1 to 8
e=e+1/fact(n)
print e
next
```

Resultado:

```
2
2.5
2.66667
2.70833
2.71667
2.71806
2.71825
2.71828
```

Exercício - Fibonacci

Implemente uma versão recursiva do algorítmo de Fibonacci.

```
sub fib(a)
if (a=1) then return 1
elseif (a=2) then return 2
else
i=1
j=2
while(a>2)
  k=i+j
  i=j
  j=k
  a=a-1
wend
return k
endif
end sub
print fib(5)
print fib(15)
```

Exercício - Fibonacci recursivo

```
sub fib(a)
if (a=1) then return 1
elseif (a=2) then return 2
else return fib(a-1)+fib(a-2)
endif
end sub
print fib(5)
print fib(15)
```

Resultado:

```
8
987
```

Exercício - Fibonacci genérico

Adapte a rotina de fibonacci e faça um programe que pede ao utilizador os valores iniciais da série e quantos parcelas da série devem ser impressas.

Exercício - Fibonacci genérico

```
input "Primeiro termo:" p
input "Segundo termo:" s
input "Tamanho da série:" t
sub fib(a)
if (a=1) then return p
elseif (a=2) then return s
else return fib(a-1)+fib(a-2)
endif
end sub
for n=1 to t
print fib(n)
next
```

Crivo de Eratostenes

Em 200 AC o matemático grego Eratostenes inventou um método para o calculo de sequências de primos.
A lista de primos menores do que N é obtida cancelando todos os múltiplos até $\sqrt{N}$.

```
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20
1 2 3   5   7   9    11    13    15    17    19
1 2 3   5   7        11    13          17    19
```

Neste exemplo foram retirados os multiplos de 2 e 3, uma vez que 5 já é maior do que a raiz de 20.
Exercício: Construa um programa que calcula todos os primos até um dado *n* de input.

Exercício - Factorização

- Construir uma função `isprime(num)` que devolve 1 se o número `num` for primo.
- Decompor um dado número *x* obtido por *input* nos seus factores primos.

Rotina isprime

```
sub isprime(num)
   for n=int(sqrt(num)) to 2 step -1
      if mod(num,n)=0 then return 0 fi
   next n
return 1
end sub
```

Decompor em factores primos

```
input a
i=int(sqrt(a))
repeat
   if mod(a,i)=0 and isprime(i)=1 then
      print i," ";
      a=a/i
      i=int(sqrt(a))
   else
      i=i-1
   fi
until (i=1)
print a
```

Gráficos em Yabasic

- `open window x,y`. Abre uma janela para desenho de imagens monocromáticas com x por y pixels.
- `clear window`. Limpa a janela de desenho, caso seja necessário.
- `dot x,y`. Marca um ponto na posição x,y.
- `line x,y to i,j`. Desenha uma linha da origem (x,y) ao destino (i,j).
- `circle x,y,r`. Desenha um circula com centro em x,y e raio r.

Monte Carlo chega a π

```
open window 400,400
do
  x=ran(400):y=ran(400)
  total=total+1
  if (sqrt((x-200)^2+(y-200)^2)<200) then
   in=in+1
   dot x,y
  fi
  print 4*in/total
loop
```

Visualização de funções continuas

A visualização de funções matemáticas (do tipo $\sin x$, $\cos x$, ...) é uma das utilizações típicas das máquinas de calcular gráficas.
Os programas residentes nestas máquinas têm de efectuar uma série de procedimentos por forma a determinar a melhor estratégia de visualização para cada função.
É possível programar em poucas linhas uma estratégia simplista, que funcionará adequadamente para algumas funções menos complexas.

Estratégia básica

Supondo que se faz a visualização num plano de 500 por 500 pixels, o gráfico da função será feito à custa da determinação das coordenadas de 500 pontos que correspondem a identico número de amostras na gama de valores nas quais se pretende analisar a função.
Ex: Se a gama a analizar for $x : [-100, 1900]$ as 500 amostras serão feitas com x a ”avançar”de quatro em quatro a partir de zero.

Inicialização

O primeiro passo é definir a função a visualizar.
Por exemplo $\sin x$:

```
sub func(x)
   return sin(x)
end sub
```

Escolha da gama

Seguidamente define-se a gama de x e calcula-se qual o segmento de x que corresponderá a um pixel.

```
open window 500,500

xi=-2*3.14
xf=2*3.14
xd=(xf-xi)/500
```

Determinação da gama de *y*

A gama de *y* é determinada à custa de uma primeira passagem de amostragem:

```
ymin=func(xi)
ymax=func(xi)

for n=0 to 499
   x=xi+n*xd
   y=func(x)
   if y<ymin then ymin=y fi
   if y>ymax then ymax=y fi
next n
print "ymin=",ymin," ymax=",ymax
```

Desenho dos pontos

```
yd=ymax-ymin

for n=0 to 499
   x=xi+n*xd
   y=func(x)
   dot n, (y-ymin)/yd*500
next n
```

Desenho de linhas

```
yd=ymax-ymin

dot 0, (func(0)-ymin)/yd*500
for n=1 to 499
   x=xi+n*xd
   y=func(x)
   line to n, (y-ymin)/yd*500
next n
```

Exercícios

- Definir uma segunda função e fazer a apresentação das duas funções com o desenho de pontos.
- Fazer a apresentação das duas funções com o desenho de linhas.
- Tentar identificar as limitações deste programa.

Fractais Naturais

Um definição possível de fractal passa pela observação de *auto-semelhança a diferentes escalas*.
A Natureza é prolifera em exemplos de fractais.
A exemplificação da existência de dimensões factais pode ser obtida pela comparação da estimação do perímetro de um circulo pela soma dos lados de sucessivos polígonos inscritos, com a tentativa da aplicação do mesmo método para o calculo do perímetro de uma linha de costa numa ilha.
As linhas de costa possuem uma dimensão de aproximadamente 1.2 (Mandelbrot e L. F. Richardson).

Conjunto Mandelbrot

Um dos fractais matemáticos (não naturais) mais conhecido é o conjunto de Mandelbrot.
É obtido analisando pontos p num plano complexo e observando se a série $z_0 = 0; z_n = z_{n-1}^2 + p$ ultrapassa as fronteiras de um círculo de raio 2 centrado no ponto $(0, 0)$.
O comportamento desta série concretiza uma orbita fractal.
Ex: O ponto $.37 + .4i$ escapa para fora do círculo, enquanto o ponto $.37 + .2i$ não consegue ultrapassar este círculo ao fim de 100 iterações.

Criação Computacional do Plano Mandelbrot

Para se visualizar este fractal é necessário mapear uma área do plano complexo numa janela de pontos no monitor.
A visualização do plano circunscrito entre $-2-2i$ e $2+2i$ numa janela com 500 por 500 pontos, passa pela identificação do ponto complexo que corresponde a cada ponto da janela.
Na prática, para cada ponto complexo torna-se necessário determinar se ele ultrapassa o distância de 2 ao centro de coordenadas após um dado número máximo de iterações.

Varrimento do espaço

```
sub escape(a,b)
   if a*a+b*b<2 then return 1 else return 0 fi
end sub

open window 500,500

xi=-2 : xf=2 : xd=(xf-xi)/500

yi=-2 : yf=2 : yd=(yf-yi)/500

for i=0 to 499
   for j=0 to 499
      x=xi+i*xd
      y=yi+j*yd
      if (escape(x,y)=1) then dot i,j fi
   next j
next i
```

Orbita de Escape de Mandelbrot

```
sub escape(a,b)
   zr=0 : zi=0
   r=30
   zr2=zr*zr : zi2=zi*zi
   while (r>0 and zr2+zi2 < 4)
      zi=2*zr*zi  : zr=zr2-zi2
      zr=zr+a     : zi=zi+b
      zr2=zr*zr   : zi2=zi*zi
      r=r-1
   wend
   if r=0 then return 1 else return 0 fi
end sub
```

Exercício

- Modificar o programa por forma a fazer *zoom* de uma zona à escolha. Tal pode ser feito pedindo novas corrdenadas para (x_i, x_f) e (y_i, y_f) ou adaptando o código de exemplo no manual Yabasic, zona função `mouseb`, por forma a definir com o rato a zona a ampliar.

Template

Template

Listings