Escola de Engenharia
Universidade do Minho

Campus de Gualtar
4710-057 Braga

Departamento de
Informática

Gonçalo Alonso Pereira - 49323 25/02/08

- Introdução do mini projecto
- Introdução da primeira ficha prática.

→ Programa a leccionar.

- Passos da compilação de um programa.
  - Passagem pelo cpp e gcc
  - Criação do programa objecto.
  - Passagem no linker e criação do executável
- Escrita de um programa em C
  - Utilização de um editor de texto.
  - Funções em C (tipo de retorno, nome, lista de parâmetros, corpo de instruções).
  - A função main.
  - Introdução ao conceito de variável (sua declaração numa função).
  - Conceito de memória principal.
  - Tipos de dados e seus tamanhos.
  - Bibliotecas e num programa.
  - Simples programa exemplificativo (calcular maior de 2 números).
  - Instrução if. (condições e instruções)
  - As funções printf e scanf.
  - Condição em C. (Operadores relacionais, Operadores lógicos).

Escola de Engenharia
Universidade do Minho

Campus de Gualtar
4710-057 Braga

Departamento de
Informática

Gonçalo Alonso Pereira - 49323 25/02/08

- Introdução do mini-projecto
- Introdução da primeira ficha prática
  - → Programa a leccionar.
- Passos da compilação de um programa.
  - Passagem pelo cpp e gcc
  - Criação do programa objecto.
  - Passagem no linker e criação do executável.
- Escrita de um programa em C
  - Utilização de um editor de texto.
  - Funções em C (tipo de retorno, nome, lista de parâmetros, corpo de instruções).
  - A função main.
  - Introdução ao conceito de variável (sua declaração numa função).
  - Conceito de memória principal.
  - Tipos de dados e seus tamanhos.
  - Bibliotecas e num programa.
  - Simples programa exemplificativo (calcular maior de 2 números).
  - Instrução if. (condições e instruções)
  - As funções printf e scanf.
  - Condição em C. (Operadores relacionais, Operadores lógicos).

Escola de Engenharia
Universidade do Minho

Campus de Gualtar
4710-057 Braga

Departamento de
Informática

Pedro Fortunato 52851

```
int maior3 (int a, int b, int c)
{
 if (a>b)
    {
     if (a>c)
        res = a;
     else
        res = c;
    }
 else
    {
     if (b>c)
        res = b;
     else
        res = c;
    }
return res;
}
```

Escola de Engenharia
Universidade do Minho

Campus de Gualtar
4710-057 Braga

Departamento de
Informática

```
#include <stdio.h>
int main( )
{
int n,s,t;
printf("Introduza 3 inteiros separados por espaço:");
scanf("%d□%d□%d",&n,&s,&t);
printf("res=%d\n",maior3(n,s,t));
return 1;
}
```

— // —

```
While (condição)
{
 bloco de instruções
}
```

Escola de Engenharia
Universidade do Minho

Campus de Gualtar
4710-057 Braga

Departamento de
Informática


```
#include <stdio.h>
int main ( )
{
 int min, max, i, n;
 printf ("N vezes:");
 scanf ("%d", &n);
```

```
#include <stdio.h>
int main ()
{
 int limit, maior, valor;
 printf ("... limite:");
 scanf ("%d", &limit);
 printf ("... primeiro:");
 scanf ("%d, &maior);
 limit--;
 While (limit)
 { printf ("... valor:");
 scanf ("%d", &valor);
 if (valor > maior) maior = valor;
 limit--;
 }
 printf ("o maior é: %d \n", maior);
}
```

Escola de Engenharia
Universidade do Minho

Campus de Gualtar
4710-057 Braga

Departamento de
Informática

```
#include <stdio.h>
int main()
{
int lim, val = 0
printf("... limite:"); scanf("%d, &lim);
While (val =< lim)
{
printf ("%d", val);
val + = 2
}
```

Programa para exibir os códigos dos caracteres ASCII códigos → letras

Nota: código C

lê os caracteres
em vez dos decimais

```
#include <stdio.h>

int main ( )
{
  char letra = 'a';
  while (letra <= 'z')
  {
    printf ("%c = %d \n", letra, letra);
    letra ++;
  }
  return 1;
}
```

equivale → letra = letra + 1

```
#include <stdio.h>
int main ( )
{
  int valor = 1 , soma = 0;
  while (valor < 100)
  {
    soma = soma + valor;
    valor += 2
  }
  printf ("Soma = %d \n", soma);
  return 1;
}
```

equivale → soma += valor;

MVC

M - modelo de dados

V - interface

C - controlo

```
#include <stdio.h>
int main( )
{
  printf ("Soma = %d \n", somaw(100));
```

```
int Somaw (int v)
{ int n=1, res=0
  while (n<=v)
  {
    res+ = n
    n+=2
  return res;
  }
```

recursiva

```
int somaw (int v, int n)
{
  if (n<=v) return (n+ somaw (v, n+2))
  else
    return 0
```

28-2 Júlio Cesar Sousa

```
if (condição)
  {
  bloco de instruções
  }
else
  {
  ...
  }

while (condição)
  {
  ...
  }

switch (var)
  {
  case expr: bloco instru.
             break;          → (para sair)

  case expr2: ...
              break;

  default: ...
  }
```

igualdade	==
desigualdade	!=
maior / maior igual	> / >=
menor / menor igual	< / <=
negação	!
0 (zero)	falso
> 0	Verdadeiro
e	&&
ou	||

```
char ofcao
ofcao = getchar( )

Switch (ofcao)
{
    case '1' : Accao1
               break;
    case '2' : Accao 2;
               break
    case '3' : Accao 3;
               break;

    default : printf ("ofcao desconhecida:")
}
```

Se

```
ofcao = getchar( );
```

printf ("%c", ofcao) → lê o caracter

printf ("%d", ofcao) → lê o codigo do caracter

'ó' - 49

A - 65

'á' - ?

2008-03-03

T1

16 balões

12 balões mdc = 4

Linguagem Algoritmica

Atribuição:

x ← 5 + a * b

Controlo

Condição

se { ...

senão { ...

Enquanto condição

{ -
-

Leitura

y ← ler()

Escrita

escrever(...)

mdc (a, b)

se (a > b)

se ((a % b) == 0) → b

senão → { mdc (b, a % b)

12	8
8	4

18	16
16	2

senão

{ se ((b % a) == 0) → a

senão → { mdc (a, b % a)

24	10
14	10
10	4
6	4
4	2

24	10
10	4
4	2

pot (a, b) a^b

{ se (b == 0) → 1
senão → a * pot (a, b - 1)

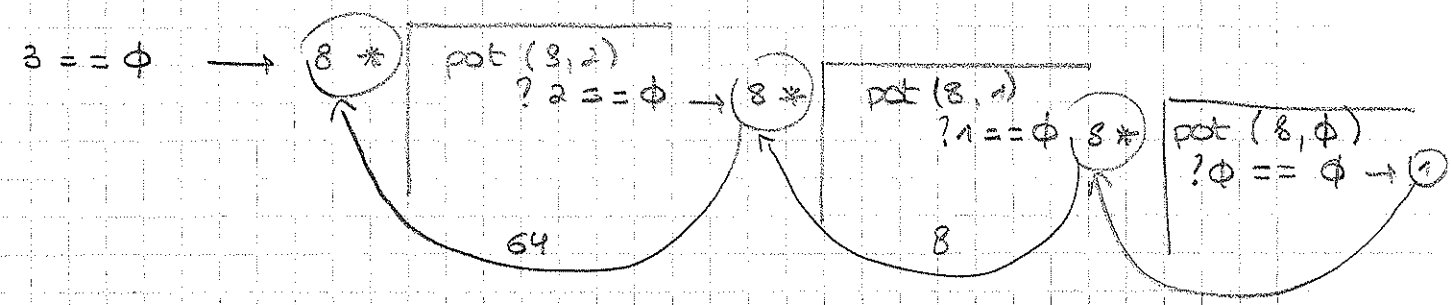
pre-condição: b >= 0

pot-it (a, b)

{ res ← 1
enq (b != 0)
{ res ← a * res
b ← b - 1

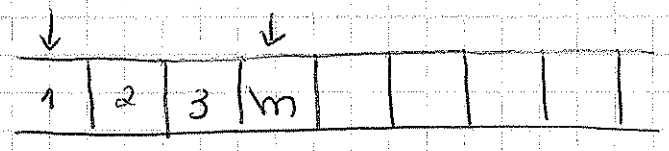
retorna (res)

pot (8, 3)



a	b	res
8	3	1
	2	8
	1	64
	0	512

Leitura de dados



buffer entrada

i
123

```
int i;
xanf("%d", &i)
```

xanf ("%*c")

Manta Pereira nº 54057 03/03/2008

divide (a, b) → (só podemos usar as operações ⊕ e ⊖)

```
se (a < b) → 0
senão ⟶ divide (a - b, b) + 1
```

testando:

18 6 → 1 + 12 6 → 1 + 6 6 → 1 + 0 6

0

3

divide (a, b) → sem recursividade

```
res <— 0
enq. (a >= b)
    { res <— res + 1
    { a <— a - b
retorna (res)
```

$$fact(n) = \begin{cases} n = 0 \longrightarrow 1 \\ n > 0 \longrightarrow n * fact(n-1) \end{cases}$$

Definir a função fact usando:

main ()

- ler um número
- calcular o factorial
- escrever o fact.

```
main ()
{
 int n, res;
  res = 1;
  printf ("Diga um n para calcular o fatorial");
  scanf ("%d", &n);
 res = fat(n);
 } printf ("O fatorial é %d", res);
```

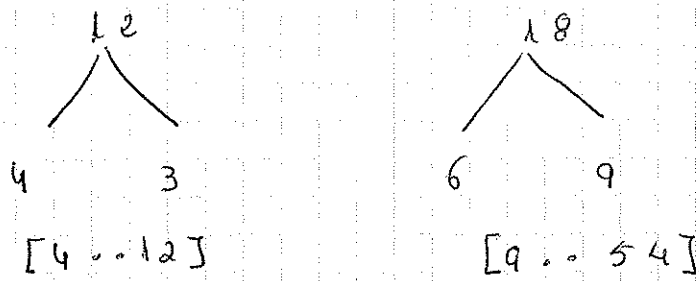
Definir a função fat usando recursividade:

```
fat (n)
{
 if (n == 0) return 1;
 else    return (n * fat(n-1))
 }
```

Definir a função fat sem usar recursividade:

```
int  fat (int n)
{
 int  res = 1
while (n)
{
  res = res * n;
  n = n - 1;
}
return (res);
}
```

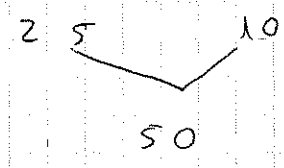
mmc (a, b)



32 24

64

96



4 3

8

12

Função iterativa:

```
res <— a
enq ( res % b )
  res <— res + a
retorna res
```

Luís Miguel Pereira de Castro

Nº: 54049

Função em C:

```
tipo nome (tipo1 arg1, tipo2 arg2)
{
    return exp;
}
```

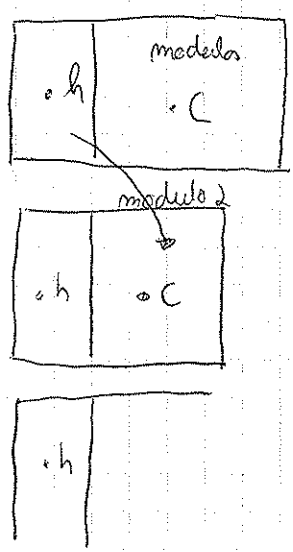
teste ? acção se V : acção se F

```
int e_dígito (char c)
{
return ((c>='0')&&(c<='9')) ? 1 : 0;
}
```

exp:

```
int maior (int a, int b)
{
return (a>b) ? a : b;
}
```

Módulos em C



bool.h

```
#ifndef _bool
typedef int boolean;
#define TRUE 1
#define False 0

#endif
```

m1.C

```
#include "bool.h"
:
```

proj.C

```
#include "bool.h"
int main ()
:
```

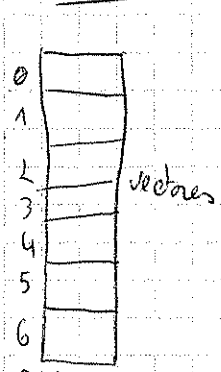
→

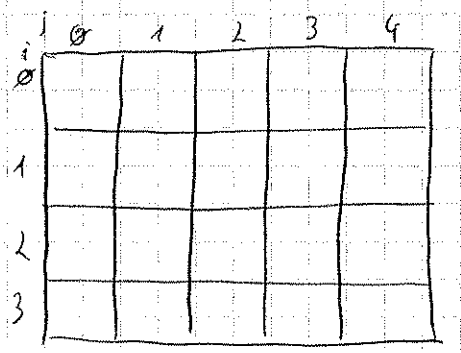
prog.c

```
#include "bool.h"
int main ()
{
  char c
  c = getchar ();
  if ( e-digito (c) )
      printf ("É um digito");
}
```

```
boolean e-digito (char c)
{
return (...) ? TRUE : FALSE;
}
```

ARRAYS





Declaração

tipo nome [nº de células] ;
[m] [m]
↑ ↑
dimensão da coluna
da linha

declarar 10 inteiros:

```
int A [10];
int B [4][6];
```

Acesso

```
x = A[3] + 5;
y = B[2,5] * 2;
```

Inicialização:

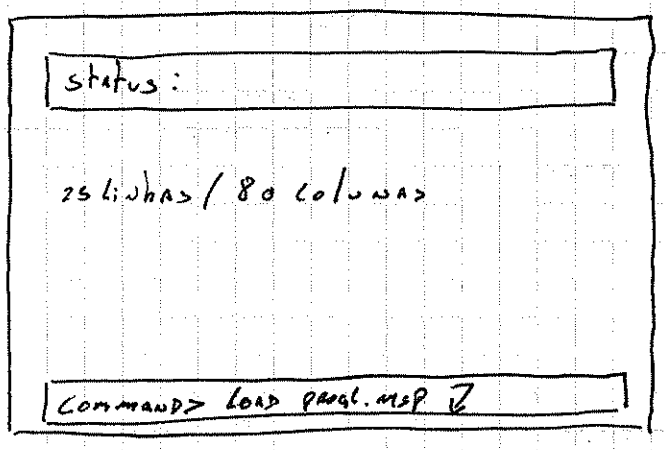
```
int A [10] = {1, 2, 3, 4, 5, 6, 7, 9, 9, 10}
int B [][] = {{1, 2, 3, 4, 5, 6}, {7, 9, 9, 10, 11, 12}, ...};
```

```
#include <stdio.h>
int main()
{
    int Valores[10], cont = 0, soma = 0;

while (cont < 10)
{
    printf("\n num: ");
    scanf("%d", &Valores[cont]);
    cont++;
}

while (cont > 0)
{
soma += Valores[cont - 1]
cont--;
}
printf("A media e' %f\n", soma/100.0);
}
```

10/3/2008

PEDRO ALEXANDRE MARQUES LOBATO 51856

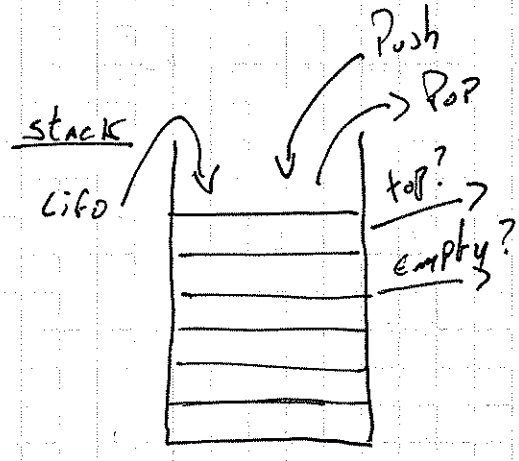
Projecto

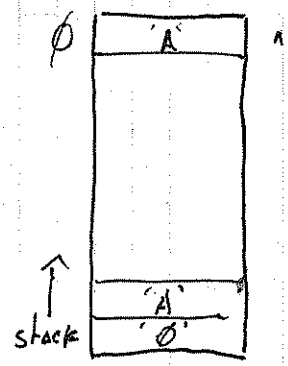


load <ficheiro>
save <ficheiro
exit
history

Prog1. msp (exemplo)

Memoria De Dados
x Ø Tam 1
CODIGO
PSHA x
INC
STORE
PUSH x
OUTC
HALT





Arrays

```
char linha[maxlinha +1]
```

```
#Define maxlinha 80
#Define maxprog. 2048
char programa[maxprog][maxlinha +1]
...
gets(linha); // scanf("%s", linha);
```

Programa MSP::[linha]
linha::[char]

*string

'\0' — ∅ → código
Fim da string

String

string.h

```
#include <string.h>
int strlen(char *s);
```

endereço de memória

```
int strlen (char *s)
{ if (s[0] == '\0') return 0;
  else
     return 1+strlen(s+1);
}
```

```
int res = 0, i = 0
while( s[i] != '\0' )
{
    res = res+1
    i++;
} return res;
```

Nuno Duarte nº 52821

PI (TP) 10/03/08

- Ler uma String e verificar se é capicua

```
int capicua (char *s)
{
  int res = 1;
  int l = strlen (s);
  int n = 0;
while (res && (n <= l/2))
{
  if s[n] != s[l-n-1]  res = 0;
    n++;
}
Return res;
}
```

- Ler uma String em numeração Romana e converter para decimal

```
int RomanoArabo (char *s)
{
  int ant=0, soma=0, t=0;
while (t < strlen s)
  {
    if ((valor s[t] >= ant)
        soma += valor s[t]
    else
        soma -= s[t];
    ant = valor s[t];
     t++;
  }
Return soma;
}
```

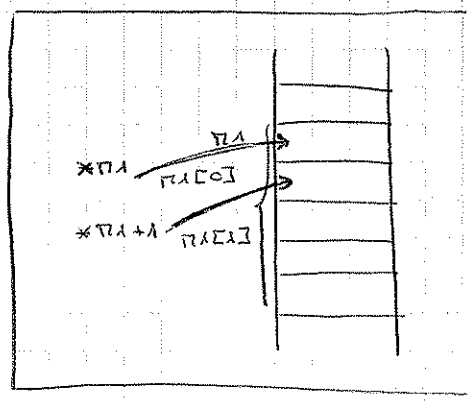
```
int valor (char c)
{
  swtch c
      I : return 1
      V : return 5
      X : return 10
      .
      .
      .
}
```

Ana Isabel Coelho Dias
48391

. Tecleica
13-3-08

→ Void

```
void listaAeeay(iut A[], iut diue)
{
  iut i=0;

  while (i<diue)
  {
    peiutf("%d\u", A[i]);
    i++;
  }
}
#doefiue nAXi 5
iut uaiu()
{
  iut n1[nAXi] = {1,2,3,4,5}, n2[10] = {1,2};
  ...
  listaAeeay(n1, nAXi);
  ...
  listaAeeay(n2, 10);
  ...
}
```

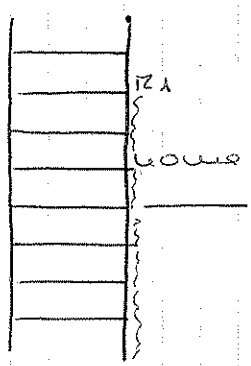


——//——

→ gets

```
chae uoue[20];
  ...
gets(uoue);
```

$ 20chaes, codigo, 2



Memória de dados

Memória de código

→ fgets

fgets (stdiu, vaeiável, nº máx chaes)

```
void limpa_buffer()
{
  while (getchar()!='\n');
}
```

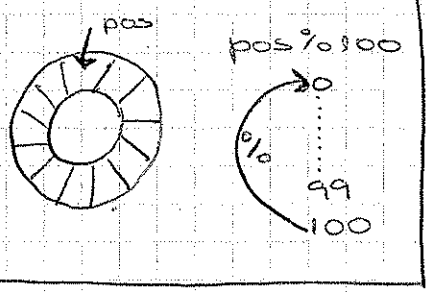
11

```
status: fich... ; xlinha
30.
:
:
54.
prompt > 3090
```

```
#define MAXLINHA 81
#define MAXH 100

char Historial[MAXH][MAXLINHA];
char comando[MAXLINHA];

fgets (stdin, comando, MAXLINHA-1);
strcpy (Historial[0], comando);
```

String.h
- strcpy
- strcmp

Round Robin

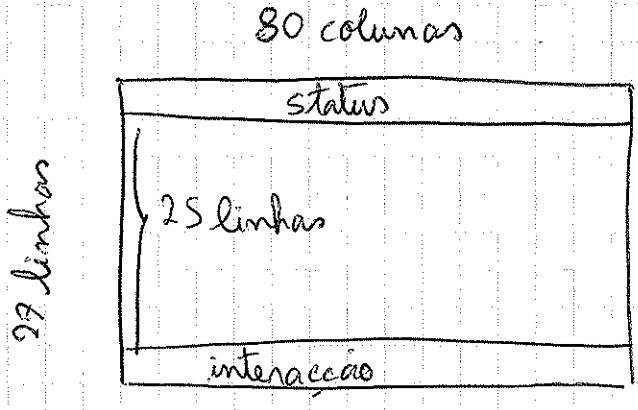


31/03/08 Nº 52853

Model

View

Ccontrol



Ordenação

ordenado

2
7
13
14
15
16
17
21
23
24

Nº de operações para encontrar um elemento numa lista ordenada

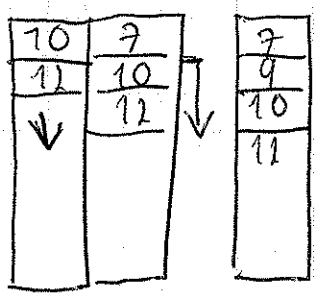
4 - 2

8 - 3

16 - 4

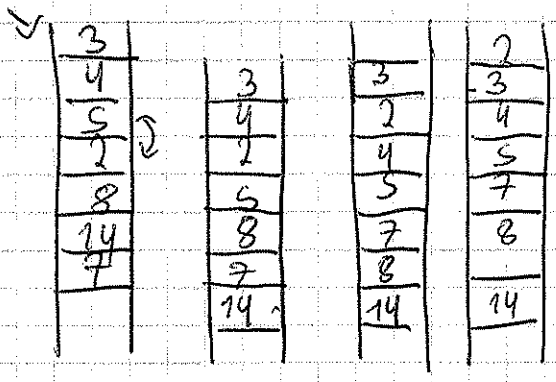
32 - 5

$\log_2 32 = 5$



→ Inserção Ordenada

Ordenar uma lista



Termina quando não ouver trocas

Definir tipos em C

typedef tipo-Base tipo-novo;

```
# define MAXDIM 10;
typedef int boolean;
typedef int Vector[MAXDIM];
.
.
.
Vector V1;
```

Algoritmo de inserção ordenada

```
void init (Vector V) // inicializar vector
{
  int i;
  for (i=0; i < MAXDIM; i++)
      V[i] = -1;
}

int posLivre (Vector v)  // procura a primeira posição livre
{
  int i = 0;
  While (V[i] != -1)
      i++;
  return i;
}
```

for (inicialização, cond, acção)

```
void inserin_ord(vector v, int elem)
{
    int i=0;
    while ((v[i] < elem) && (v[i] != -1))
        i++;
    if (v[i]==-1)
        v[i] = elem;
    else
    {   j = golire(v);
        while (j>i)
        {
            v[j]=v[j-1];
            j--;
        }
        v[j]=elem;
    }
}
```

03/04

Referência

valor

```
int soma (int a, int b)
{
    return a + b;
}

main ()
{
  int r, s, x;
  ...
  x = soma (r, s);
}
```

valor

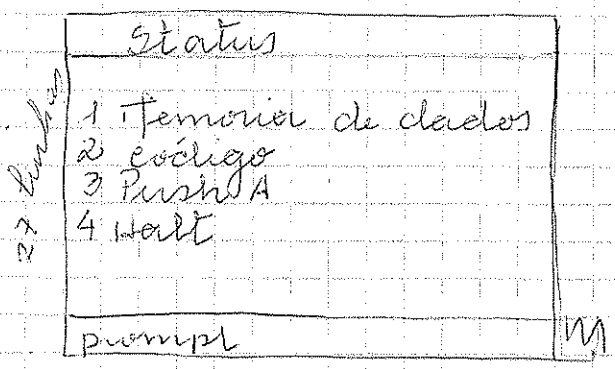
```
void soma 2 (int c)
{
  c = c + 2
}

main ()
{
  int y = 0
  soma 2 (y);
  printf ("%d", y);
}
```

```
void soma 2 (int * c)
{
  * c = * c + 2;
}

main ()
{
  int y = 0;
  soma 2 (& y);
  printf ("%d", y);
}
```

A[ ]

&(A[i])

80 colunas




```
#define Max_linha 80
typedef char linha [Max_linha + 1];

typedef linha Programa [Max_Prog];

int main()
{
Programa p1 = { "memória de dados",
                "código",
                "Push 4"
                "Halt" };

int n_linhas = 4;
...
lista_Prog (p1);

void lista_Prog (Programa p, int n_linhas)
{
  int i;
  for (i = 0; i < n_linhas; i++)
      printf ("%d: %s\n", i+1, p[i]);

                    type Nome = (string, string)
}
```

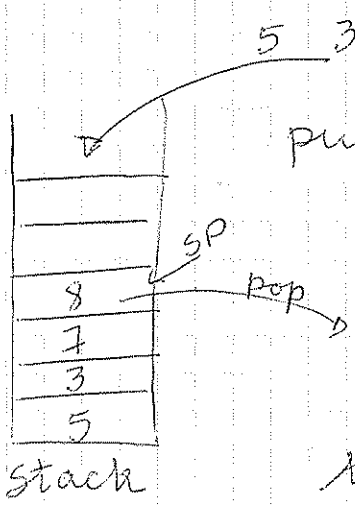
```
typedef struct sPonto
{
 int x;
 int y;
} Ponto;
```

```
main()
{ Ponto p1, p2;
  p2.x = p1.x + 5;
```

```
main()
{ struct sPonto
   { int x;
     int y;
   } P1, P2;
  P1.x = 2;
  P1.y = 7;
}
```

```
# define MaxLinha 80
typedef char Linha [MaxLinha+1];
typedef Linha Linhas [MaxProg];
typedef struct sPrograma
   {
     Linhas l;
     int nLinhas;
   } Programa;
```

```
main()
{
 Programa p1;
 ListaProg(p1.l, p1.nLinhas);
```



push (i)

Stack [SP+1]
SP ++

Pop

SP --
return (stack [SP+1]

top
return stack [SP]

empty ? SP

Henrique Silva Lopes N: 54066 07/04/08

MSP

— Operações sobre valores e endereços

(intervalo dos valores numéricos [-127, 255])

Push valor

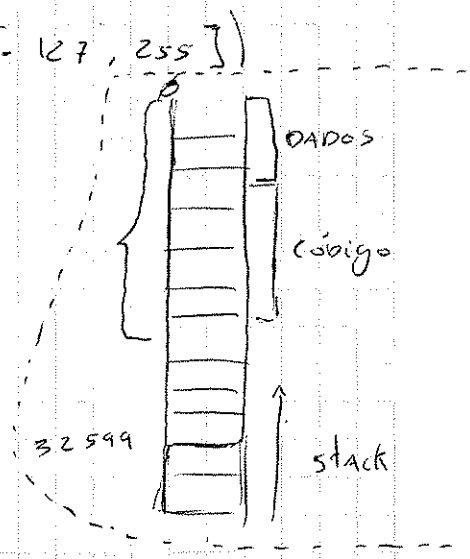
Psha endereço | id_variável

Load

Lda

Store

Stra

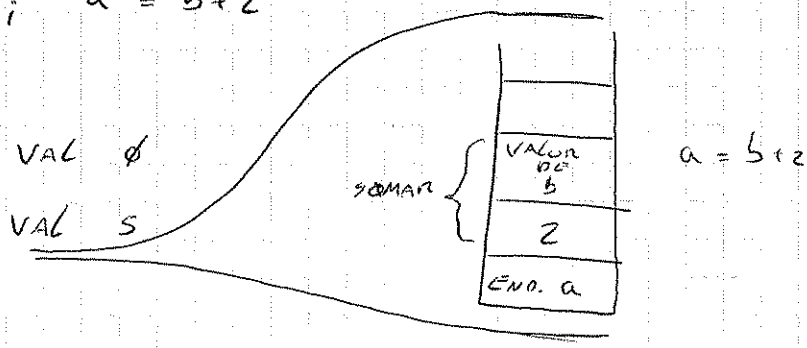


; Programa exemplo p/ atribuição
; a = 0 ; b = 5 ; a = b + 2

Memoria de dados

a 0 TAM 1 VAL 0

b 1 TAM 1 VAL 5



Psha a

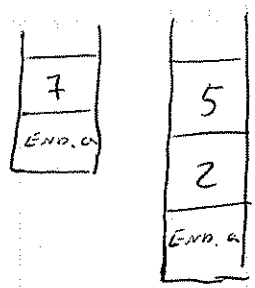
Push 2

Psha b > Coloca valor de b na stack

Load

ADD → 5 + 2 →

Store → a = 7 →



Psha a

Out

Halt

```
; Programa exemplo p/ Arrays
;
Memoria de dados
A   ø   TAM 10
Psha A
Push 7
ADDA
Push 5
Stone
Halt
```

```
Int A[10];
A[7] = 5;
```

AND	A AND B
OR	
Not	
EQ	
NE	
LT	<
LE	<=
GT	>
GE	>=

Instruções de Controlo

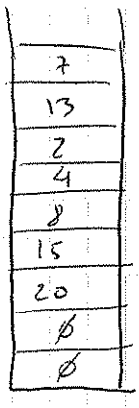
JMP label ou endereço

JMPF " "

CALL " "

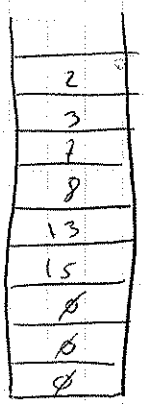
Ret

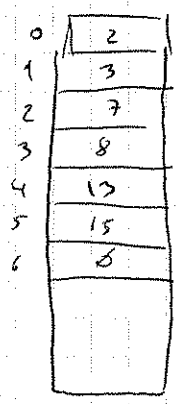
Pesquisa Linear (Lista não ordenada)

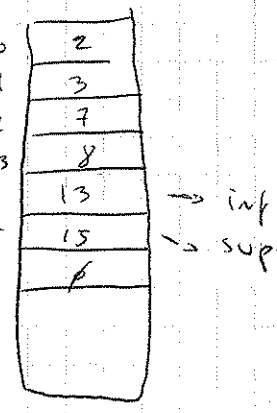



```
Pesquisa (Lista l, elem e)
    i ← ϕ
    Enq ( (l[i] != ϕ) && (l[i] != e) )
        i ← i + 1
    Retorna ( l[i] == ϕ ? -1 : i )
```

Pesquisa Dicotômica (lista l, elem e)







Carlos Ramos 54073 07-04-08

```
typedef struct sPonto
        { float x;
          float y;
        } ponto;

typedef struct sRectangulo
        { Ponto p1, p2;
        } Rectangulo;


float area (Rectangulo r)
 {
   return abs((r.p1.x - r.p2.x) * (r.p1.y - r.p2.y));
 }


 float perimetro (Rectangulo r)
 {
   return abs (2*((r.p1.x - r.p2.x) + (r.p1.y - r.p2.y)));
 }


  float abs (float x)
   {
     return (x<0)? (x*(-1)) : x;
   }
```

Jornadas de futebol

```
typedef int golos;
typedef char equipa[20];
typedef struct sJornada
        { equipa e;
          golos g;
        } jornada;
```

```
typedef struct sjogo
  { jornada j1, j2;
  } jogo

typedef jogo jornada[20];
```

```
bool igual (jornada j)
{ int i=0
  while strcmp((j[i].j1),
    (j[i].j2)) && (i<20)
    i++;
  return !(strcmp(j[i].j1.e,
    j[i].j2.e));
```

```
int main()
{ jornada j1, j2;
  jornada campeonato[26];
```

```
bool semagregat (jornada j)
 { int i=0, k=20;
equipa s1, s2;
   while (i<20 && k==20)
   { strcpy (s1, j[i].j1.e);
     strcpy (s2, j[i].j2.e);
     k=i+1;
while ((k<20) && strcmp (s1, j[k].j1.e) && strcmp (s1, j[k].j2.e)
&& strcmp (s2, j[k].j1.e) && strcmp (s2, j[k].j2.e))
   k++; i++;
return (i==2)? 1: 0;
```