Programas em C

Leitura de um inteiro com scanf:

```
#include <stdio.h>

int main()
{
        int n;

        /* A função scanf devolve um inteiro indicando quantos dos argumentos
        pretendidos conseguiu ler. Devolve um número negativo em situação de
        erro e EOF quando chega ao fim do buffer de input. */

        printf("\nIntroduza um inteiro: ");
        while(scanf("%d",&n)!=1) /* Conseguimos ler um inteiro? */
         {
                scanf("%*[^\n]"); /* Se não, limpámos o buffer! */
                printf("\n\nInteiro inválido!\n\nIntroduza-o de novo: ");
         }
        printf("\n\nFoi lido o inteiro: %d\n\n",n);
        return 0;
}
```

Leitura de um inteiro com getchar():

```
/* parseint.c

  Função que lê um inteiro do stdin seguindo
  a definição dada pela expressão regular:

  inteiro = (+|-)?[0-9]+
*/

#include <stdio.h>
#define INPUTERROR -1001

int limpa_buffer()
{
  char c;
  do
  {
        c=getchar();
  } while(c!='\n');
}
```

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

int parseint()
{
  char c;
  int valor=0, sinal=1;
  c = getchar();
  if(c=='+') c=getchar();
  if(c=='-')
  {
              sinal=-1;
              c=getchar();
  }
  while(isDigit(c))
  {
              valor = valor*10 + c-'0';
              c=getchar();
  }
  if(c=='\n') return (valor*sinal);
  else
  {
              limpa_buffer();
              return INPUTERROR;
  }
}

int main()
{
              int n;
              do
              {
                printf("\n\nIntroduza um valor inteiro: ");
                n = parseint();
              }
               while(n==INPUTERROR);
              printf("\n\nFoi lido o inteiro: %d\n\n",n);
              return 0;
}
```

Leitura de uma string com getchar():

```
/* ler_str.c

   Função que lê uma string do stdin terminada
   por '\n' ou por ter atingido o limite de caracteres:

*/

#include <stdio.h>

int ler_string(char *s, int nchars)
{
  int i=0;

  do
  {
            s[i] = getchar();
            i++;
  }
while((s[i-1]!='\n')&&(i<nchars));
  if(s[i-1]=='\n')
  {
            s[i-1] = '\0';
            return i-1;
  }
  else
  {
            s[i] = '\0';
            return i;
  }
}

int main()
{
            char frase[51];
            printf("\n\nIntroduza uma frase: ");
            while( !ler_string(frase,50) )
              printf("\n\nIntroduza uma frase: ");
            printf("\n\nFoi lida a frase: %s\n\n",frase);
            return 0;
}
```

Programa exemplo com várias funções sobre sequências de inteiros (2006 - Ficha4 - Ex1):

```
/* ----------------------------
  arrays.c - Processamento de Arrays
           2006-03-21: Created by jcr
  ---------------------------------- */

void menu()
{
  printf("\n\n    Sequencias de inteiros: lista de operacoes\n");
  printf("A - Ler a Sequencia\n");
  printf("B - Escrever a Sequencia\n");
  printf("C - Calcular o Maximo da Sequencia\n");
  printf("D - Calcular o Minimo da Sequencia\n");
  printf("E - Determinar a subsequencia de elementos acima da media\n");
  printf("F - Determinar a subsequencia de elementos abaixo da media\n");
  printf("G - Calcular o mmc da sequencia\n");
  printf("H - Ordenar a Sequencia com o BubbleSort\n");
  printf("I - Ordenar a Sequencia com o QuickSort\n");
  printf("J - Procurar um elemento\n");
  printf("S - Sair do Programa\n");
  printf("    Opcao: ");
}

/* ---------------------------
  quick sort
           2006-03-22: created by jcr
  --------------------------- */

void troca( int *a, int *b )
{ int t=*a; *a=*b; *b=t; }
```

```
void quickSort( int A[], int i, int j )
{
  int pivot, l, r;
  if(j>i+1)
  {
            pivot = A[i];
            l = i+1;
            r = j;
            while(l<r)
            {
              if(A[l]<=pivot) l++;
              else
              {
                        troca(&A[l],&A[r]);
                        r--;
              }
            }
            troca(&A[l],&A[i]);
            l--;
            escreverSeq(A,j+1);
            quickSort(A,i,l);
            quickSort(A,r,j);
  }
}

/* ----------------------------
   bubble sort
             2006-03-22: created by jcr
   ----------------------------- */

void bubbleSort( int A[], int nelems )
{
  int i=nelems-1, j, temp, troca=1;
  while((i>0)&&troca)
  {
            troca = 0; // Para já não houve nenhuma troca
            for(j=0;i>j;j++)
            {
              if(A[j]>A[j+1])
              {
                        troca = 1; // Houve uma troca nesta iteração
                        temp = A[j];
                        A[j] = A[j+1];
                        A[j+1] = temp;
              }
            }
            i--;
  }
}
```

```
/* ---------------------------
   Pesquisa Binária
            2006-03-30: created by jcr
   ---------------------------- */

int procura( int A[], int nelems, int elem )
{
  int i, inf=0, sup=nelems-1;
  i = (sup+inf)/2;
  while( (A[i]!= elem) && (inf <= sup))
  {
            if(A[i]<elem) inf = i+1;
            else sup = i-1;
            i = (sup+inf)/2;
  }
  if(A[i]==elem) return i;
  else return -1;
}


/* ------------------------------------------------
   Minimo Multiplo Comum entre dois numeros inteiros
   ------------------------------------------------
*/

int mmc( int a, int b )
{
  int aux=a;
  if(a<b) return mmc(b,a);
  else
            if(a%b==0) return a;
            else
            {
              do
              { aux+=a;
              } while((aux%a!=0)||(aux%b!=0));
              return aux;
            }
}
```

```
/* --- Aplicando o algoritmo anterior à Seq
                        Optei por fazê-lo recursivo para fins didácticos --- */

int mmcSeq( int A[], int nelems )
{
  if(nelems>1)
            return mmcSeqAux( A+2, nelems-2, mmc(A[0],A[1]) );
  else
            if(nelems) return A[0];
            else return 0;
}

int mmcSeqAux( int A[], int nelems, int mmcact )
{
  if(nelems==0) return mmcact;
  else return mmcSeqAux( A+1, nelems-1, mmc(mmcact,A[0]));
}


/* -------------------
   Máximo da Sequencia
   ------------------- */

int maxSeq( int A[], int nelems )
{
  int max = A[0], i;
  for(i=1;i<nelems;i++)
            if(A[i]>max) max=A[i];
  return max;
}


/* -------------------
   Mínimo da Sequencia
   ------------------- */

int minSeq( int A[], int nelems )
{
  int min = A[0], i;
  for(i=1;i<nelems;i++)
            if(A[i]<min) min=A[i];
  return min;
}
```

```
int lerSeq( int A[], int nelems )
{
  int i=0, cont=0, n;
  do
  {
            printf("\nIntroduza o elemento %d: ", i+1);
            cont += scanf("%d",&n);
            if(n) A[i]=n;
            i++;
  }
            while((i<nelems) && n);
  if(n)
            return cont;
  else
            return cont-1;
}

int escreverSeq( int A[], int nelems )
{
  int i;
  printf("\n\n");
  for(i=0;i<nelems;i++)
            printf("%d ",A[i]);
  printf("\n\n");
  return 0;
}

int somaSeq( int A[], int nelems )
{
  if(!nelems) return 0;
  else return A[0] + somaSeq(A+1,nelems-1);
}

int acimaMedia( int A[], int nelems , int subA[], float media)
{
  int i, j=0, cont=0;
  for(i=0;i<nelems;i++)
            if(A[i]>media)
            {  subA[j++]=A[i];
               cont++;
            }
  return cont;
}
```

```
int abaixoMedia( int A[], int nelems , int subA[], float media)
{
  int i, j=0, cont=0;
  for(i=0;i<nelems;i++)
            if(A[i]<media)
            {  subA[j++]=A[i];
               cont++;
            }
  return cont;
}

#define MAX 10 // numero maximo de elementos na sequencia

int main()
{
  char op;
  int sequencia[MAX] = {0}; // inicializacao a 0
  int subseq[MAX] = {0};    // vector para guardar as subseqs
  int lidos;   // numero de posicoes ocupadas na sequencia
  int nsubseq; // numero de elementos na subseq
  int soma;    // soma dos elementos lidos para a sequencia
  float media;
  int elem;    // elemento a procurar
  int res;     // resultado da procura
  menu();
  op = getchar();
  while(op!='S')
  {
            switch(op)
            {
              case 'A' : lidos = lerSeq(sequencia,MAX);
                                        soma = somaSeq(sequencia,lidos);
                                        media = soma / lidos;
                                        printf("\n\nForam lidos %d elementos.\n",lidos);
                                        printf("    Soma: %d\n",soma);
                                        printf("    Media: %f\n",media);
                                        break;
              case 'B' : escreverSeq(sequencia,lidos);
                                        break;
              case 'C' : printf("\n\nMax: %d\n\n", maxSeq(sequencia,lidos));
                                        break;
              case 'D' : printf("\n\nMin: %d\n\n", minSeq(sequencia,lidos));
                                        break;
              case 'E' : nsubseq = acimaMedia(sequencia, lidos, subseq, media);
                                        escreverSeq(subseq,nsubseq);
                                        break;
              case 'F' : nsubseq = abaixoMedia(sequencia, lidos, subseq, media);
                                        escreverSeq(subseq,nsubseq);
                                        break;
```

```
            case 'G' : escreverSeq(sequencia,lidos);
                                        printf("\n\nMMC: %d\n\n",
        mmcSeq(sequencia,lidos));
                                        break;
            case 'H' : bubbleSort(sequencia,lidos);
                                        escreverSeq(sequencia,lidos);
                                        break;
            case 'I' : quickSort(sequencia,0,lidos-1);
                                        escreverSeq(sequencia,lidos);
                                        break;
            case 'J' : printf("\nIntroduza o elemento a procurar: ");
                                        scanf("%d",&elem);
                                        res = procura(sequencia,lidos,elem);
                                        if(res>=0)
                                          printf("\nElemento encontrado na posição
%d\n",res);
                                        else
                                          printf("\n\nElemento Inexistente!\n\n");
                                        break;
          }
          getchar(); // limpar o \n do buffer de entrada
          menu();
          op = getchar();
 }
 return 0;
}
```

Ficha4 - 2008 - Avaliação de Alunos:

```
/* ----------------------------------------
  Ficha 4: Alunos
  ----------------------------------------
*/

#include <stdio.h>

#define MAXALUNO 100
#define MAXNUM   10
#define MAXNOME  60
#define MAXNOTA  10

typedef char Numero[MAXNUM];
typedef char Nome[MAXNOME];
typedef int Notas[MAXNOTA];
```

```
void listaAlunos(Numero X[], Nome Y[], Notas Z[])
{
  int i=0, j;

  printf("\n          LISTAGEM DOS ALUNOS\n");
  printf("-----------------------------------------\n");
  while((Y[i][0]!='\0') && (i<MAXALUNO))
    {
      printf("%10s:%30s:",X[i],Y[i]);
      for(j=0;j<MAXNOTA;j++)
        printf("%2d:",Z[i][j]);
      printf("\n");
      i++;
    }
  printf("\n-----------------------------------------");
}

void MediaDisc( Notas A[], Nome B[] )
{
  int i=0, j, Totais[MAXNOTA]={0};
  while((B[i][0]!='\0') && (i<MAXALUNO))
  {
    for(j=0;j<MAXNOTA;j++)
      Totais[j]+=A[i][j];
    i++;
  }
  printf("\n          MEDIA POR DISCIPLINA (%d alunos)\n",i);
  for(j=0;j<MAXNOTA;j++)
    printf("%d ",Totais[j]/i);
  printf("\n ----------------------------\n");
}

int getPos( char *nome, Nome X[] )
{
  int i=0;
  while((strcmp(nome,X[i])) && (X[i][0]!='\0'))
    i++;
  return (X[i][0]!='\0')?i:-1;
}
```

```
int main()
{
  Numero AlunoNum[MAXALUNO] = {"4140","4156","4238"};
  Nome AlunoNom[MAXALUNO] = {"Jose Alberto","Jose Carlos","Paulo Jorge","\0"};
  Notas AlunoNot[MAXALUNO] = {{12,15,0,0,12,0,0,18,17,0},
                              {13,14,11,0,0,11,10,0,0,0},
                              {15,10,0,0,0,13,12,14,0,0}};
  listaAlunos(AlunoNum,AlunoNom,AlunoNot);
  MediaDisc(AlunoNot, AlunoNom);
  printf("\n%d : %d : %d \n\n", getPos("Jose Carlos",AlunoNom), getPos("Paulo
Jorge",AlunoNom), getPos("Jose Alberto", AlunoNom));
}
```

Aplicação demo de alunos: constructores, visualizadores, persistência em ficheiros de texto:

- **Módulo Aluno: Interface: aluno.h, Implementação;**

Interface: aluno.h

```
/*
 *  aluno.h
 *
 *
 *  Created by JosŽ Carlos Ramalho on 08/11/03.
 *  Copyright 2008 __MyCompanyName__. All rights reserved.
 *
 */

#ifndef _Aluno
#define _Aluno

typedef struct sAluno
 {
   char *num;
            char *nome;
            char *curso;
 } Aluno;

typedef struct sTurma
 {
   Aluno a;
            struct sTurma *seg;
 } TurmaNodo, *Turma;

  Aluno consAluno( char *nu, char *no, char *cu );
  Turma consTurma( Turma t, Aluno a );
```

```
  void showAluno( Aluno a );
  void showTurma( Turma t );
  void saveAluno( Aluno a, FILE *f );
  void saveTurma( Turma t, FILE *f );

  Aluno readAluno( char *s );
  Turma readTurma( FILE *f );
#endif
```

Implementação:

```
/*
 *  aluno.c
 *
 *
 *  Created by JosŽ Carlos Ramalho on 08/11/03.
 *  Copyright 2008 __MyCompanyName__. All rights reserved.
 *
 */

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "aluno.h"

Aluno consAluno( char *nu, char *no, char *cu )
 {
   Aluno aux;
            aux.num = strdup(nu);
            aux.nome = strdup(no);
            aux.curso = strdup(cu);
            return aux;
 }

Turma consTurma( Turma t, Aluno al )
 {
   Turma aux;
            aux = (Turma)malloc(sizeof(TurmaNodo));
            aux->a = al;
            aux->seg = t;
            return aux;
 }

void showAluno( Aluno a )
{
  printf("%s:%s:%s\n", a.num, a.nome, a.curso );
}
```

```
void showTurma( Turma t )
{
  if(t)
    {
              showAluno(t->a);
              showTurma(t->seg);
            }
}

void saveAluno( Aluno a, FILE *f )
{
  fprintf(f,"%s:%s:%s\n",a.num,a.nome,a.curso);
}

void saveTurma( Turma t, FILE *f )
{
  if(t)
    {
              saveAluno(t->a,f);
              saveTurma(t->seg,f);
            }
}

Aluno readAluno( char *s )
{
  Aluno aux;
  char campo[100];
  int i=0;
  while(*s != ':')
    {
              campo[i] = *s;
              i++;
              s++;
            }
  campo[i] = '\0';
  aux.num = strdup(campo);
  i = 0; s++;
  while(*s != ':')
    {
              campo[i] = *s;
              i++;
              s++;
            }
  campo[i] = '\0';
  aux.nome = strdup(campo);
  i = 0; s++;
```

```
  while(*s)
   {
              campo[i] = *s;
              i++;
              s++;
            }
  campo[i] = '\0';
  aux.curso = strdup(campo);
  return aux;
}

Turma readTurma( FILE *f )
{
  Turma aux = NULL;
  char buffer[500];
  while(fgets(buffer,500,f))
              aux = consTurma( aux, readAluno(buffer) );
  return aux;
}
```

- **Aplicação de teste de escrita em ficheiro: teste.c:**

```
/*
 *  teste.c
 *
 *
 *  Created by JosŽ Carlos Ramalho on 08/11/03.
 *  Copyright 2008 __MyCompanyName__. All rights reserved.
 *
 */

#include <stdio.h>
#include "aluno.h"

int main()
{
  FILE *fp;
  Aluno a1 = {"4140","Jose Carlos Ramalho","LESI"};
  Aluno a2 = {"4238","Jose Alberto Rodrigues","LESI"};
  Aluno a3 = {"4156","Paulo Jorge Domingues","LESI"};
  showAluno(a2);
  Turma t1;
   t1 = consTurma(NULL,a1);
  t1 = consTurma(t1,a2);
  t1 = consTurma(t1,a3);
```

```
showTurma(t1);
  fp = fopen("ALUNO.TXT","w");
  saveTurma(t1,fp);
  fclose(fp);
}
```

- **Aplicação de teste de leitura em ficheiro: teste2.c:**

```
/*
 *  teste2.c
 *
 *
 *  Created by JosŽ Carlos Ramalho on 08/11/03.
 *  Copyright 2008 __MyCompanyName__. All rights reserved.
 *
 */

#include <stdio.h>
#include "aluno.h"

int main()
{
  FILE *fp;
  Turma t1;
  fp = fopen("ALUNO.TXT","r");
  t1 = readTurma(fp);
  fclose(fp);
  showTurma(t1);
}
```

http://www3.di.uminho.pt/~jcr/AULAS/didac/programasC/tp2-2006/