| INF 1620 – P3 - 21/06/02 | Questão 1 |
|---|---|
| Nome: | |
| Matrícula: | Turma |

Considere um arquivo texto que descreve um conjunto de retângulos e triângulos. Cada linha do arquivo contém a descrição de uma figura. O primeiro caractere da linha indica seu tipo: `r` para retângulo e `t` para triângulo. Esse caractere é seguido pelos valores, números reais, da base e da altura das figuras. Considere ainda que não existe linhas em branco no arquivo. Um exemplo desse formato é mostrado abaixo.

```
r  2.0  4.3
t  4.0  5.0
r  1.0  1.5
r  6.0  0.5
t  1.0  1.02
```

Escreva um programa completo que leia o arquivo "entrada.txt", que contém as descrições das figuras no formato descrito acima, e imprima na tela o valor da maior área das figuras listadas no arquivo. Se não for possível abrir o arquivo, o programa deve ter como saída a mensagem "ERRO". Se não existir nenhuma figura no arquivo (arquivo existente, mas vazio), deve-se imprimir o valor zero.

```
#include <stdio.h>

int main (void)
{
   char c;
   float b, h;
   float a, amax = 0.0;
   FILE* fp;

   fp = fopen("entrada.txt","rt");
   if (fp == NULL) {
      printf("ERRO\n");
      return 1;
   }

   while (fscanf(fp," %c %f %f",&c,&b,&h) == 3) {
      if (c == 't')
         a = (b*h)/2;
      else
         a = b*h;

      if (a > amax)
         amax = a;
   }
   fclose(fp);

   printf("Area maxima: %f\n", amax);
   return 0;
}
```

---

*Outra opção para captura valores (leitura linha a linha):*

```
...
char linha[121];
...
while (fgets(linha,121,fp) != NULL) {
   sscanf(linha,"%c %f %f",&c,&b,&h);
   ...
```

*Não separe as folhas deste caderno. Todas as folhas devem ter seu nome. Responda cada questão na folha correspondente. Use o verso se necessário.*

| INF 1620 – P3 - 21/06/02 | Questão 2 |
|---|---|
| Nome: | |
| Matrícula: | Turma |

Considere o tipo que representa um produto de um estoque dado por:

```
struct produto {
   int id;          /* código de identificação */
   char nome[81];   /* nome do produto */
   int quant;       /* quantidade disponível no estoque */
   float preco;     /* preço de venda */
};
typedef struct produto Produto;
```

Utilizando a função `qsort` da biblioteca padrão, escreva uma função para ordenar, em ordem crescente do código de identificação, os produtos de um estoque. A função recebe como parâmetros o número de produtos e um vetor que armazena ponteiros para os produtos presentes no estoque. O protótipo da função deve ser:

```
void ordena (int n, Produto** vet);
```

Obs: a função `ordena` consistirá apenas da chamada da função `qsort`, fazendo uso de uma função auxiliar para comparar os elementos do vetor, que deve ser escrita.

```
/* função auxiliar para comparação */
int compara (const void* v1, const void* v2)
{
   /* converte para ponteiro do elemento; no caso,
      ponteiro para ponteiro de produto
   */
   Produto** pp1 = (Produto**) v1;
   Produto** pp2 = (Produto**) v2;

   /* converte para ponteiro de produto */
   Produto *p1 = *pp1;
   Produto *p2 = *pp2;

   /* faz a comparação */
   if (p1->id < p2->id)
      return -1;
   else if (p1->id > p2->id)
      return 1;
   else
      return 0;
}

/* função para ordenação */
void ordena (int n, Produto** vet)
{
   qsort(vet, n, sizeof(Produto*), compara);
}
```

---

Protótipos de funções que podem ser úteis:

```
void qsort (void* vet, int n, int tam, int (*comp) (const void*, const void*));
```

*Não separe as folhas deste caderno. Todas as folhas devem ter seu nome. Responda cada questão na folha correspondente. Use o verso se necessário.*

| INF 1620 – P3 - 21/06/02 | Questão 3 |
|---|---|
| Nome: | |
| Matrícula: | Turma |

Considere, novamente, o tipo que representa um produto de um estoque dado por:

```
struct produto {
   int id;          /* código de identificação */
   char nome[81];   /* nome do produto */
   int quant;       /* quantidade disponível no estoque */
   float preco;     /* preço de venda */
};
typedef struct produto Produto;
```

Considere um vetor que armazena ponteiros para os produtos presentes em um estoque ordenado (ordem crescente) pelo código de identificação de cada produto. Sem usar funções da biblioteca padrão, escreva uma função que faça uma *busca binária* no vetor. Essa função recebe como parâmetros de entrada o número de produtos, o vetor e o código do produto que se deseja buscar. A função deve ter como valor de retorno o ponteiro para o produto cujo código é igual ao código fornecido. Se não existir um produto com o código fornecido, a função deve ter `NULL` como valor de retorno. O protótipo da função deve ser dado por:

```
Produto* busca (int n, Produto** vet, int id);
```

```
Produto* busca (int n, Produto** vet, int id)
{
   int i = 0;          /* índice inicial */
   int f = n-1;        /* índice final */
   int m;              /* índice médio */

   while (i <= f) {
      m = (i + f) / 2;
      if (id < vet[m]->id)
         f = m - 1;           /* ajusta fim */
      else if (id > vet[m]->id)
         i = m + 1;           /* ajusta início */
      else
         return vet[m];       /* achou o produto */
   }
   return NULL;               /* não achou o produto */
}
```

---

*Outra opção: implementação recursiva.*

```
Produto* busca (int n, Produto** vet, int id)
{
    if (n > 0) {
       int m = (n - 1) / 2;
       if (id < vet[m]->id)
          return busca(m,vet,id);
       else if (id > vet[m]->id)
          return busca(n-1-m, &vet[m+1],id);
       else
          return vet[m];             /* achou o produto */
    }
    return NULL;                     /* não achou o produto */
 }
```

*Não separe as folhas deste caderno. Todas as folhas devem ter seu nome. Responda cada questão na folha correspondente. Use o verso se necessário.*

| INF 1620 – P3 - 21/06/02 | Questão 4 |
|---|---|
| Nome: | |
| Matrícula: | Turma |

Considere uma árvore binária de busca que armazena nomes (cadeias de caracteres). O tipo que representa um nó da arvore é dado por:

```
struct arv {
   char nome[81];
   struct arv* esq;
   struct arv* dir;
};
typedef struct arv Arv;
```

Sabendo-se que numa árvore binária de busca as informações associadas aos nós da sub-árvore à esquerda são menores (no caso, ordem alfabética) que a informação associada à raiz e que as informações dos nós da sub-árvore à direita são maiores, escreva uma função que insira um novo nome na árvore de busca. Se o nome já existir na estrutura, a árvore não deve ser alterada. A função recebe como parâmetros a raiz da árvore e a cadeia de caracteres a ser inserida, e deve ter como retorno o valor atualizado da raiz. O protótipo dessa função deve ser:

```
Arv* insere (Arv* a, char* s);
```

Obs: Para simplificar, considere que os nomes são compostos apenas por letras minúsculas.

```
Arv* insere (Arv* a, char* s)
{
   if (a == NULL) {
      /* árvore vazia: cria elemento como sendo a raiz */
      a = (Arv*) malloc(sizeof(Arv));
      strcpy(a->nome, s);
      a->esq = a->dir = NULL;
   }
   else if (strcmp(s,a->nome) < 0)
      a->esq = insere(a->esq,s);
   else if (strcmp(s,a->nome) > 0)
      a->dir = insere(a->dir,s);

   return a;
}
```

---

Protótipos de funções que podem ser úteis:

```
int strlen (char* s);
int strcmp (char* s, char *t);
char* strcpy (char* destino, char* fonte);
char* strcat (char* destino, char* fonte);
```

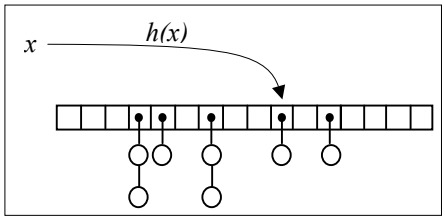
*Não separe as folhas deste caderno. Todas as folhas devem ter seu nome. Responda cada questão na folha correspondente. Use o verso se necessário.*

| INF 1620 – P3 - 21/06/02 | Questão 5 |
|---|---|
| Nome: | |
| Matrícula: | Turma |

Considere um cadastro de alunos armazenado numa tabela de dispersão (*hash*) que usa como chave de busca o número de matrícula. Considere também a existência de uma função de *hash* que, dado o número de matrícula, retorna o índice na tabela. O protótipo dessa função é dado por:

```
int hash (int mat);
```

O tratamento de colisão é feito utilizando-se listas encadeadas para armazenar os elementos que colidem. Dessa forma, cada posição na tabela representa um ponteiro para o primeiro nó de uma lista encadeada, conforme ilustra a figura ao lado.




Considere as seguintes declarações, relativas à tabela de *hash* dos alunos.

```
struct aluno {
   int matric;               /* número de matrícula */
   char nome[81];            /* nome do aluno */
   struct aluno* prox;       /* próximo elemento da lista*/
};
typedef struct aluno Aluno;

#define N 127
typedef Aluno* Hash[N];
```

Considerando que posições não preenchidas da tabela de *hash* armazenam o valor `NULL` (lista vazia), escreva uma função de busca que verifica se um aluno, dado seu número de matrícula, está armazenado no cadastro. A função deve retornar o ponteiro da estrutura `Aluno` se estiver no cadastro ou `NULL` se não estiver. O protótipo da função deve ser:

```
Aluno* busca (Hash tab, int mat);
```

```
Aluno* busca (Hash tab, int mat)
{
   int h = hash(mat);
   Aluno* a;
   for (a = tab[h]; a != NULL; a = a->prox) {
      if (a->matric == mat)
         return a;
   }
   return NULL;
}
```

*Não separe as folhas deste caderno. Todas as folhas devem ter seu nome. Responda cada questão na folha correspondente. Use o verso se necessário.*

RASCUNHO

*Respostas nesta folha não serão consideradas.*

*Não separe as folhas deste caderno.*