MAC2166 Introdução à Computação para Engenharia

Escola Politécnica

Segunda Prova — 19 de maio de 2008

Nome: _____

Assinatura: _____

Nº USP: _____ Turma: _____

Professor: _____

Instruções:

1. Não destaque as folhas deste caderno.
2. A prova consta de 4 questões. Verifique antes de começar a prova se o seu caderno de questões está completo.
3. A prova pode ser feita a lápis. Cuidado com a legibilidade.
4. Qualquer questão pode ser resolvida em qualquer página. Se a questão não estiver na página correspondente ao enunciado basta indicar isto na página e escrever QUESTÃO i em letras ENORMES antes da solução.
5. Não é necessário apagar rascunhos no caderno de questões.
6. Não é permitido o uso de folhas avulsas para rascunho.
7. Não é permitido o uso de calculadoras.
8. Não é permitido a consulta a livros, apontamentos ou colegas.

DURAÇÃO DA PROVA: 2 horas

Questão	Valor	Nota
1	2,5	
2	2,5	
3	2,5	
4	2,5	
Total	10,0	

Questão 1 (valor: 2,5 pontos)

Simule a execução do programa abaixo, destacando a saída do programa. A saída do programa é tudo que resulta dos comandos printf. Para efeito de correção só será considerada a saída do programa.

```
# include <stdio.h>

int f1 (int a, int b) {
  int z;
  a = b - a;
  b = a + b;
  z = a + b;
  return z;
}
int f2 (int *a, int b) {
  int z;
  *a = b - *a;
   b = *a + b;
   z = *a + b;
  return z;
}
int f3 (int *a, int b) {
  b = *a + b;
  *a = b + 2;
  return b;
}

int main () {
  int nusp;
  int a, b, c, d, e;
  float f;
  printf ("Entre com seu no. USP: ");
  scanf ("%d", &nusp); /* use aqui seu numero USP */
  printf ("nusp = %d\n", nusp);
  a = nusp % 10;
  b = a + 2;
  printf ("1: a=%d b=%d\n", a, b);
  c = a;
  d = b;
  e = f1 (c, d);
  printf ("2: a=%d b=%d c=%d d=%d e=%d\n", a, b, c, d, e);
  d = b;
  e = f2 ( &a, d);
  printf ("3: a=%d b=%d c=%d d=%d e=%d\n", a, b, c, d, e);
  d = b;
  a = f3 ( &d, d);
  printf ("4: a=%d b=%d d=%d\n", a, b, d);
  d = 2 * b + 1;
  b = 2;
  f = d / b;
  printf ("5: a=%d b=%d d=%d e=%d, f=%f\n", a, b, d, e, f);
  f = a;
  f = (2 * f + 1) / 2;
  e = f;
  printf ("6: a=%d b=%d c=%d e=%d f=%f\n", a, b, c, e, f);
  return 0;
}
```

Para efeito de correção só será considerada a saída do programa. Você pode usar a tabela abaixo como bem entender. Cada turma está habituada a simular de maneira diferente, fazendo tabelas com "caras" diferentes.

saída

Questão 2 (valor: 2,5 pontos)

Abaixo seguem algumas possíveis implementações para uma função que lê uma sequência de caracteres e decide se seus parênteses estão *quase balanceados*. Isto significa que cada fecha-parênteses corresponde a um abre-parênteses (pode haver abre-parênteses que não fechou). O protótipo da função é

```
int quase(char fim);
```

ele devolve 1 se os parênteses estiverem quase balanceados até a primeira ocorrência de `fim`, 0 caso contrário. Para evitar confusões, se o caractere `fim` for um fecha-parênteses, a função pode devolver qualquer coisa.

Por exemplo, se o texto a ser lido for

$$a(a(b)(a+b(cd))*)/c(ab))*(?)$$

então os valores retornados por `quase('*')`, `quase('/')` e `quase('?')` são respectivamente 1, 1 e 0. Para cada função, indique se está certa ou errada; se estiver errada, dê um texto e uma chamada da função que devolve o valor errado.

(a) Correto ( ) Incorreto ( )

```
int quase(char fim) {
  int conta = 0, bal = 1; char c;
  scanf ( "%c", &c );
  while ( c != fim ) {
    if ( c == '(' ) conta++;
    else if ( c == ')' ) conta--;
    if ( conta < 0 ) bal = 0;
  }
  return bal;
}
```

(b) Correto ( ) Incorreto ( )

```
int quase(char fim) {
  int conta = 0, bal = 1; char c;
  scanf ( "%c", &c );
  while ( c != fim ) {
    if ( conta < 0 ) bal = 0;
    if ( c == '(' ) conta++;
    else if ( c == ')' ) conta--;
    scanf ( "%c", &c );
  }
  return bal;
}
```

(c) Correto ( ) Incorreto ( )

```
int quase(char fim) {
  int conta = 0; char c;
  scanf ( "%c", &c );
  while ( c != fim ) {
    if ( c == '(' ) conta++;
    else if ( c == ')' ) conta--;
    else if ( conta < 0 ) return 0;
    scanf ( "%c", &c );
  }
  return 1;
}
```

(d) Correto ( ) Incorreto ( )

```
int quase(char fim) {
  int conta = 0, bal = 1; char c;
  for (scanf ( "%c", &c );  c != fim; scanf ( "%c", &c ) ) {
    conta++;
    if ( c == '(' ) conta++;
    else if ( c == ')' ) conta -= 2;
    else conta --;
    if ( conta > 0 ) bal = 0;
  }
  return bal;
}
```

(e) Correto ( ) Incorreto ( )

```
int quase(char fim) {
  int conta = 0, bal = 1; char c;
  for (scanf ( "%c", &c );  c != fim; scanf ( "%c", &c ) ) {
    if ( c == '(' ) conta++;
    else if ( c == ')' ) conta--;
  }
  if ( conta < 0 ) return 0
  else return 1;
}
```

Questão 3 (valor: 2,5 pontos)

No curso de cálculo numérico você irá aprender que deve evitar somar dois números reais de valores quase opostos no computador. Caso isso ocorra, você pode perder rapidamente precisão em suas contas. Esse problema é conhecido como *erro de cancelamento*.

O cálculo das raízes de uma equação do segundo grau, $ax^2 + bx + c = 0$, pela forma usual pode esbarrar em um erro de cancelamento sempre que a raiz do discriminante e b tiverem valores absolutos parecidos. Mas é possível evitar esse problema usando a seguinte fórmula alternativa:

1. Calcule
$$pre_r = \frac{-b - \text{sinal}(b)\sqrt{b^2 - 4ac}}{2},$$
em que $\text{sinal}(b) = 1$ se $b \geq 0$ ou -1 caso contrário.

2. Calcule a primeira raiz por $r_1 = pre_r/a$, calcule a outra raiz por $r_2 = c/pre_r$.

Escreva um programa em C que lê $a \neq 0$, b e c e imprime as raízes da equação do segundo grau obtidas pelas operações descritas acima. Caso o discriminante seja negativo, seu programa deve avisar que não há raízes.

Para fazer isso, você deve implementar a função `sinal` com protótipo dado por

```
int sinal(float x);
```

Além disso, você pode chamar a função `raiz_quadrada`, com o seguinte protótipo:

```
float raiz_quadrada(float x);
```

Você não precisa implementar essa função, basta usá-la.

Questão 4 (valor: 2,5 pontos)

Quando a Física vira Matemática Pura, vem como subproduto o jogo do bilhar ideal. Nele a mesa é um quadrado de lado 1 com uma caçapa em cada canto, a bola é um ponto (acabe justinho na caçapa), e os choques são totalmente elásticos. Isto significa que, ao tocar numa parede, a componente da velocidade na direção da parede não se altera, e a componente normal troca de sinal mantendo o valor absoluto.

Escreva um programa que leia as coordenadas de uma bola, as componentes de sua velocidade e um inteiro positivo n e decide se a bola entra em uma caçapa após bater em *no máximo* n lados (não mais que n *tabelas*). O programa deve imprimir uma mensagem

`Caçapa em` n `tabelas` (onde n é o número certo)

ou

`Não entrou!` (se for esse o caso)

Seu programa deve definir e usar as funções

- `void bate( float *x, float *y, float vx, float vy)`
  em que, dadas a posição inicial da bola em `(*x, *y)`, e sua velocidade em `(vx, vy)`, devolve o ponto em que a bola bate na parede em `(*x, *y)`.

- `void nova_velocidade( float x, float y, float *vx, float *vy)`
  em que, dadas a posição de batida da bola em `(x, y)`, e sua velocidade em `(*vx, *vy)`, devolve a velocidade refletida em `(*vx, *vy)`.