Próbny pisemny egzamin dyplomowy

na Uniwersytecie Wrocławskim

na kierunku matematyka

część II

specjalność matematyka z informatyką

Kwiecień 2002

1. Wykazać, że jeżeli
$$f(x) = e^x - 2, \qquad x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}, \quad n > 1,$$
to dla każdego $x_1 \in R$ – punktu startowego, zachodzi:
$$\lim_{n\to\infty} x_n = \ln 2.$$
Wsk.: Dla $e_n = x_n - \ln 2$ istnieje $\xi_n \in R$, takie że
$$e_{n+1} = \frac{f''(\xi_n)}{2f'(x_n)} e_n^2.$$

---

2. Dla $x = (x_1, x_2) \in R^2$ definujemy *normę wektorową* $\|x\|_\infty \stackrel{\text{def}}{=} \max(|x_1|, |x_2|)$ oraz odpowiednio *normę macierzową* $\|A\|_\infty \stackrel{\text{def}}{=} \max\limits_{i=1,2} \sum\limits_{j=1}^{2} |a_{ij}|$. Wektor $\tilde{b} \in R^2$ przybliża $b \in R$ z *błędem względnym* $\frac{\|\tilde{b}-b\|_\infty}{\|b\|_\infty}$. Niech będzie dana macierz
$$A = \begin{bmatrix} 1 & 1+\varepsilon \\ 1-\varepsilon & 1 \end{bmatrix}, \quad \text{gdzie} \quad 0 < \varepsilon.$$
Dla jakiego $0 < \varepsilon$ rozwiązanie przybliżone równania $A\tilde{x} = \tilde{b}$ przybliża rozwiązanie dokładne równania $Ax = b$ z *błędem względnym* spełniającym:
$$\frac{\|\tilde{x} - x\|_\infty}{\|x\|_\infty} \le \frac{\|\tilde{b} - b\|_\infty}{\|b\|_\infty}.$$

---

3. Niech będą dane następujące deklaracje w pseudo **Pascalu**:

```
type
  ref   = ^slowo;
  slowo = record
            nast,poprz : ref;
                     n : integer;
          end;

procedure Add(var w:ref);
var
  r : ref;
begin
```

```
  if w<>nil then
    begin
      r        := new(ref);
      r^.nast  := nil;
      r^.poprz := w;
      r^.n     := w^.n+1;
      w^.nast  := r;
      w        := r
    end else begin
               w        := new(ref);
               w^.nast  := nil;
               w^.poprz := nil;
               w^.n     := 1
             end
end;
```

co odpowiada następującym deklaracjom w **C**:

```
struct StSlowo
{
   StSlowo *nast,*poprz;
   int n;
};


void Add(StSlowo **w)
{
   StSlowo *r;

   if(*w!=NULL)
   {
      r          = new StSlowo;
      r->nast    = NULL;
      r->poprz   = *w;
      r->n       = (*w)->n+1;
      (*w)->nast = r;
      *w = r;
   }
   else
   {
      *w          = new StSlowo;
      (*w)->nast  = NULL;
      (*w)->poprz = NULL;
```

```
      (*w)->n     = 1;
   }
}
```

Rozważmy następujący fragment programu w **Pascalu**

```
var
  i,n,S  : integer;
  w      : ref;
begin
  n := 3;
  S := 0;
  w := nil;
  for i:=0 to n do
    Add(w);
  while w^.poprz<>nil do
    begin
      S := S+w^.n;
      w := w^.poprz
    end;
  while w^.nast<>nil do
    begin
      S := S+w^.n;
      w := w^.nast
    end;
{S := ? }
end;
```

i odpowiednio w **C**:

```
int i,n,S;
StSlowo *w;

n = 3;
S = 0;
w = NULL;
for(i=0;i<=n;++i)
  Add(&w);
while(w->poprz!=NULL)
{
   S += w->n;
   w = w->poprz;
```

```
}
while(w->nast!=NULL)
{
   S += w->n;
   w = w->nast;
}
/* S := ? */
```

Pytanie: Po wykonaniu programu, ile wynosi S? Odpowiedz uzasadnić opisując jak zmienia się S.

---

4. Rozważmy następujące deklaracje

```
 struct TRec
 {
    char Na[35],Im[25];
    double Oc;
 };

 const struct TRec RecDef[]=
   {{//Kowalski//, //Jan//,    3.5 },{//Nowak//,  //Ewa//,  4.75},
    {//Alfawicka//,//Barbara//,4.5 },{//Adamski//,//Piotr//,3.1},
    {//Zero//,     //Siedem//, 3.25},{//Nowak//,  //Adam//, 4.25}};


struct TRec Rec[20],R,*wRec;
int i,j,n;

n = sizeof(RecDef)//sizeof(RecDef[0]);

for(i=0;i<n;++i)
{
   wRec = Rec+i;
   strcpy(wRec->Na,RecDef[i].Na);
   strcpy(wRec->Im,RecDef[i].Im);
   wRec->Oc = RecDef[i].Oc;
}
```

Czy instrukcja:

```
for(i=1;i<n;++i)
  for(j=0;j<n-i;++j)
    if(0<strcmp(Rec[j].Na,Rec[j+1].Na))
    {
       memmove(&R,Rec+(j+1),sizeof(struct TRec));
       memmove(Rec+(j+1),Rec+j,sizeof(struct TRec));
       memmove(Rec+j,&R,sizeof(struct TRec));
    }
```

sortuje tablicę `Rec` ? Podać ile jest wykonywanych porównań i wymian przy realizacji tego fragmentu kodu.

---

5. Znowu mamy deklarację tym razem funkcji rekurencyjnej

```
int N=0;

int F(int n)
{
   N++;
   if(0<n)
     return (n%2 ? F(n+1) : F(n-3) );
   return N;
}
```

I po wykonaniu

```
int i,w;

N = 0;
w = F(i);
```

dla `i = 5;` oraz `i = 6;` odpowiednio, ile wynosi `w`. Napisać jawnym wzorem `F(i)`. Dla jakich wartości i (dziedziny $i \in N$) działanie funkcji zakończy się (wyznaczyć zbieżnik)?

6. Teraz rozważmy klasyczną deklarację *klasy* opisującej liczby zespolone

```
class TZesp
{
```

```
private:
        double r,i;
public:
        TZesp();
        TZesp(double Ar, double Ai);
        double Re();
        double Im();
        TZesp operator=(TZesp a);
        friend TZesp operator*(TZesp a,TZesp b);
};

TZesp::TZesp()
{
   r = i = 0.0;
}

TZesp::TZesp(double Ar, double Ai)
{
   r = Ar;
   i = Ai;
}

double TZesp::Re()
{
   return r;
}

double TZesp::Im()
{
   return i;
}

TZesp TZesp::operator=(TZesp a)
{
   r = a.r;
   i = a.i;
   return *this;
}

TZesp operator*(TZesp a,TZesp b)
{
   return TZesp(a.r*b.r-a.i*b.i,a.r*b.i+a.i*b.r);
}
```

Pytanie: Po wykonaniu

```
TZesp z1(0,1),*z2,z3;

z2 = new TZesp(1,2);
z3 = z1* *z2;
```

Ile wynosi `z3`? Na przykładzie przeciążenia `operatora *` przeciążyć `operator /` w ten sposób, aby opisywał dzielenie liczb zespolonych. W przypadku dzielenia przez zero zdefiniować odpowiednią *klasę*, która będzie podstawą do obsłużenia tej sytuacji krytycznej przez generowanie odpowiedniego *wyjątku*.
