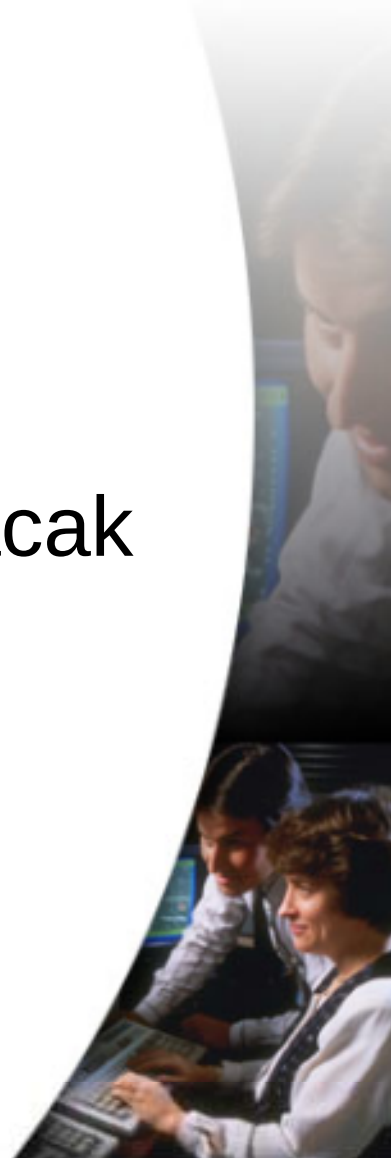
Monte Carlo Sebagai Metode Pencarian Acak

Achmad Basuki

Politeknik Elektronika Negeri Surabaya PENS-ITS, Surabaya 2004

Teknik Pencarian Acak

- Teknik pencarian solusi dengan membangkitkan atau mendapatkan solusi secara acak yang dilakukan berkali-kali hingga akhir ditemukan solusi yang diinginkan.
- Baik tidaknya hasil dari pencarian acak tergantung pada nilai acuan yang diberikan apakah yang dicari nilai tertentu, nilai maksimal dan nilai minimal.



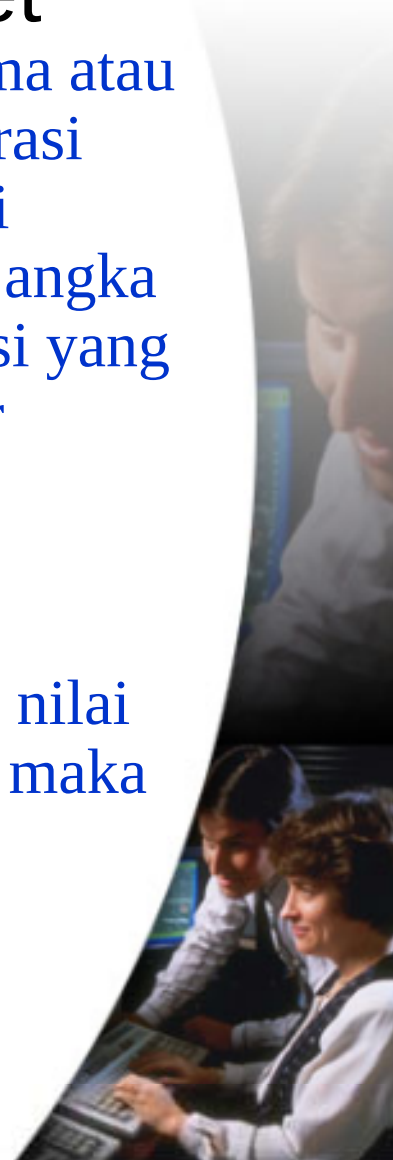
Model Pencarian Acak

- Pencarian Acak Dengan Nilai Target

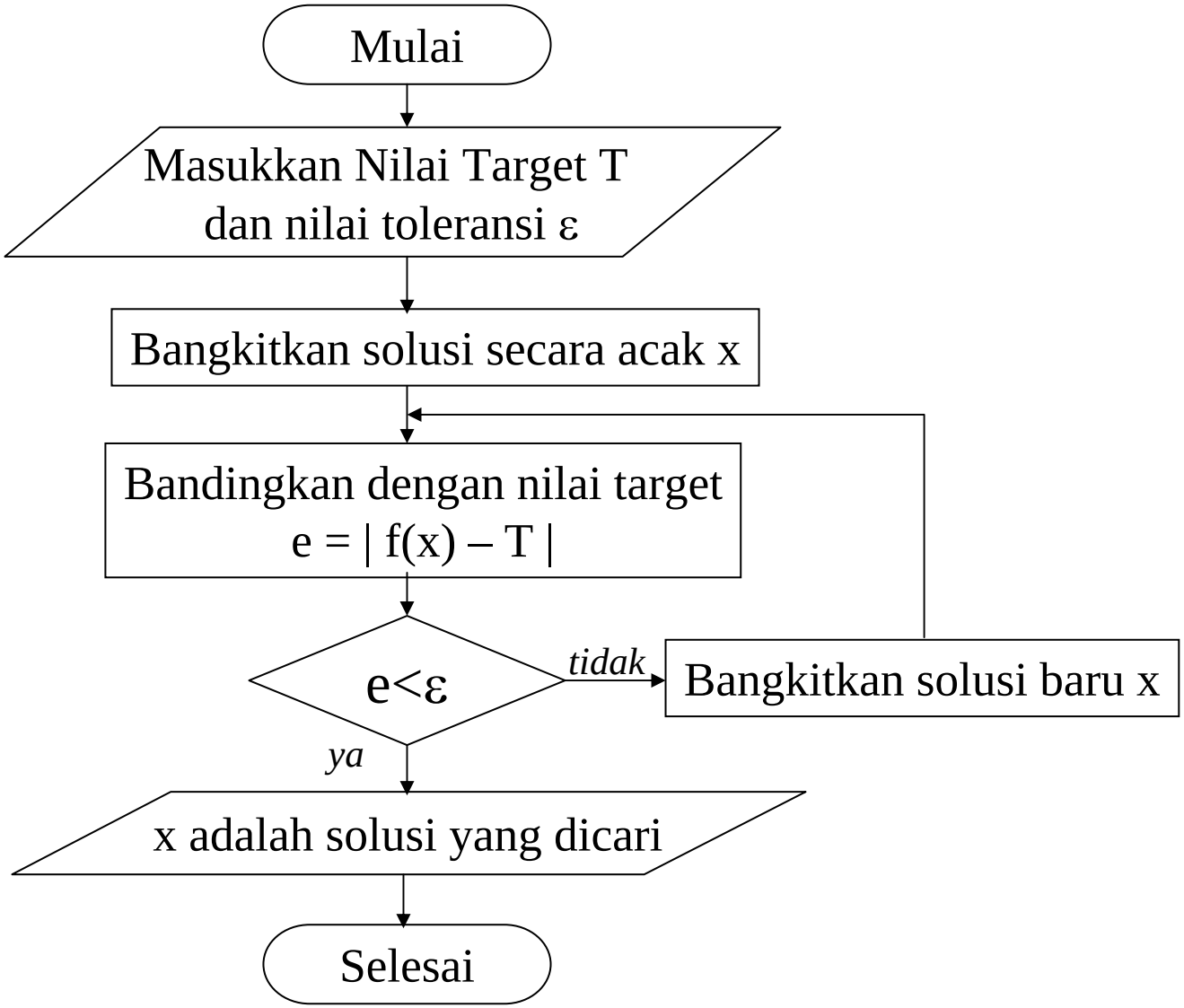
  Pencarian dilakukan hingga diperoleh nilai yang sama atau mendekati nilai target yang diberikan, sehingga akurasi pencarian ini ditentukan oleh kesamaan dengan nilai target. Beberapa contoh persoalan seperti pencarian angka atau kata yang disembunyikan, pencarian nilai fungsi yang mendekati nilai tertentu $f(x)=c$, pencarian pola biner dengan pembelajaran supervised.

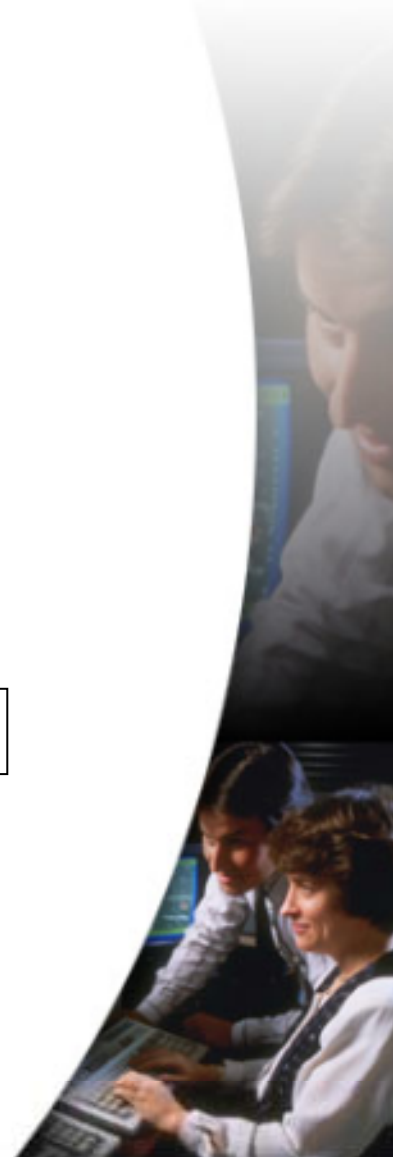
- Pencarian Acak Tanpa Nilai Target

  Pencarian dilakukan hingga diperoleh nilai tertinggi, nilai terendah atau error terkecil, karena tanpa nilai target maka solusi saat ini selalu dibandingkan dengan solusi sebelumnya untuk menunjukkan akurasi dari solusi. Beberapa contoh persoalan seperti pembelajaran unsupervised, pencarian nilai maksimal/minimal



Flowchart Pencarian Acak Dengan Nilai Target



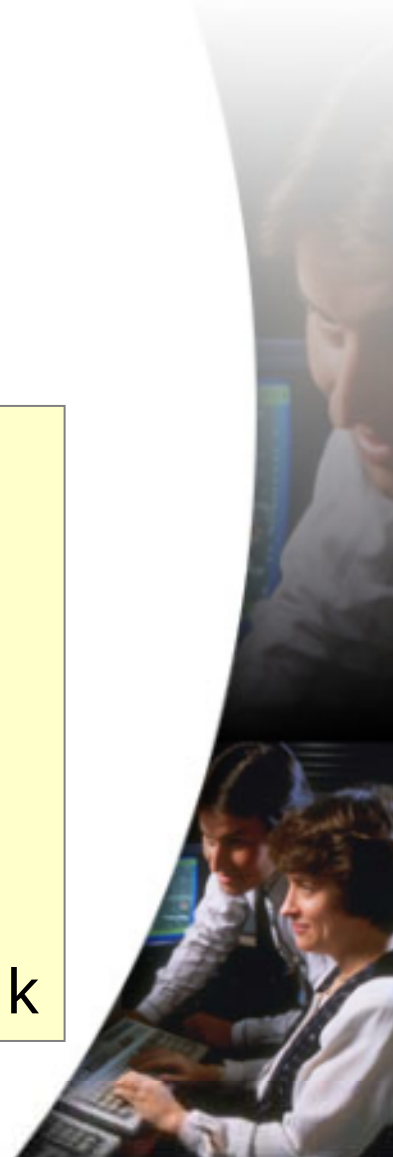


Pencarian Acak Untuk Menebak Angka Yang Disembunyikan Oleh Komputer

Angka yang disembunyikan

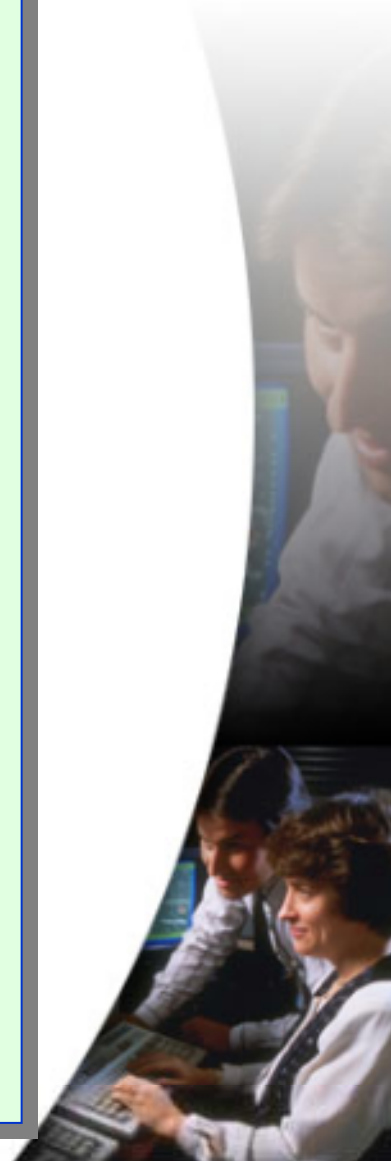
12

Bangkitkan bilangan acak 0-20	Penilaian terhadap setiap solusi acak
3	|3-12| = 9
8	|8-12| = 4
15	|15-12| = 3
4	|4-12| = 8
5	|5-12| = 7
10	|10-12| = 2
15	|15-12| = 3
12	|12-12| = 0 → Ok



Listing Program Bahasa C:
Pencarian Acak Untuk Menebak Angka Yang Disembunyikan Oleh Komputer

```
#include <stdlib.h>
#include <fstream.h>

void main()
{
int angkaTersembunyi, angkaTebakan;
int sw,n;
// Memasukkan angka tersembunyi
angkaTersembunyi=12;
// Mengulang tebakan sampai ketemu
// sw nilai yang menunjukkan sudah ketemu atau belum
// sw=0 belum ketemu, dan sw=1 sudah ketemu
// n jumlah tebakan
sw=0;
n=0;
while(sw==0) {
        n++;
        // Mengacak tebakan baru
        angkaTebakan=21*rand()/RAND_MAX;
        cout << "Tebakan ke " << n << " adalah " ;
        cout << angkaTebakan << endl;
        // Jika angka tebakan = angka tersembunyi maka
        // tebakan benar dan proses berhenti
        if(angkaTebakan==angkaTersembunyi) sw=1;
}
}
```



Pencarian Acak Untuk Menebak Kata Yang Disembunyikan Oleh Komputer

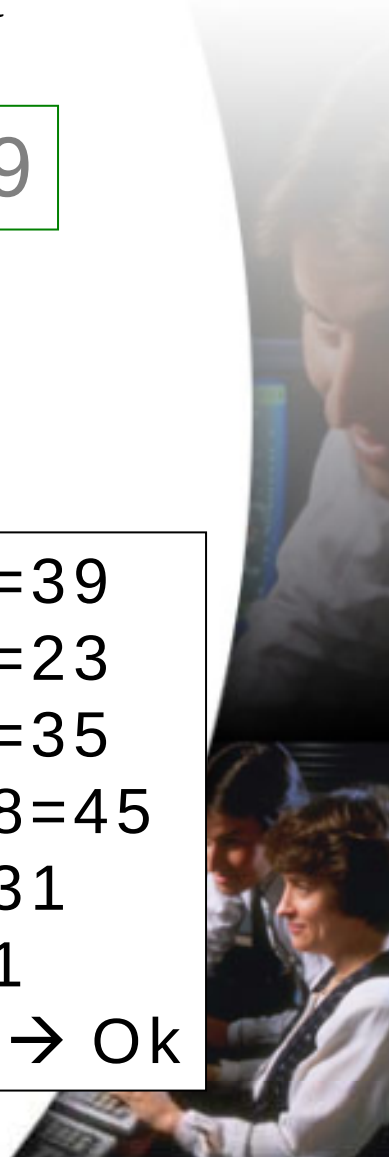
Kata yang disembunyikan

EEPIS

Nilai numerik target dari urutan karakter

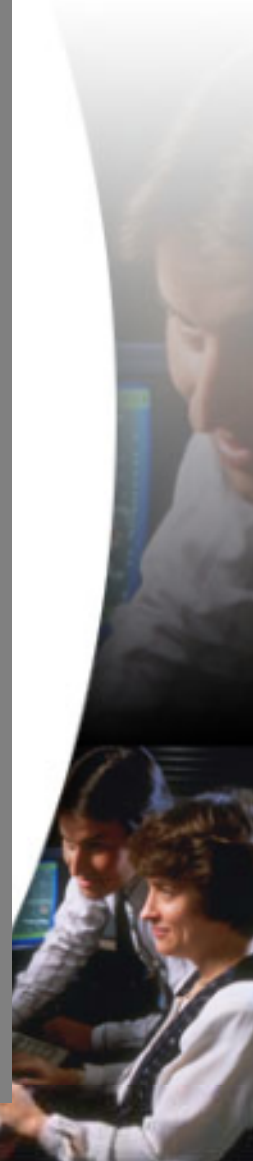
5 5 16 9 19

Kata yang dibangkitkan secara acak	Nilai numerik dari kata acak	Error
TGAPI	20 7 1 16 9	5+2+15+7+10=39
EPEIR	5 16 5 9 18	0+11+11+0+1=23
PEABU	16 5 1 2 21	11+0+15+7+2=35
ROTIA	18 15 20 9 1	13+10+4+0+18=45
EMBAN	5 13 2 1 14	0+8+14+4+5=31
IMPAS	9 13 16 1 19	4+8+0+9+0=21
EEPIS	5 5 16 9 19	0+0+0+0+0=0 → Ok



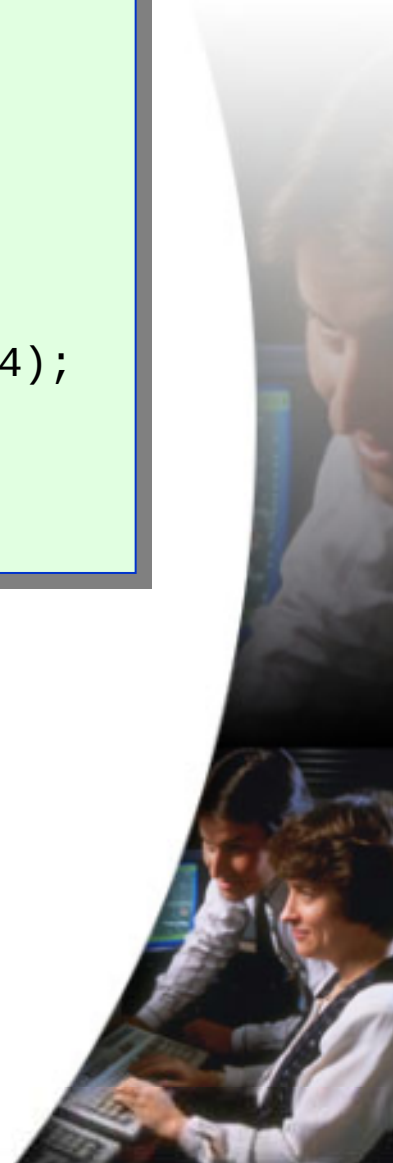
Listing Program Bahasa C:
Pencarian Acak Untuk Menebak Kata Yang Disembunyikan Oleh Komputer [1]

```
#include <stdlib.h>
#include <fstream.h>
#include <math.h>

void main()
{
        int kataTarget[5]={5,5,16,9,19};
        int kataAcak[5],iterasi,MaxIter=100,i,E;
        int kata[5],Emin;
        // Mengacak kata sebagai solusi awal
        for(i=0;i<5;i++)
                kata[i]=26*rand()/RAND_MAX+1;
        // Menghitung Error
        Emin=0;
        for(i=0;i<5;i++)
                Emin+=abs(kata[i]-kataTarget[i]);

        for(iterasi=1;iterasi<=MaxIter;iterasi++){
                // Mengacak kata
                for(i=0;i<5;i++)
                        kataAcak[i]=26*rand()/RAND_MAX+1;
                // Menghitung Error
                E=0;
                for(i=0;i<5;i++)
                        E+=abs(kataAcak[i]-kataTarget[i]);
```

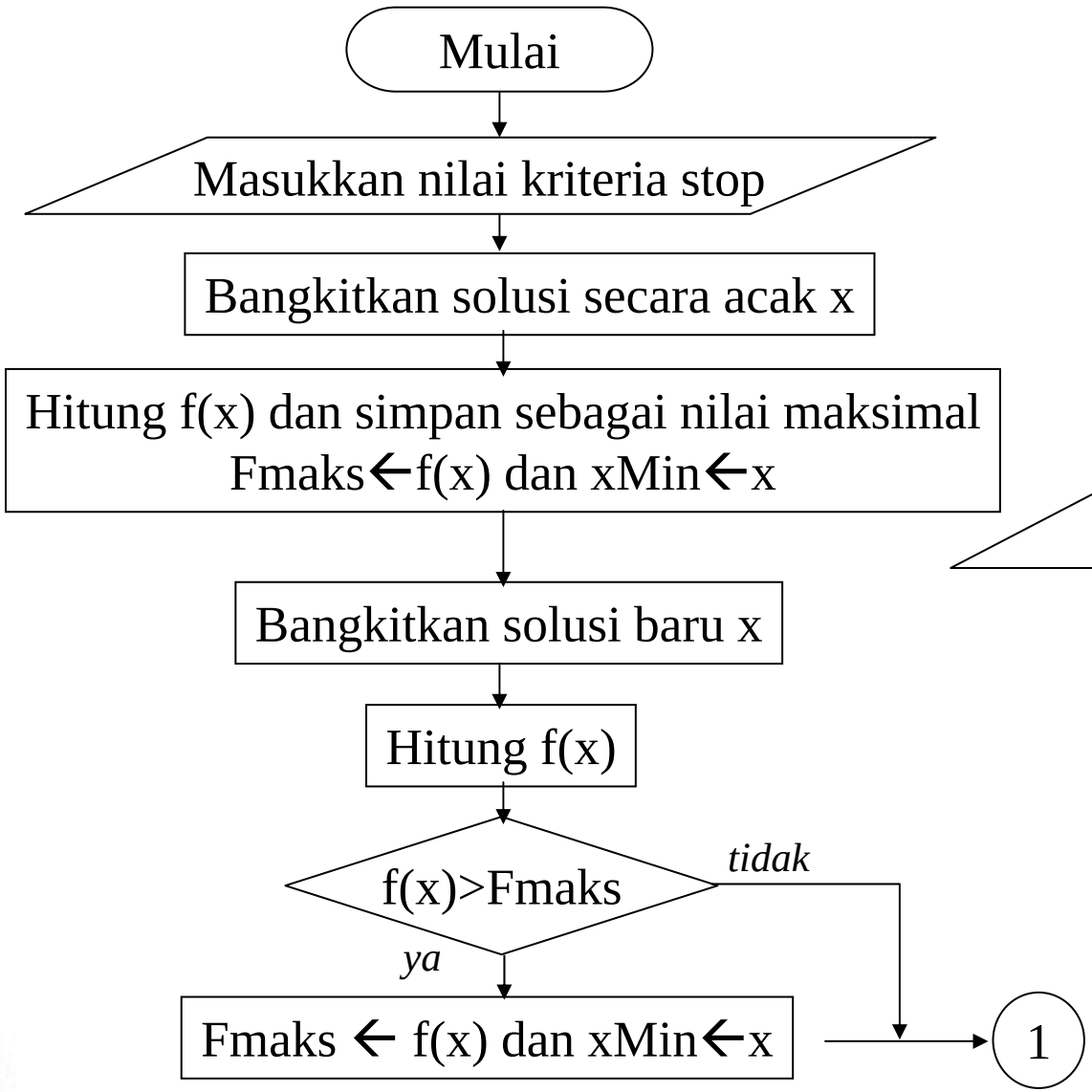


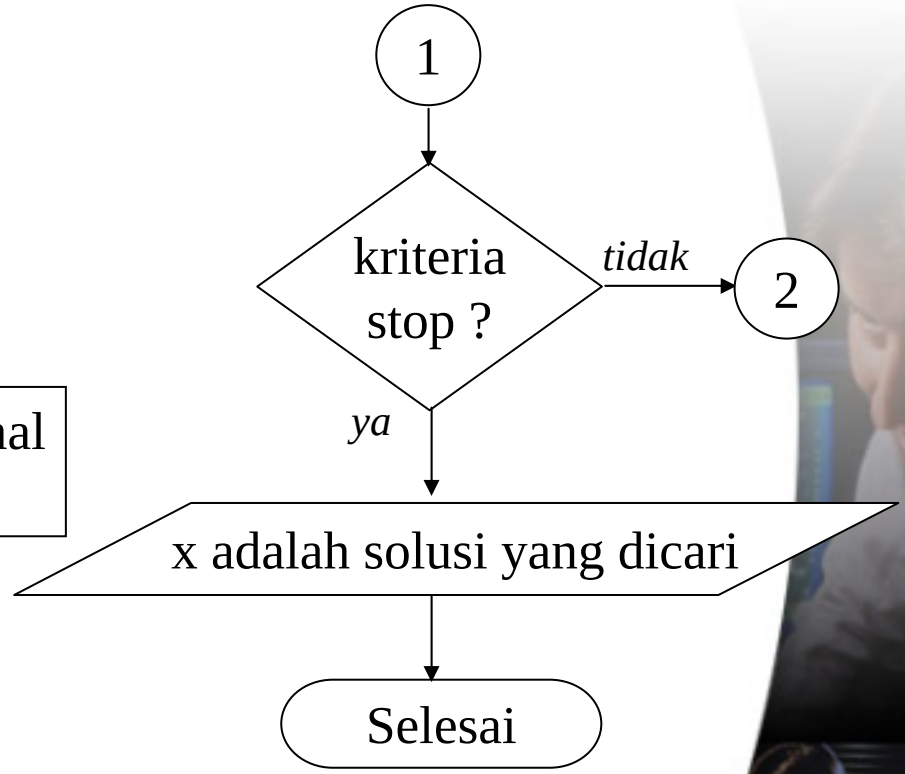
Listing Program Bahasa C:
Pencarian Acak Untuk Menebak Kata Yang Disembunyikan Oleh Komputer [2]

```
                // Membandingkan Error
                if(E<Emin) {
                        for(i=0;i<5;i++)
                                kata[i]=kataAcak[i];
                        Emin=E;
                }
                // Menampilkan hasil pencarian
                cout << iterasi << " : ";
                for(i=0;i<5;i++) cout << char(kata[i]+64);
                cout << "  Error = " << Emin << endl;
        }
}
```



Flowchart Pencarian Acak Tanpa Nilai Target Untuk Mencari Nilai Maksimal





Pencarian Acak Untuk Mencari Nilai Maksimal F(x)

Fungsi

y=f(x)

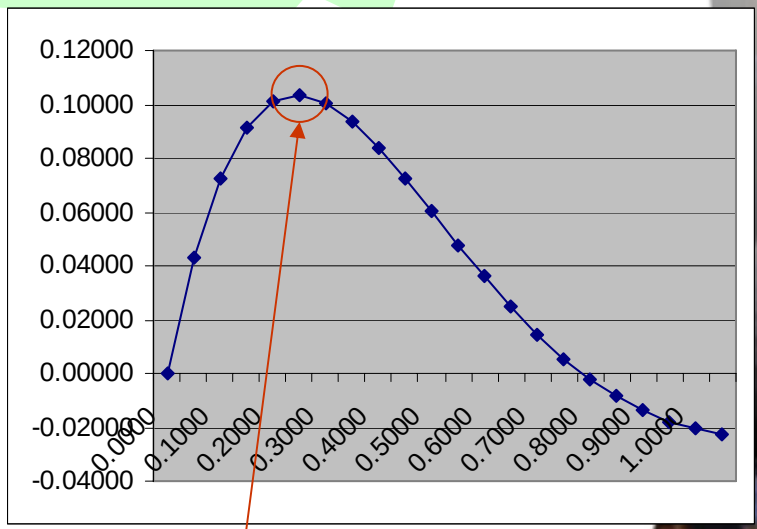
Contoh fungsi

$$f(x) = xe^{-3x}\cos(2x)$$

Nilai x yang dibangkitkan secara acak	Nilai F(x)
0.2886	0.10175
0.8139	-0.00403
0.0374	0.03336
0.0269	0.02475
0.5846	0.03955
0.8306	-0.00621
0.3238	0.09776
0.5135	0.05693
0.5248	0.05412
0.8528	-0.00887
0.2413	0.10363
0.8528	-0.00887

Nilai f(x) yang paling besar

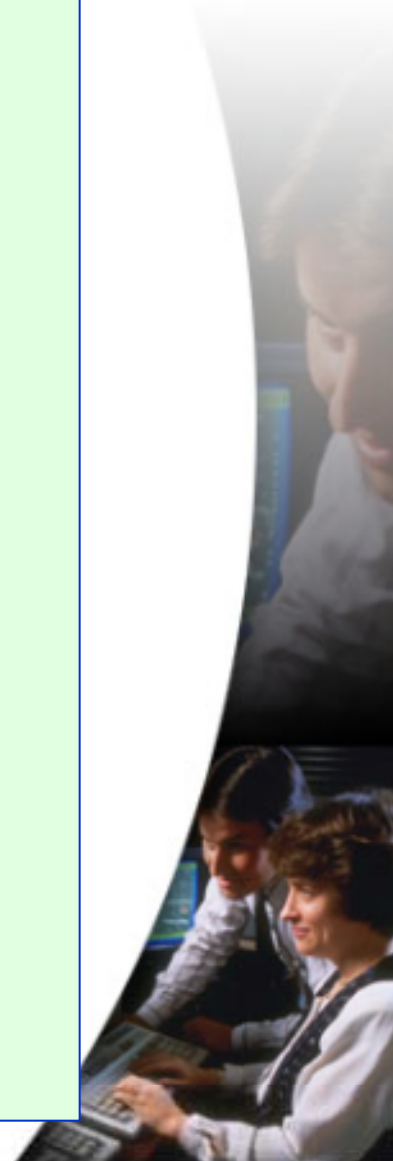
solusi



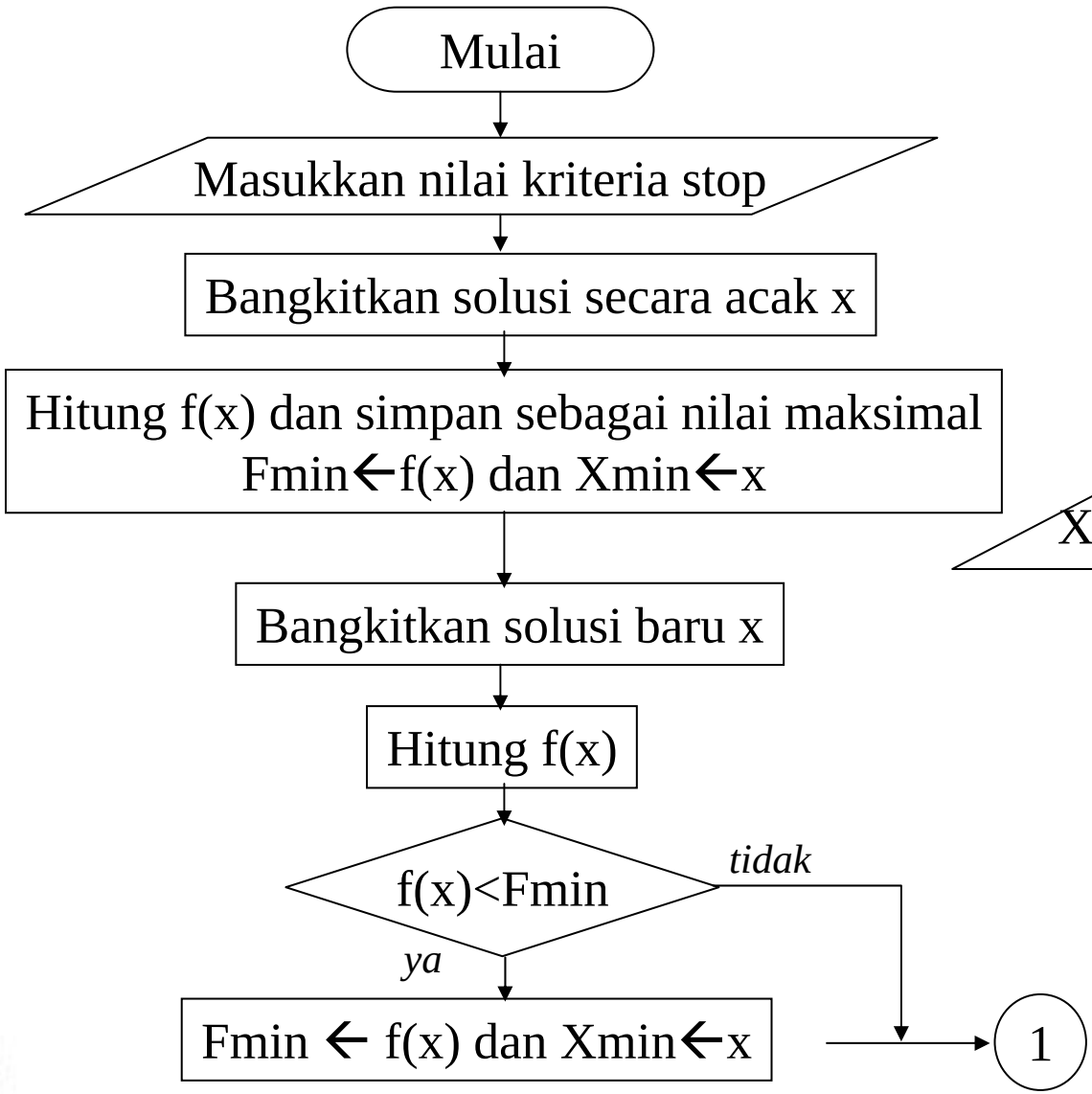
Nilai maksimal

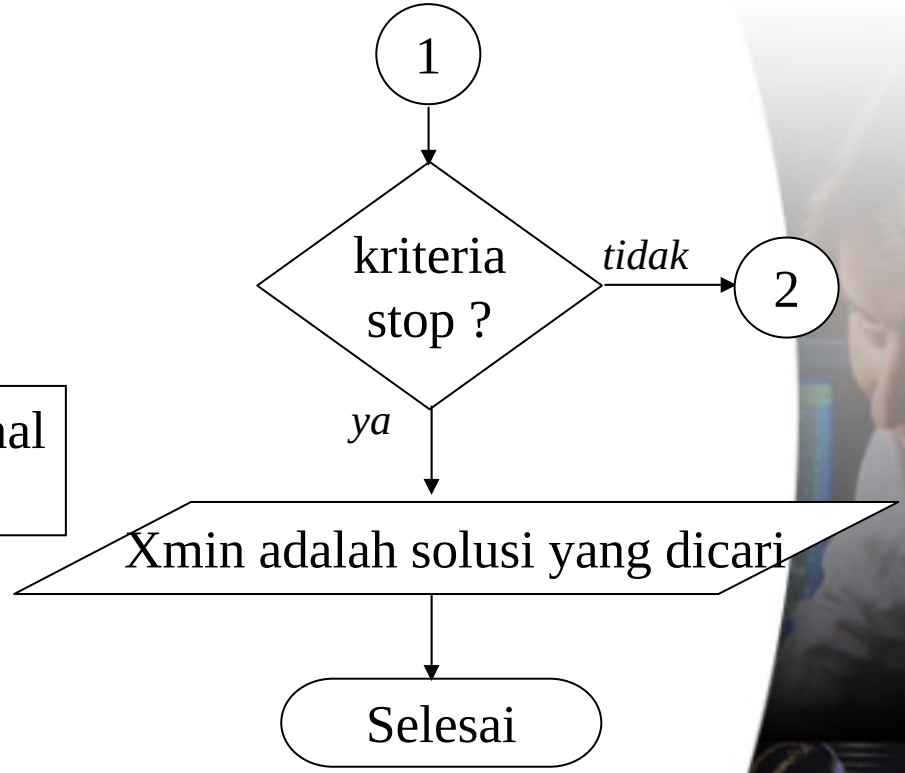
Listing Program Bahasa C:
Pencarian Acak Untuk Mencari Nilai Maksimal F(x)

```
#include <stdlib.h>
#include <math.h>
#include <fstream.h>

float fungsi(float u){
        return (float)(u*exp(-3*u)*cos(2*u));
}

void main()
{
        float x,y,xMaks,yMaks;
        int iterasi, MaxIter=100;
        // Inisialisasi Solusi
        xMaks=(float)rand()/RAND_MAX;
        yMaks=fungsi(xMaks);

        for(iterasi=1;iterasi<=MaxIter;iterasi++){
                // Mengacak Solusi Baru
                x=(float)rand()/RAND_MAX;
                y=fungsi(x);
                if(y>yMaks) {
                        xMaks=x; yMaks=y;
                }
                cout << iterasi << ":  f(" << xMaks;
                cout << ") = " << yMaks << endl;
        }
}
```



Flowchart Pencarian Acak Tanpa Nilai Target Untuk Mencari Nilai Minimal





Pencarian Acak Untuk Mencari Nilai Minimal F(x)

Fungsi

y=f(x)

Contoh fungsi

$$f(x) = xe^{-3x}\cos(2x)$$

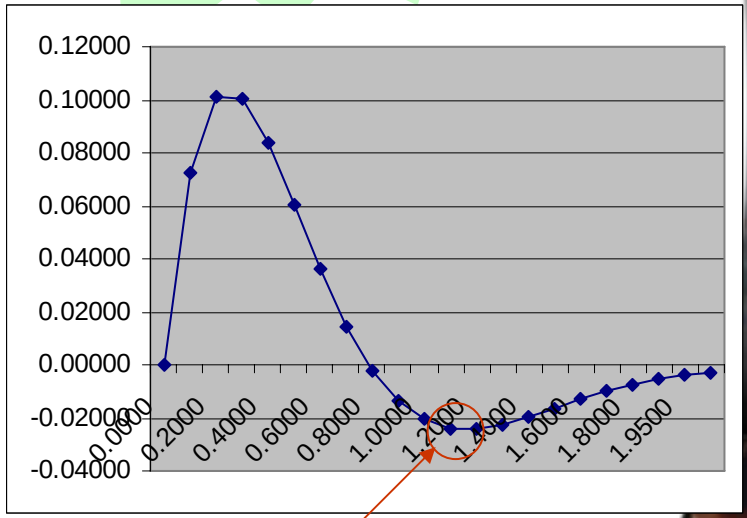
Nilai x yang dibangkitkan secara acak

Nilai F(x)

1.8826	-0.00539
1.6544	-0.01141
0.0671	0.05439
1.9757	-0.00363
1.1339	-0.02425
0.0182	0.01722
1.4709	-0.01748
0.0641	0.05242
0.0664	0.05390
1.1119	-0.02404
1.9180	-0.00467

solusi

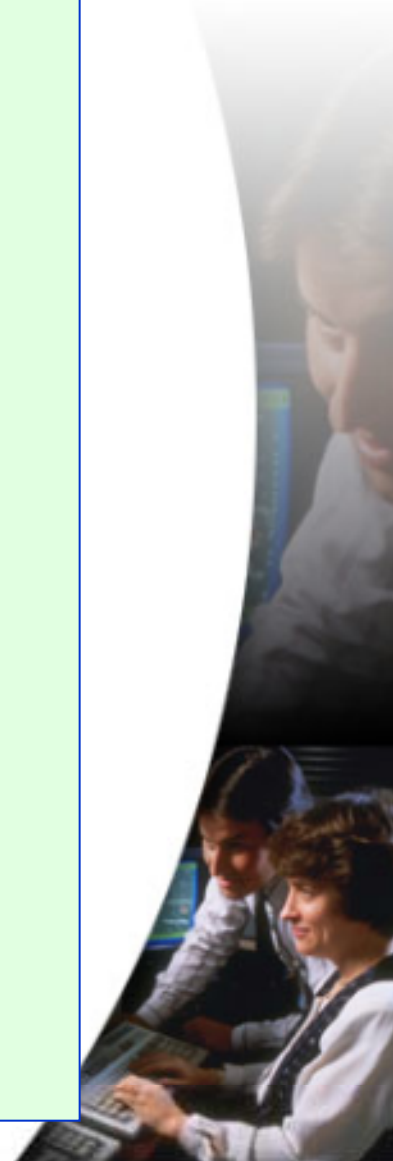
Nilai f(x) yang paling kecil



Nilai minimal

Listing Program Bahasa C:
Pencarian Acak Untuk Mencari Nilai Minimal F(x)

```
#include <stdlib.h>
#include <math.h>
#include <fstream.h>

float fungsi(float u){
        return (float)(u*exp(-3*u)*cos(2*u));
}

void main()
{
        float x,y,xMin,yMin;
        int iterasi, MaxIter=100;
        // Inisialisasi Solusi
        xMin=(float)rand()/RAND_MAX;
        yMin=fungsi(xMin);

        for(iterasi=1;iterasi<=MaxIter;iterasi++){
                // Mengacak Solusi Baru
                x=(float)rand()/RAND_MAX;
                y=fungsi(x);
                if(y<yMin) {
                        xMin=x; yMin=y;
                }
                cout << iterasi << ":  f(" << xMin;
                cout << ") = " << yMin << endl;
        }
}
```



Pencarian Acak Untuk Mencari Akar Persamaan Transendental

Persamaan transedental

y=f(x)

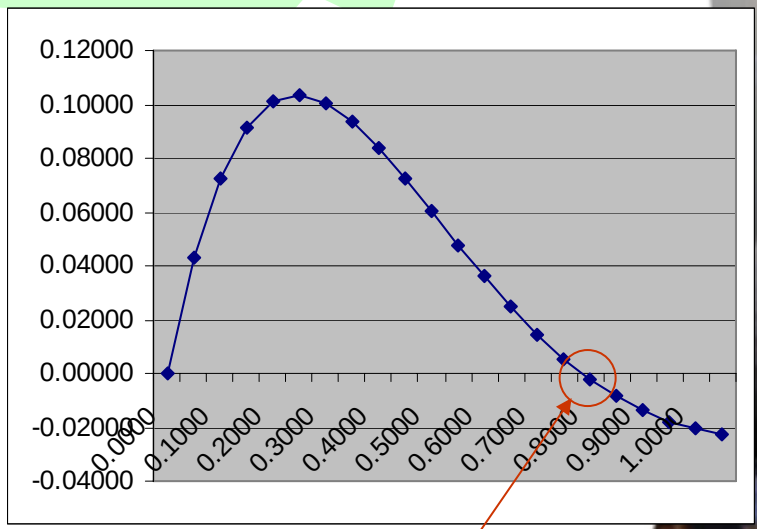
Contoh persamaan yang diselesaikan

$$f(x) = xe^{-3x}\cos(2x)$$

Nilai x yang dibangkitkan secara acak	Nilai F(x)
0.2886	0.10175
0.8139	-0.00403
0.0374	0.03336
0.0269	0.02475
0.5846	0.03955
0.8306	-0.00621
0.3238	0.09776
0.5135	0.05693
0.5248	0.05412
0.8528	-0.00887
0.2413	0.10363
0.8528	-0.00887

solusi

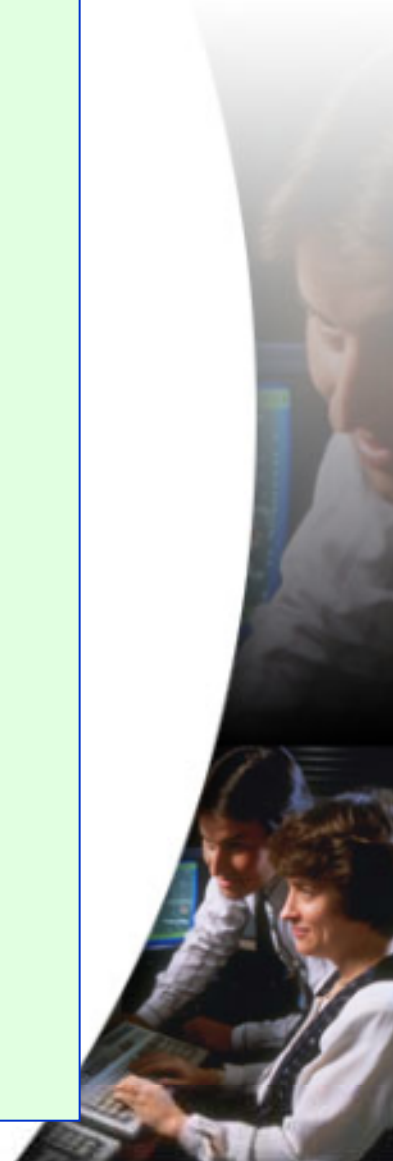
Nilai f(x) yang paling dekat dengan nol



akar

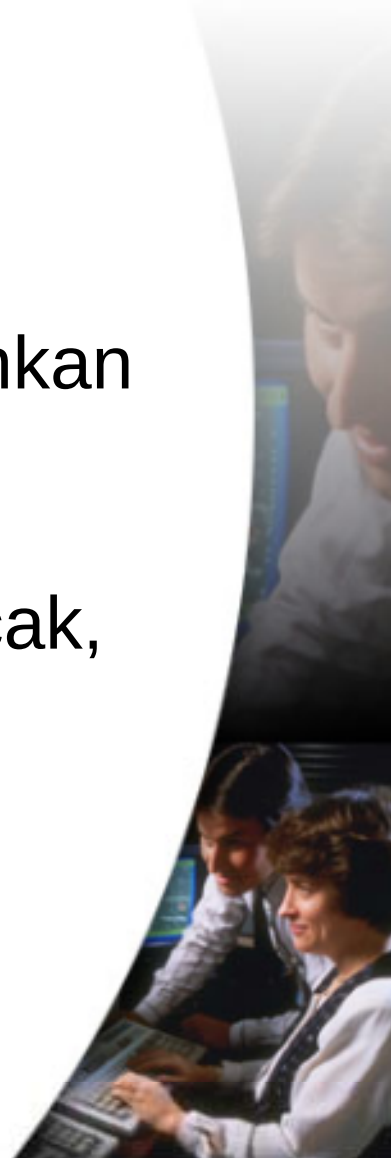
Listing Program Bahasa C:
Pencarian Acak Untuk Mencari Akar Persamaan

```
#include <stdlib.h>
#include <math.h>
#include <fstream.h>

float fungsi(float u){
        return (float)(u*exp(-3*u)*cos(2*u));
}

void main()
{
        float x,y,xMin,yMin;
        int iterasi, MaxIter=100;
        // Inisialisasi Solusi
        xMin=(float)rand()/RAND_MAX;
        yMin=fabs(fungsi(xMin));

        for(iterasi=1;iterasi<=MaxIter;iterasi++){
                // Mengacak Solusi Baru
                x=(float)rand()/RAND_MAX;
                y=fabs(fungsi(x));
                if(y<yMin) {
                        xMin=x; yMin=y;
                }
                cout << iterasi << ":  f(" << xMin;
                cout << ") = " << yMin << endl;
        }
}
```

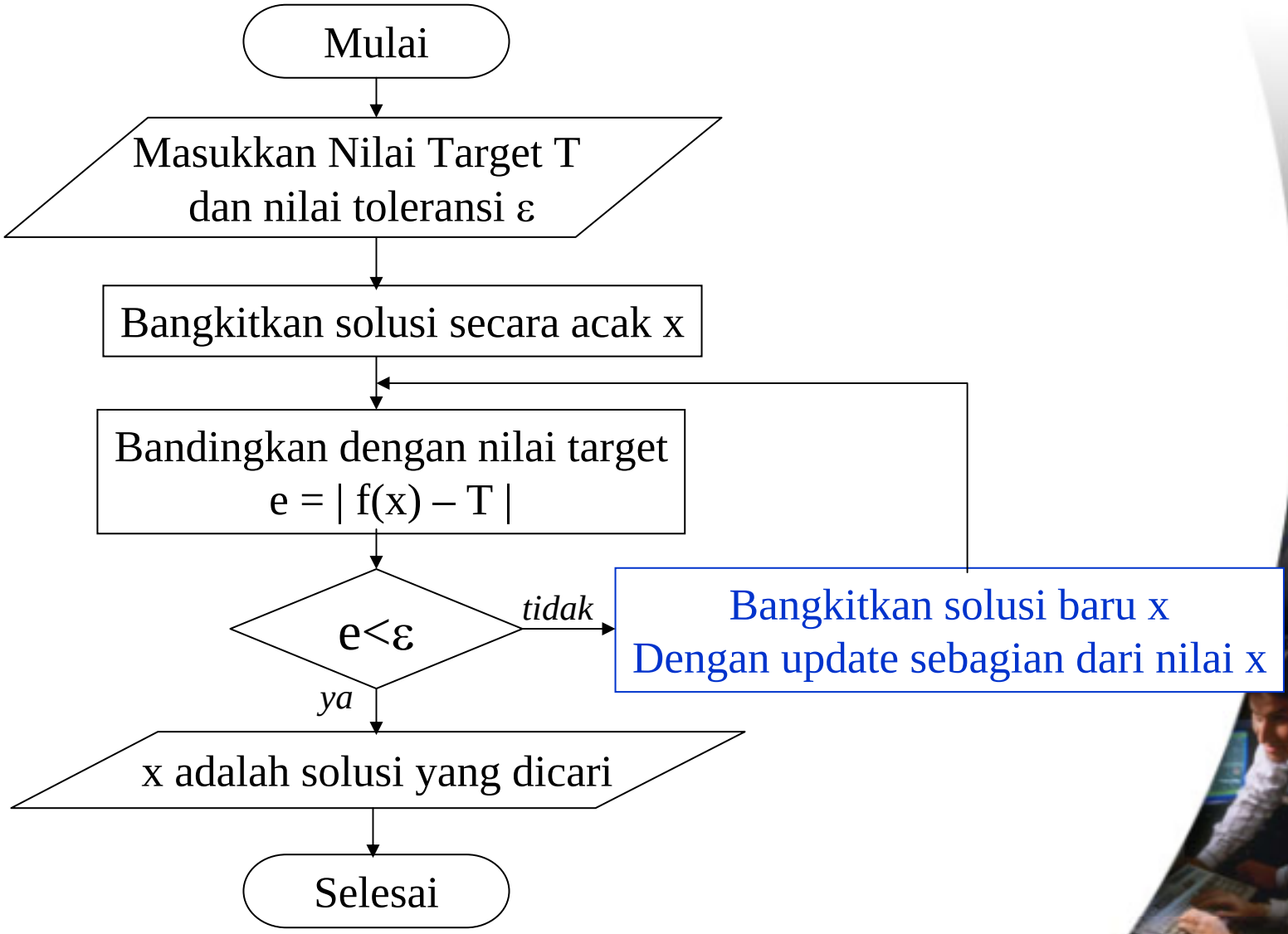


Monte Carlo Sebagai Metode Pancarian Acak

- Metode pencarian acak (random walk) mengganti semua kombinasi dengan kombinasi yang baru hingga diperoleh kombinasi terbaik. Hal ini menyebabkan pencarian menjadi sangat lama atau bahkan hasil yang diperoleh bukan hasil yang optimal.
- Monte Carlo seperti halnya pencarian acak, hanya saja penggantian dilakukan pada sebagian elemen dari kombinasi solusi. Penggantian bisa pencakan ulang atau pergeseran. Hal ini merupakan ide yang sangat baik.

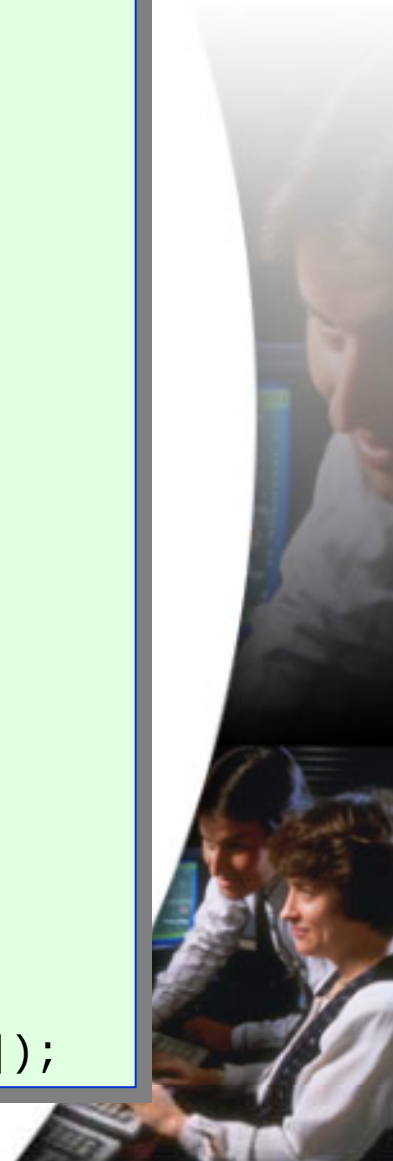


Flowchart Metode Monte Carlo Dengan Nilai Target



Listing Program Bahasa C:
Monte Carlo Untuk Menebak Kata Yang Disembunyikan Oleh Komputer [1]

```
#include <stdlib.h>
#include <fstream.h>
#include <math.h>

void main()
{
        int kataTarget[5]={5,5,16,9,19};
        int kataAcak[5],iterasi,MaxIter=100,i,E;
        int kata[5],Emin;
        // Mengacak kata sebagai solusi awal
        for(i=0;i<5;i++)
                kata[i]=26*rand()/RAND_MAX+1;
        // Menghitung Error
        Emin=0;
        for(i=0;i<5;i++)
                Emin+=abs(kata[i]-kataTarget[i]);

        for(iterasi=1;iterasi<=MaxIter;iterasi++){
                // Mengacak satu karakter dari kata
                i=5*rand()/RAND_MAX;
                kataAcak[i]=26*rand()/RAND_MAX+1;
                // Menghitung Error
                E=0;
                for(i=0;i<5;i++)
                        E+=abs(kataAcak[i]-kataTarget[i]);
```



Listing Program Bahasa C:

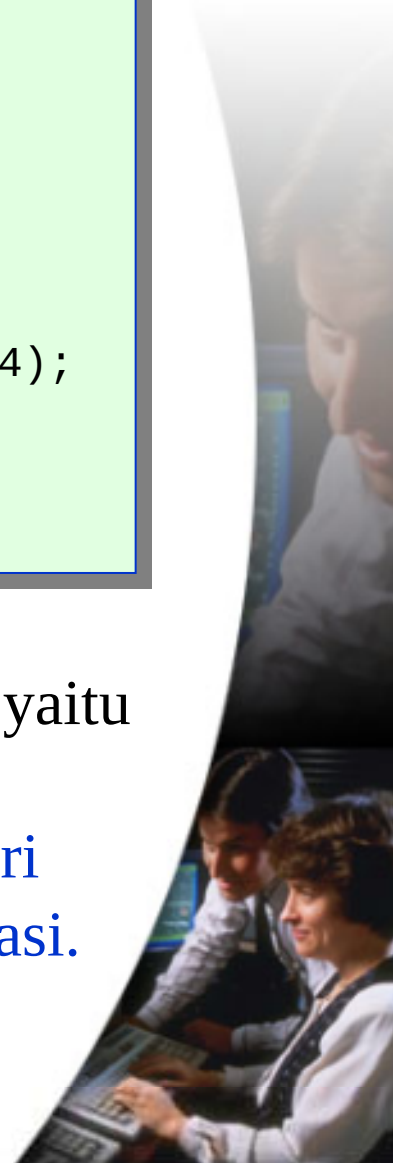
Pencarian Acak Untuk Menebak Kata Yang Disembunyikan Oleh Komputer [2]

```
                // Membandingkan Error
                if(E<Emin) {
                        for(i=0;i<5;i++)
                                kata[i]=kataAcak[i];
                        Emin=E;
                }
                // Menampilkan hasil pencarian
                cout << iterasi << " : ";
                for(i=0;i<5;i++) cout << char(kata[i]+64);
                cout << "  Error = " << Emin << endl;
        }
}
```

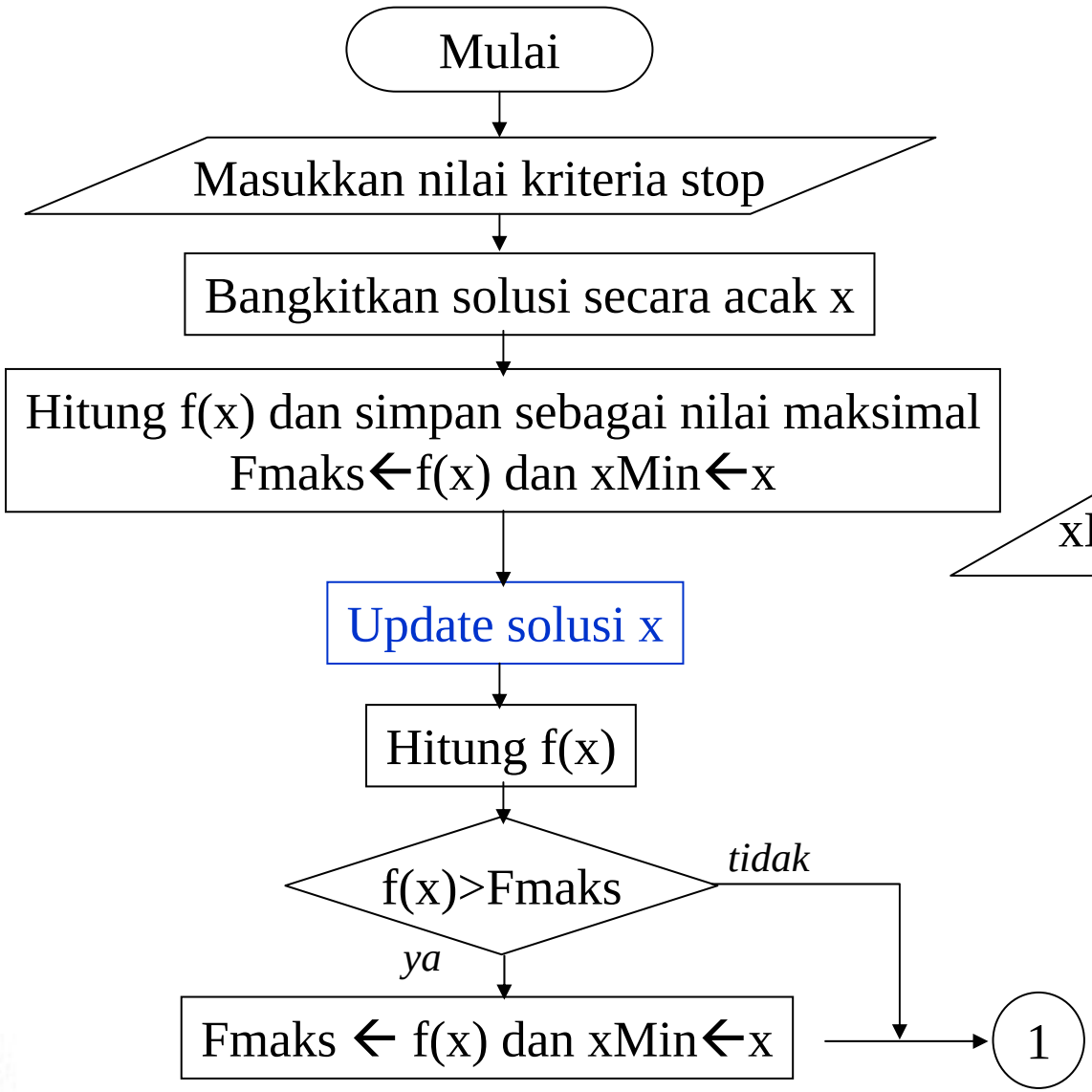
Perbandingan Hasil Pencarian Acak dan Monte Carlo yaitu pada pencapaian hasil optimal:

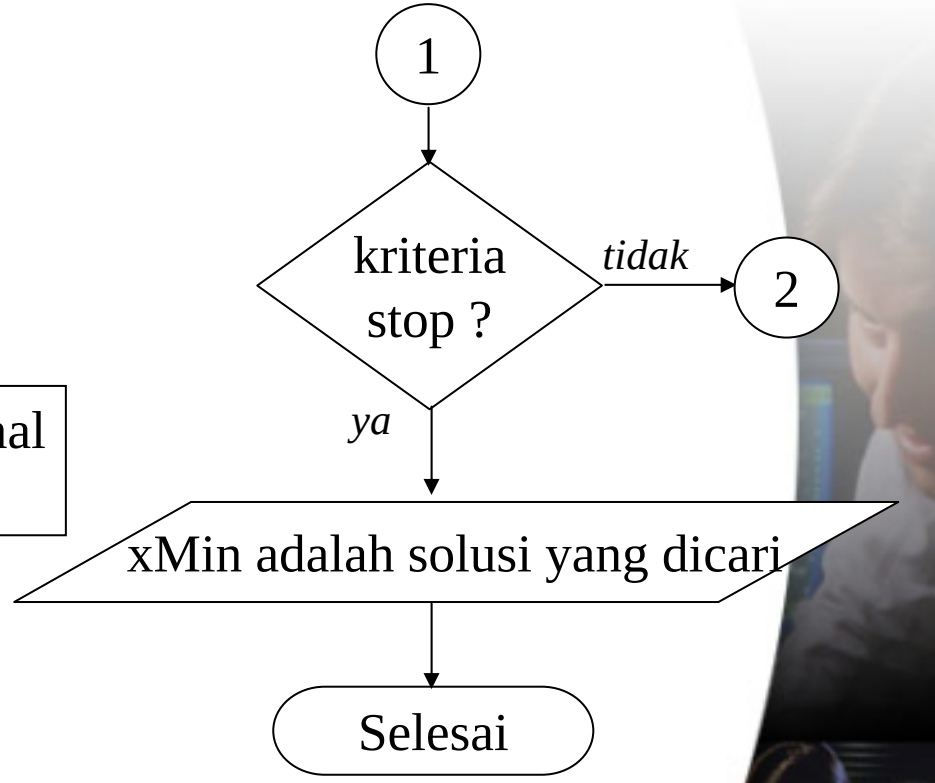
Pencarian acak → Tidak menghasilkan kata yang dicari

Monte Carlo → Mendapatkan hasil setelah ±1000 iterasi.



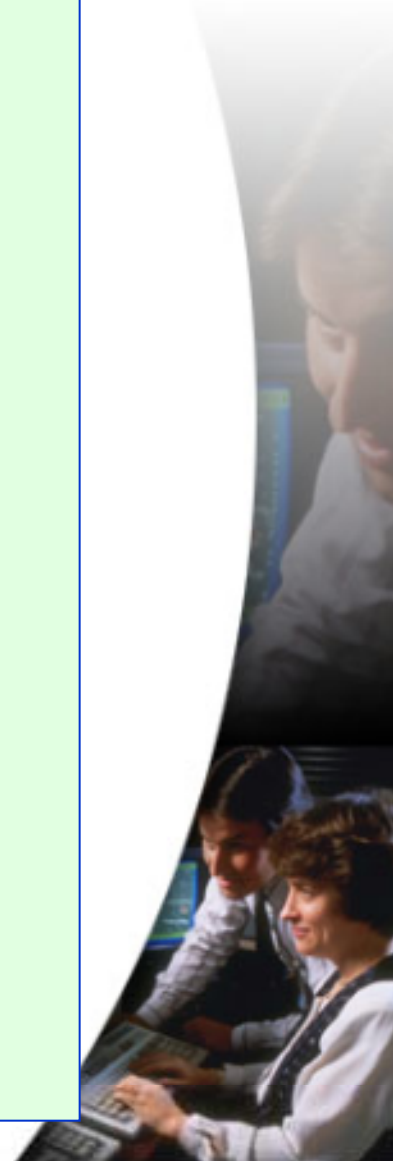
Flowchart Pencarian Acak Tanpa Nilai Target Untuk Mencari Nilai Maksimal



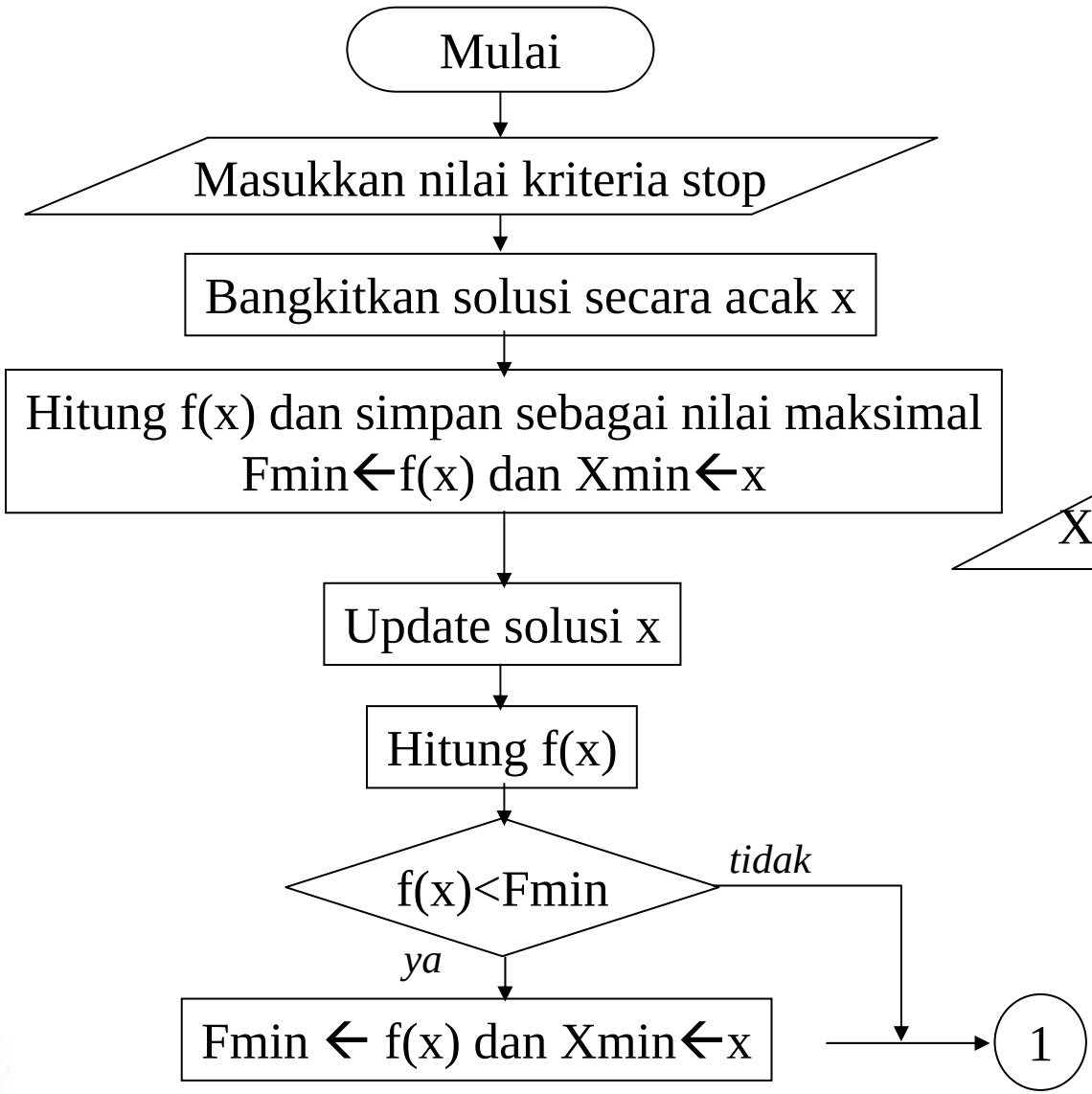


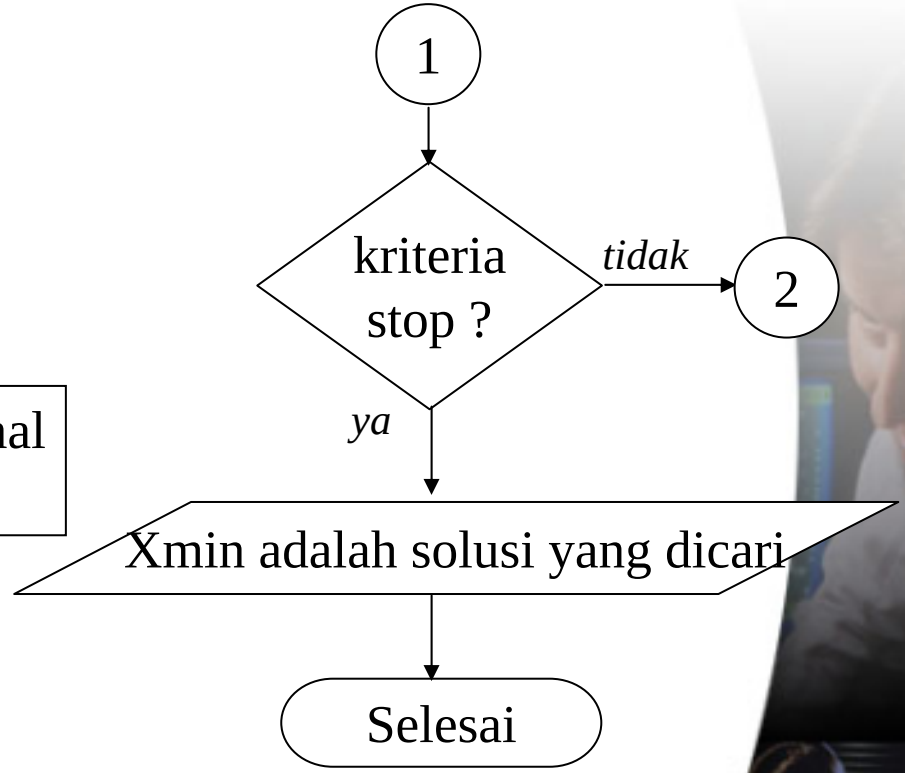
Listing Program Bahasa C:
Monte Carlo Untuk Mencari Nilai Maksimal F(x)

```
#include <stdlib.h>
#include <math.h>
#include <fstream.h>

float fungsi(float u){
        return (float)(u*exp(-3*u)*cos(2*u));
}

void main()
{
        float x,y,xMaks,yMaks;
        int iterasi, MaxIter=100;
        // Inisialisasi Solusi
        xMaks=(float)rand()/RAND_MAX;
        yMaks=fungsi(xMaks);

        for(iterasi=1;iterasi<=MaxIter;iterasi++){
                // Update Solusi Baru
                x=Xmin+(float)rand()/RAND_MAX-0.5;
                y=fungsi(x);
                if(y>yMaks) {
                        xMaks=x; yMaks=y;
                }
                cout << iterasi << ":  f(" << xMaks;
                cout << ") = " << yMaks << endl;
        }
}
```



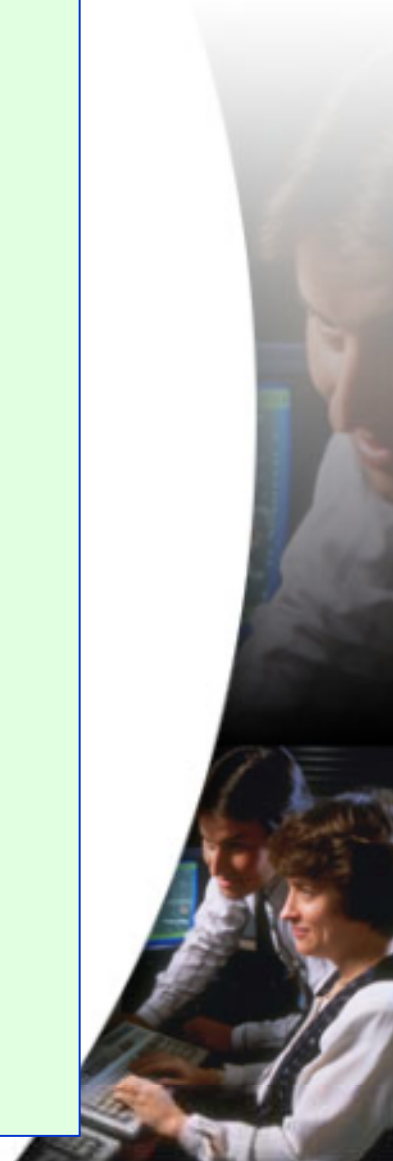
Flowchart Pencarian Acak Tanpa Nilai Target Untuk Mencari Nilai Minimal



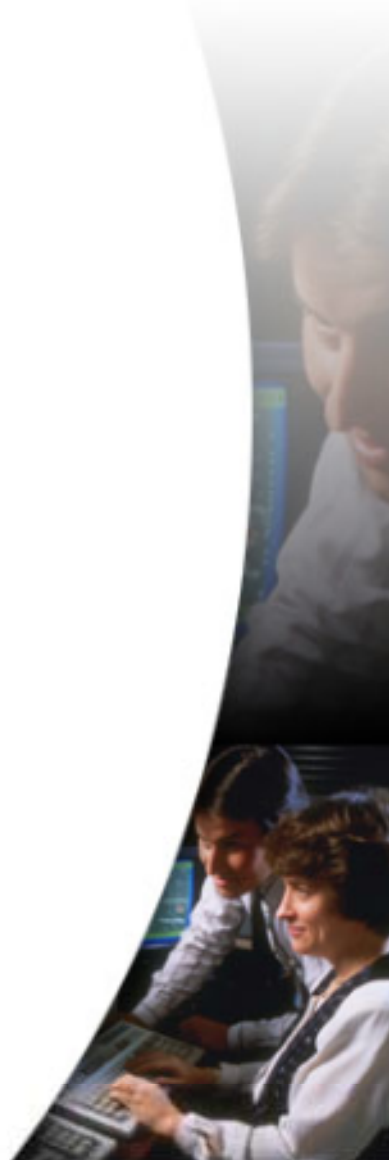


Listing Program Bahasa C:
Monte Carlo Untuk Mencari Nilai Minimal F(x)

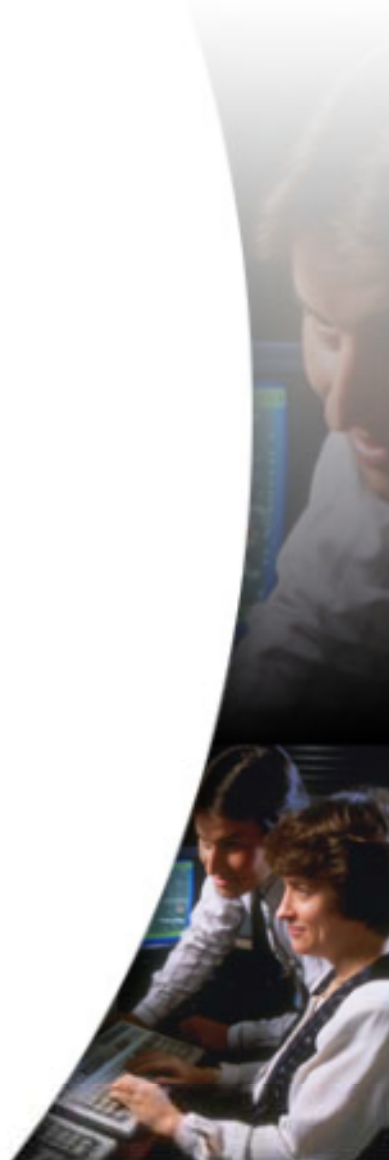
```
#include <stdlib.h>
#include <math.h>
#include <fstream.h>

float fungsi(float u){
        return (float)(u*exp(-3*u)*cos(2*u));
}

void main()
{
        float x,y,xMin,yMin;
        int iterasi, MaxIter=100;
        // Inisialisasi Solusi
        xMin=(float)rand()/RAND_MAX;
        yMin=fungsi(xMin);

        for(iterasi=1;iterasi<=MaxIter;iterasi++){
                // Mengacak Solusi Baru
                x=Xmin+(float)rand()/RAND_MAX-0.5;
                y=fungsi(x);
                if(y<yMin) {
                        xMin=x; yMin=y;
                }
                cout << iterasi << ":  f(" << xMin;
                cout << ") = " << yMin << endl;
        }
}
```



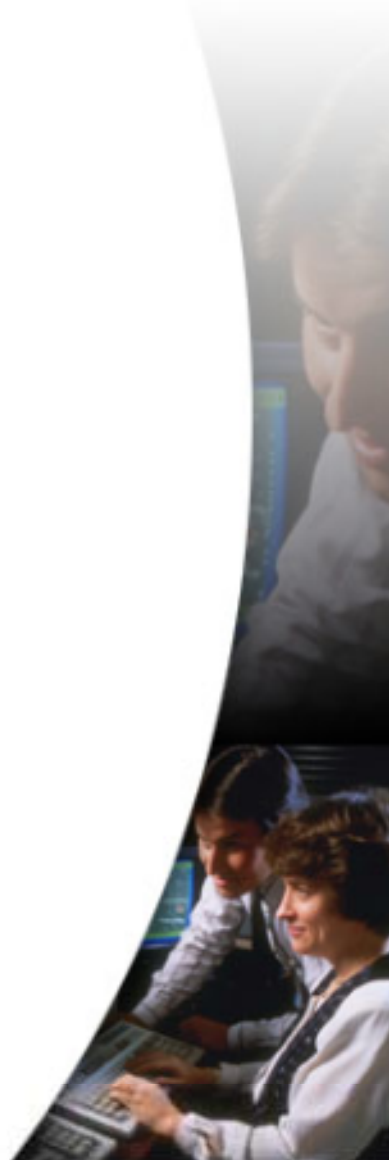
Note:



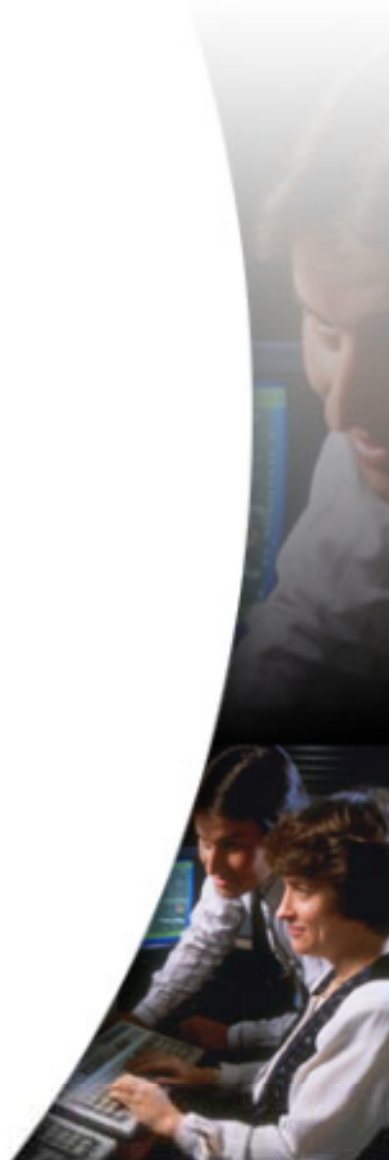
Note:



Note:



Note:



Note: