計算機程式與應用hw6

H34996158 侯瑋明g6

/\*

This code can be compiled and run ok.

Simulate the system of demand, inventory, cost, ordering quantity and safety stock.

Pass through four different methods to calculate some value.

usage (how to run):

./hw6 (then input filename)

input file:

HW6\_input.txt

output files:

none

compile (how to compile):

g++ -o hw6 hw6.cpp

pseudocode:

part2:

for i = 0 ~ T

random variable

part3.1:

for i = 0 ~ 3

Plc[i] allocate memory

part3.2:

Safety

while (I[t-1] - all demand) small than safety stock

if true O[t]=Q1 or Q2

else O[t]=0

Reorder

while Inventory small than Reorder

if true O[t]=Q1 or Q2

else O[t]=0

coded by Wei-Ming Hou, group: 6, ID: h34996158, email: jacob0701@hotmail.com.tw

date: 2011.05.22

\*/

#include <iostream>

#include <fstream>

#include <string>

#include <math.h>

#include <ctime>

#include <cstdlib>

#include <iomanip>

using namespace std;

void initialize\_p(const char \*filename, int \*, int \*\*, int \*, int \*, double \*\*, int \*, int \*, int \*, int \*, int \*, int \*, int \*, int \*, int \*, int \*);

void GetProbability (int , int , double \*&);

int GetDemand(int , double \*&, int, double);

typedef struct {

int M, Q; //分別代表M1或M2，Q1或Q2

int \*I, \*O; //分別代表I[ ],O[ ]

int \*DC; //代表日花費DC[ ]

int TCost; //代表總花費

} Policy;

void GetCost1(Policy \*, int, int , int, int \*, int , int , int , int , int);

void GetCost2(Policy \*, int, int , int, int \*, int , int , int , int , int);

void Print\_Policy(Policy \*, int , int \*, int );

int main()

{

int i, j, min;

int T, \*I, K, l, M1, M2, S, H, C, Q1, Q2, L, A, R, \*O, \*DC, \*D;

double \*p;

char filename[50];

cout << "Please intput filename: ";

cin >> filename;

initialize\_p(filename ,&T, &I, &K, &l, &p, &M1, &M2, &S, &H, &C, &Q1, &Q2, &L, &A, &R);

//----------------------------------------------------------------------------------

//---PART2 以隨機亂數產生對應的顧客需求量-------------------------------------------

GetProbability(l, K, p);

D = new int [T+1];

double val;

srand(static\_cast<unsigned int>(time(NULL)));

for(i=0; i<=T; i++){ //將D陣列歸0

D[i]=0;}

for(i=0; i<=T; i++){ //隨機變數傳入GetDemand函式

val = rand()\*1.0/(RAND\_MAX);

D[i]= GetDemand(K, p, T, val);}

//----------------------------------------------------------------------------------

//---PART3.1設定4種機制之相關參數及allocate memory----------------------------------

Policy Plc[4];

for(i=0; i<4; i++){ //allocate memory

Plc[i].I = new int[T+1];

Plc[i].O = new int[T+1];

Plc[i].DC = new int[T+1];

Plc[i].I[0]=I[0];

Plc[i].O[0]=0;

Plc[i].DC[0]=0;

Plc[i].TCost=0;

if(i==0||i==1){

Plc[i].M = M1;

Plc[i].Q = Q1;}

else{

Plc[i].M = M2;

Plc[i].Q = Q2;}

}

for(i=0; i<4; i++){ //將O陣列中負值項歸0

for(j=T; j>=-L; j--){

Plc[i].O[j]=0;}}

//----------------------------------------------------------------------------------

//---PART3.2計算各機制每天之存缺貨量I[t]、訂貨量O[t]、日花費DC[t]，及其總花費TCost--

GetCost1(Plc, T, K, \*I, D, S, H, C, L, A);

GetCost2(Plc, T, K, \*I, D, S, H, C, L, R);

for(i=0; i<4; i++){ //四種機制中找出最小成本

for(j=1; j<4; j++){

if(Plc[i].TCost > Plc[j].TCost)

min=j;}}

cout<<endl;

Print\_Policy( Plc, T, D, min); //印出

//----------------------------------------------------------------------------------

system("pause");

return 0;

}

void initialize\_p(const char \*filename, int \*T, int \*\*I, int \*K, int \*l, double \*\*p, int \*M1, int \*M2, int \*S, int \*H, int \*C, int \*Q1, int \*Q2, int \*L, int \*A, int \*R)

{

char z;

ifstream inFile(filename);

while(!inFile.eof()) //讀檔

{

inFile >> z;

switch(z)

{

case 'T':

inFile >> (\*T);

(\*I) = new int[(\*T)+1];

break;

case 'i':

inFile >> (\*I)[0];

break;

case 'K':

inFile >> (\*K);

break;

case 'l':

inFile >> (\*l);

break;

case 'M':

inFile >> (\*M1) >> (\*M2);

break;

case 'S':

inFile >> (\*S);

break;

case 'H':

inFile >> (\*H);

break;

case 'C':

inFile >> (\*C);

break;

case 'Q':

inFile >> (\*Q1) >> (\*Q2);

break;

case 'L':

inFile >> (\*L);

break;

case 'A':

inFile >> (\*A);

break;

case 'R':

inFile >> (\*R);

break;

}

}

inFile.close();

}

void GetProbability(int l, int K , double \*&p)

{

int i, sum=1;

double total=0.;

p = new double [K+1];

p[0] = exp(-l); //算出p陣列的值

for(i=1; i<=K; i++){

sum \*= i;

p[i] = exp(-l)\*pow(l,i)/sum;

total += p[i];}

p[K]=1-total-p[0];

}

int GetDemand(int K, double \*&p, int T, double val)

{

int i;

double \*cp, tmp=0;

cp = new double[K+2];

for(i=0; i<=K+1; i++){ //將cp陣列歸0

cp[i]=0.;}

for(i=0; i<=K; i++){ //算出cp陣列的值

tmp += p[i];

cp[i+1] += tmp;}

cp[K+1]=1;

for(i=0; i<=K+1; i++){ //傳入隨機變數放入cp陣列中比較大小

if(cp[i]<=val && val<cp[i+1])

return i;}

}

void GetCost1(Policy \*Plc, int T, int K, int I, int\*D, int S, int H, int C, int L, int A)

{

int i, j, k, c, demandTmp, tmp;

for(c=0 ; c<4 ; c++){

if(c==0 || c==2){

for(i=1 ; i<=T ; i++){

for(k=1 ; k<=L ; k++){ //若有訂貨,設置tmp為1做下面的判斷式用

if(Plc[c].O[i-k]==Plc[c].Q)

tmp=1;

}

demandTmp=0; //暫存第t~t+L-1的需求

for(j=0 ; j<L ; j++)

demandTmp+=D[i+j];

if((Plc[c].I[i-1]-demandTmp)<A && tmp!=1)

Plc[c].O[i]=Plc[c].Q;

else

Plc[c].O[i]=0;

Plc[c].I[i]=Plc[c].I[i-1]+Plc[c].O[i-L]-D[i];

tmp=0; //歸零

if(Plc[c].I[i]>0 && Plc[c].O[i]>0)

Plc[c].DC[i]=H\*Plc[c].I[i]+C+Plc[c].M\*Plc[c].O[i];

else if(Plc[c].I[i]>0 && Plc[c].O[i]==0)

Plc[c].DC[i]=H\*Plc[c].I[i];

else if(Plc[c].I[i]<=0 && Plc[c].O[i]>0)

Plc[c].DC[i]=S\*(-Plc[c].I[i])+C+Plc[c].M\*Plc[c].O[i];

else if(Plc[c].I[i]<=0 && Plc[c].O[i]==0)

Plc[c].DC[i]=S\*(-Plc[c].I[i]);

Plc[c].TCost+=Plc[c].DC[i];

}

}

}

}

void GetCost2(Policy \*Plc, int T, int K, int I, int\*D, int S, int H, int C, int L, int R)

{

int i, j, k, c, tmp;

for(c=0 ; c<4 ; c++){

if(c==1 || c==3){

for(i=1 ; i<=T ; i++){

for(k=1 ; k<=L ; k++){ //若有訂貨,設置tmp為1做下面的判斷式用

if(Plc[c].O[i-k]==Plc[c].Q)

tmp=1;

}

if(Plc[c].I[i-1]<R && tmp!=1)

Plc[c].O[i]=Plc[c].Q;

else

Plc[c].O[i]=0;

Plc[c].I[i]=Plc[c].I[i-1]+Plc[c].O[i-L]-D[i];

tmp=0; //歸零

if(Plc[c].I[i]>0 && Plc[c].O[i]>0)

Plc[c].DC[i]=H\*Plc[c].I[i]+C+Plc[c].M\*Plc[c].O[i];

else if(Plc[c].I[i]>0 && Plc[c].O[i]==0)

Plc[c].DC[i]=H\*Plc[c].I[i];

else if(Plc[c].I[i]<=0 && Plc[c].O[i]>0)

Plc[c].DC[i]=S\*(-Plc[c].I[i])+C+Plc[c].M\*Plc[c].O[i];

else if(Plc[c].I[i]<=0 && Plc[c].O[i]==0)

Plc[c].DC[i]=S\*(-Plc[c].I[i]);

Plc[c].TCost+=Plc[c].DC[i];

}

}

}

}

void Print\_Policy(Policy \*Plc, int T, int \*D, int min)

{

int i;

cout << "Detailed info for best strategy is P[" << min << "]:" << endl;

cout << setw(8) << 't' << setw(8) << 'D' << setw(8) << 'O' << setw(8) << 'I' << setw(8) << "DC" << endl;

cout << "--------------------------------------------" << endl;

for(i=0 ; i<=T ; i++)

cout << setw(8) << i << setw(8) << D[i] << setw(8) << Plc[min].O[i] << setw(8) << Plc[min].I[i] << setw(8) << Plc[min].DC[i] << endl;

}

