Teste din Curs BPC, grile

Bazele programării calculatoarelor Basics of Programming (Academia de Studii Economice din Bucure□ti)

CAPITOLUL 1

1. Caracteristicile oricărui algoritm sunt: 1. Generalitate; 2. Complementaritate; 3. Claritate; 4. Finitudine; 5. Recursivitate; 6. Iterativitate.

a) toate;
b) b) 1,3,4,5 şi 6;
c) c) 1,2,3 şi 4;
d) d) 1,3 şi 4;
e) e) 1,2,5 şi 6

2. Un algoritm recursiv este:

a) un algoritm care se autoapelează;
b) un proces repetitiv static;
c) un proces repetitiv dinamic;
d) un proces repetitiv prin care valoarea unei variabile se determină pe baza a cel puţin uneia dintre valorile ei anterioare;
e) un proces alternativ prin care valoarea unei variabile se determină pe baza a cel puţin uneia dintre valorile ei anterioare.

3. Blocurile dintr-o subschemă logică sunt etichetate cu una din informaţiile: 1)START;2)citire; 3)scriere; 4)expresie aritmetică; 5)expresie logică; 6)expresie relaţională; 7)sir de caractere; 8)atribuire; 9)salt necondiţionat; 10)STOP.

a)oricare;
b)1,2,3,5,6,8 sau 10;
c)1,2,3,4,8 sau 10;
d)1,2,3,5,6,7,8 sau 10;
e)1,2,3,4,6,8,9 sau 10

4. Reprezentarea prin arbori este permisă numai pentru structurile: 1)BLOCK; 2)IF-THEN- ELSE; 3)CASE-OF; 4)WHILE-DO; 5)DO-UNTIL; 6)DO-FOR.

a) toate;
b)1,2,3,4 şi 5;
c)2,3,4,5 şi 6;
d)1,2 şi 4;
e)1,2 şi 5.

5. Structura DO-FOR(v,vi,vf,vr,s) este echivalentă cu:

a) BLOCK(v=vi, DO- UNTIL(BLOCK(v=v+vr,s),v>vf));
b) BLOCK(v=vf, DO-UNTIL(BLOCK(s,v=v- vr),v≤vi));
c) BLOCK(v=vi, IF-THEN(v≤vf, DO-UNTIL(BLOCK(s,v=v+vr),v>vf)));
d) BLOCK(v=vf, WHILE-DO(v>vi,BLOCK(s,v=v-vr)));
e) BLOCK(v=vi, WHILE- DO(v<vf,BLOCK (s,v=v+vr)));

6. Structura WHILE-DO(c,s) este echivalentă cu:

a) DO-UNTIL(s,*c*);
b) BLOCK(s,DO- UNTIL(s,*c*));
c) IF-THEN(c,DO-UNTIL(s,c));
d) BLOCK(s,IF-THEN(c,s));
e) DO- UNTIL(IF-THEN(c,s),*c*)

7. Un algoritm structurat este echivalent cu un algoritm pus sub una din formele:
1)BLOCK(s1,s2);
2)IF-THEN-ELSE(c,s1,s2);
3)IF-THEN(c,s);
4)CASE-OF(i,s1,s2,...,sn,s);
5)WHILE-DO(c,s);
6)DO-UNTIL(s,c);
7)DO-FOR(v,vi,vf,vr,s).
a)1,2,3,4,5,6,7;
b) 1,2,3,5,6;
c) 1,2,5,6,7;
d) 1,2,5;
e) 1,2,6

8. Teorema de structură stabileşte că:
a) orice schemă logică este echivalentă cu o schemă logică structurată;
b) orice schemă logică poate fi pusă sub una din formele:BLOCK(s1,s2); IF-THEN-ELSE(c,s1,s2); WHILE-DO(c,s);
c) corectitudinea unei scheme logice structurate se verifică prin examinarea fiecărui nod din arborescenţa sa;
d) o schemă logică structurată poate fi descompusă în structurile privilegiate

9. Care sunt tipurile de proiectări cristalizate în domeniul programării structurate?
-proiectarea top-down
-proiectarea modulara
-proiectarea structurata

CAPITOLUL 2

1. O dată reprezentată VF algebrică pe 2o are valoarea maximă:
   a) 2^{16};
   b) 2^{16}-1;
   c) 2^{15}-1;
   d) 2^{15};
   e) 2^{16}+1.

2. Numărul în zecimal a cărui reprezentare internă în VF algebrică este **10001111** este:
   a) 143;
   b) –15;
   c) –143;
   d) –113;
   e) 113.

3. Operaţia de scriere desemnează:
   a) afişarea datelor pe monitor;
   b) scrierea datelor pe suporti magnetici
   c) transferul datelor între zone de memorie principală;
   d) transferul datelor din memoria principală pe suporţi externi;
   e) transferul datelor în buffer.

4. Structura de date se defineşte ca:
   a) o colecţie de date pe care s-a definit un mecanism de selectare a componentelor;
   b) o colecţie de date la care o componentă este independentă de celelalte;
   c) o colecţie de date compusă din subcolecţii de acelaşi tip;
   d) o colecţie de date compusă din subcolecţii de tipuri diferite;
   e) o colecţie recursivă de date.

5. Masivul este o structură:
   a) recursivă;
   b) omogenă cu acces secvenţial;
   c) omogenă cu acces direct;
   d) eterogenă cu acces secvenţial;
   e) eterogenă cu acces direct.

6. Articolul este o structură:
   a) dinamică;
   b) omogenă cu acces secvenţial;
   c) omogenă cu acces direct;
   d) eterogenă cu acces secvenţial;
   e) eterogenă cu acces direct.

7. Stiva este o listă la care:
   a) inserarea şi ştergerea se fac la capul listei şi citirea se face la baza listei;
   b) inserarea, ştergerea şi citirea se fac la capul listei;
   c) inserarea, ştergerea şi citirea se fac la baza listei;
   d) inserarea se face la capul listei, iar ştergerea şi citirea se fac la baza listei;
   e) inserarea şi ştergerea se fac la baza listei şi citirea se face la capul listei.

CAPITOLUL 3

1. Într-o abordare schematică, enumeraţi etapele de realizare a sistemelor informatice.
   studiul şi analiza sistemului informaţional actual;
   proiectarea de ansamblu;
   proiectarea de detaliu;
   elaborarea programelor;
   implementarea şi exploatarea sistemului.

2. 2. Fazele dezvoltării programelor sunt:
1) editare; 2) verificare sintaxă; 3) compilare; 4) editare legături; 5) lansare în execuţie; 6) testare.
a) toate;
b) 1,2,3,4 şi 5;
c) 1,3,4,5 şi 6;
d)1,2,3 şi 4;
e) 1,3,4 şi 5.

CAPITOLUL 4

1. Specificaţi cum va arăta secvenţa de cod următoare, după preprocesare:

```
#define N 10
#define M 10
#define MAX  (M+N)
#define DIM(a,b) (a)*(b)
char v[N],v1[10+DIM(5+M,6)];
char v1[10*MAX];
char m[M][N];
```

După preprocesare, secvenţa de cod va deveni:

```
char v[10],v1[10+(5+10)*(6)];
char v1[10*(10+10)];
char m[10][10];
```

2. Se presupune un articol cu următoarea structură:

Cod magazin	**Vânzări lunare**			
	Luna 1	**Luna 2**	**…**	**Luna 12**
întreg	real	real	**…**	real
2	4	4	**…**	4

Scrieţi modul de declarare a articolului şi lungimea sa în număr de octeţi. Exemplificaţi referirea câmpurilor.

Articolul se declară astfel:

```
struct magazin {int cod_magazin;
                float vanzari_lunare[12];
  }artic
    ol;
```

Articolul are 50 de octeţi, iar referirea câmpurilor se realizează astfel:
articol.cod_magazin; articol.vanzari_lunare[i], cu i=0,1,…,11.

```
#include <stdio.h>
void main()
{ struct magazin {
          int cod_magazin;
          float vanzari_lunare[12];          };
//Initializarea articolului;
```

```
struct magazin gigel_srl={200, 1,2,3,4,5,6,7,8,9,10,11,12};
//sau cu evidentierea structurii de masiv:
struct magazin gigel_srl1={200, {1,2,3,4,5,6,7,8,9,10,11,12}};
printf("\n%6.2f",gigel_srl1.vanzari_lunare[10]);
```

3. Se presupune un articol cu următoarea structură:

nume	data naşterii			an de studiu	forma de învăţământ					
					zi		id			
					bursa	valoare	loc de	data angajării		
char[40]	zi	lun	an	int	char	float	char[30]	zi	lun	an

Specificaţi cum se realizează declararea şi iniţializarea câmpurilor unui student la zi pentru structura articolului prezentat.

Declararea şi iniţializarea câmpurilor unui student la zi pentru structura prezentată se realizează astfel:

```
#include <stdio.h>
void main()
{ //Declararea articolului cu structura variabila:
struct articol { char nume[40];
                 struct { int zi, luna, an;} datan;
                 int an_st; char forma_inv;
                 union { struct {char bursa; float valoare;}zi;
                         struct {char
                                 loc_m[30];
                                 struct  {int zi, luna, an;}data_ang;
                                 }id; } parte_vb;
                                 };
//Initializarea campurilor unui student la zi:
struct articol a={"Popescu Felix",{4,1,1974},1,'Z',{'D',250.5}};
printf("\nData nasterii: %i.%i.%i, Forma de inv.: %c, Val. bursa: %6.2f",
a.datan.zi, a.datan.luna, a.datan.an, a.forma_inv, a.parte_vb.zi.valoare); }
```

4. Specificaţi care va fi valoarea variabilei c.

```
int a=7,b=9,c;
c=(a>b)?a:b;
```

Variabila *c* primeşte valoarea maximă dintre *a* şi *b*, anume 9

5. Scrieţi secvenţele echivalente pentru următoarele exemple:

```
y=x++;
y=--x;
```

y=x++; este echivalent cu secvenţa y=x; x=x+1;
y=--x; este echivalent cu secvenţa x=x-1; y=x;

6. Care din următoarele secvenţe **nu** realizează suma a **n** elemente ale unui vector:
a) s=0; for(i=0; i<n; i++) s+=x[i];
b) s=0; for(i=n-1; i>=0; i--) s+=x[i];
c) s=0; i=0; while (i<n) {s+=x[i];i++;};
d) s=0; i=n-1; while (i>0) {s+=x[i]; i--;} ;
e) s=0; i=0; do { s+=x[i]; i++; } while(i<n);

7. Secvenţa: for(i=0; i<n-1; i++) {z=x[i]; p=i; for(j=i+1; j<n; j++) if(x[j]<z) {z=x[j]; p=j; } a=x[i]; x[i]=z; x[p]=a; }realizează:
a) minimul dintr-un vector cu reţinerea poziţiei primei apariţii;
b) minimul dintr-un vector cu reţinerea poziţiei ultimei apariţii;
c) sortarea unui vector prin metoda bulelor;
d) sortarea unui vector prin metoda selecţiei;
e) căutarea unei valori date într-un vector.

8. Triunghiul de sub diagonala secundară (inclusiv diagonala) unei matrice pătrate se poate parcurge numai cu secvenţele:
1. for(i=0; i<n; i++) for(j=n-i-1; j<n; j++) ...;
2. for(i=0; i<n; i++) for(j=n-1; j>=n-i-1; j--) ...;
3. for(i=n-1; i>=0; i--) for(j=n-i-1; j<n; j++) ...;
4. for(i=n-1; i>=0; i--) for(j=n-1; j>=n-i-1; j--) ...;
5. for(j=0; j<n; j++) for(i=n-j-1; i<n; i++) ...;
6. for(j=0; j<n; j++) for(i=n-1; i>=n-j-1; i--) ...;
7. for(j=n-1; j>=0; j--) for(i=n-j-1; i<n; i++) ...;
8. for(j=n-1; j>=0; j--) for(i=n-1; i>=n-j-1; i--) do ....
a) 1,2,5 şi 6;
b) b) 3,4,7 şi 8;
c) c) 1,2,3 şi 4;
d) d) 5,6,7 şi 8;
e) e) toate.

9. Următoarele secvenţe descriu algoritmi recursivi:
1) s=0; for(i=n-1; i>=0; i--) s+=x[i];
2) for(i=0; i<n; i++) y[i]=x[i];
3) nr=0; i=0; while(i<n) {if(x[i]>0) nr+=1; i++;};
4) for(i=0; i<n; i++) z[i]=x[i]*y[i];
5) i=0; z=0; do {z+=x[i]*y[i]; i++;} while(i<n);
6) s=1; fori=0; i<n; i++) s*=i;
a) toate
b) 1,3,5 si 6
c) 2,4 si 6
d) 3 si 5
e) niciunul