Seturi de teme

SET1

1. Scrieți un program care afişează un mesaj.
2. Scrieți o funcție care întoarce argumentul la puterea a treia cu prototipul
   int pw3 (int num).
   X= pw3 (y) <-> $x = y^3$.
3. Scrieți o funcție ce calculează $x = y^4$ folosind funcția sqrt(double x)
4. Scrieți un program care citeşte de la tastatură un întreg şi-l afişează la puterea a treia prin folosirea pw3.
5. Scrieți în pseudocod un program care elimină comentariile dintr-o sursă C.
6. Eliminați erorile din următoarea secvență:

```
main (x)
    {
    scanf ( ” Câți gândaci sunt ?“ , nr_gândaci );
    printf ( “ Acest program are  %d probleme “, nr_gândaci )ş
    }
```

SET2

1. Atunci când se afişează o variabilă de tip float sau double folosind specificatorul de format %f câte cifre zecimale sunt afişate ?
   printf() realizează rotunjirea sau trunchierea valorii afişate.
2. Scrieți un program ce conține următoarele declarații:

```
char c;
int j;
float f;
```

   şi afişează adresa fiecărei variabile. Vorbiți despre modul în care compilatorul alocă memorie pentru variabile.
3. Scrieți echivalentul octal, hexa, zecimal pentru următoarele numere binare:
   1. 0001.0010
   2. 0110.0101
   3. 0110.1011
   4. 1011.1011
   5. 0011.1111
   6. 0101.1111
   7. 0000.0010.0110.0100

   în ambele cazuri când se consideră că numerele sunt date în
   a) complement față de 1
   b) complement față de 2
4. Scrieți declarațiile pentru următoarele formate:
   a) Întreg lung fără semn
   b) Variabilă în virgulă mobilă cu dublă precizie
   c) Pointer la caracter
   d) Caracter inițializat cu ‘x’
   e) Funcție externă ce întoarce un întreg fără semn

5. Scrieți următoarele numere în binar în complement față de 1 sau în complement față de 2:
   1, -1, 255, 256, 511, 512, 513, 128, -128, 0xFF, 0x7F.

SET3

1. Fiecare calculator este limitat în precizia maximă care o poate folosi în reprezentarea numerelor în virgulă mobilă. Din acest motiv pentru un epsilon suficient de mic următoarea expresie va fi adevărată.

$$1.0 = 1.0 + eps$$

Să se scrie un program care găseşte cea mai mare valoare pentru eps pe computerul pe care e executat.
Obs.: această valoare diferă între float şi double. Se va folosi 1.0 pentru teste şi nu 0.0 pentru că majoritatea calculatoarelor nu au instrumente de lucru specifice pentru manipularea aritmeticii lui 0.

2. Să se rescrie următorul program fără a se folosi instrucțiunile break, continue şi go to.

```
/* programul numără caracterul 'a' din intrare
#include <stdio.h>
#include <ctype.h>
void main (void)
{
        int nr_a=0;
        char c;
        c=getch();
        while(1)
        {
                if(c== '\n') break
                if (isdigit(c)) continue
                if (c=='a') goto add_a;
                get_ch: c=getchar();
                goto end;
                add_a: nr_a++;
                goto get_ch;
                end: ;
        }
}
```

3. Scrieți o funcție care acceptă un întreg la intrare şi scrie un număr de spații egal cu pas de intrare. Folosind aceeaşi funcție să se scrie un program care citeşte caractere de la intrarea standard, şi le afişează dar înlocuieşte taburile cu 5 spații.
4. Multe programe care necesită transfer de caractere dintr-un loc în altul folosesc un mecanism de checksum (sumă de control) pt. a verifica corectitudinea transferului. Aceasta se realizează prin sumarea valorilor tuturor caracterelor transmise şi trimiterea ei împreună cu şirul respectiv. Dacă suma se potriveşte, există o oarecare probabilitate ca ele să fi fost transferate corect. Scrieți un program care să folosească o funcție

unsigned int CS (char *s)

care să calculeze checksum-ul pentru o linie introdusă de la tastatură.

5. Scrieți un program care clasifică caracterele citite într-una din următoarele categorii:
   white _Space (space, '\n', '\r', '\t')
   punctuator (",!;:()?)
   alfabet (a-z , A-Z)
   numeric (0-9)
   uncknown (oricare alt caracter)
   Deci la orice tastă atinsă, să apară pe ecran codul tastei şi clasa din care face parte.
   5.1. Programul să fie rezolvat numai cu if, else şi return
   5.2. Programul să fie rezolvat numai cu switch şi return.
6. Scrieți un program care afişează literele mici şi mari precum şi valorile lor zecimale.

SET 4

1. Introduceți paranteze în expresiile de mai jos a.î.să fie specificat (cu ajutorul lor) explicit modul de evaluare implicit folosit de compilator.(Se va folosi tabela de precedență)
   a) a=b*c==2
   b) a=f(x)&&a>100
   c) a==b&&x!=y
   d) a=b+=2+f(2)
   e) a=b>>2+4
   f) a=s.f+y.x
   g) a=b&&a>z?x=y:z
   h) a=*++p*p
   i) a=b^c&d

2. Să se construiasca o expresie ce va genera un long int (4 octeți) prin apelul repetat al funcției getbt() care întoarce un singur octet. Ordinea de aranjare este ca primul apel al lui getbt() va genera octetul cel mai semnificativ din întreg (nr=Expresie (getbt()).

3. Care este ieşirea următorului program ?

```
  void main(void)
{
 short i=0;
printf ("y.d\n",(i+1)*(1=1));
 }
```

4. Să se scrie o funcție unsigned long int circular_shift (unsigned long int a,int n); care face deplasare stanga cu recirculare , cu n poziții a numărului a.

   a=0001.0110.0011.1010.0111.0010.1110.0101

aplicănd circular_shift (a,5) spre stănga =>
1100.0111.0100.1110.0101.1100.1010.0010

5. Folosind <<sau>> să se determine care este cel mai mare întreg care poate fi reprezentat pe calculator.
6. Scrieți o funcție care primeşte un întreg şi-l afişează în binar (folosiți sizeof (int) pentru a afla nr de biți.
7. Scrieți o funcție care citeşte un nr în forma binară şi-l converteşte în formă hexa.
8. Scrieți o funcție pack care compune un long int din 4 caractere de intrare long int pack (char a,char b, char c, char d)

Bublle sort

```
# define FALSE 0
#define TRUE 1
#include <stdio.h>
void bubble_sort (int*v,int n)
{
    int j,k,temp,sorted=FALSE;
    while (!sorted)
    {
            sorted=TRUE  //sc pp vect sortat
            for (j=0;j<n-1;j++)
            {
                    if (v[j]>v[j+1])
                    {
                            // cel puțin un element nu este în ordine
                            sorted =FALSE;
                            temp=v[j];
                            v[j]=v[j+1];
                            v[j+1]=temp;
                    }
            }
        }
  }

int strlen1(char *s)
{
    int i=0;
    while (s[i])++i;
    return i;
}//întoarce nr elemente dintr-un nr mai puțin '\0'
```

```
void strcopy1(char *s1, char *s2)
{
    int i;
    for (i=0;*(s1+1];++I)*(s2+i)=*(s1+I);
    s2[++I]='0';
}  //classical
```

sau

```
void strcopy1(char *s, char *s2)
{
    while (*s2++=*s1++);
}    //professional
```

echivalentul funcției pas din Pascal , str str un contor un nr într-altul ….

```
Int strstr1(char *s1,char *s2)
{
    char *p,*q, *subs;
    for (subs=s1;*subs; subs++)
    {
            p=subs;
            q=s2;
            while (*q)
            if(!(*q++!=*p++)) return subs –s1;
            else break;
    }
    return( –1);
} //de verificat
```

Set 5

1. Să se explice ce va rezulta în urma evaluării următoarelor expresii:
   Fie static int ar []={10,15,4,25,3,-4}
   int *p;
   p=&ar [2];
   a) *(p+1)
   b) p[-1]
   c) (ar-p)
   d) ar [*p++]
   e) *(ar+ar[2])

2. Ce este greşit în următoarul caz:

```
Int j , v[5]={1,2,3,4,5}
……
for (j=1; j<5; ++j)
printf("x.d\n",v[j])
```

3. Modificați bubble sort a.î. în loc să rearanjeze elementele într-un tablou el să stocheze numai ordinea corectă de aranjare.

Deci v[]={$\overset{0}{13}, \overset{1}{25}, \overset{2}{11}, \overset{3}{2}, \overset{4}{14}$} în urma aplicării bubble sort modificarea se realizează în tabloul

ord[] = {1,4,0,2,3}

Se observă :

V[ord[0]]=V[1]=25
V[ord[1]]=V[4]=14
V[ord[2]]=V[0]=13

.
.

4. Scrieți o funcție
   void sort_concat(double *a, double *b, double *c)
   care ia a,b două tablouri sortate şi le depune în c care trebuie să fie şi el sortat deasemenea.
5. Modificați funcția de mai sus astfel încât să elimine valorile egale.
6. Scrieți o funcție
   char * plussir (char *a, char *b);

care concatenează pe a cu b.

7. Declarațiile
   char S[10]
   şi
   char *S
   sunt la fel dacă nu arătați diferențele scriind un program în care ele nu pot fi înlocuite.
8. Scieți o funcție care sortează un tablou de şiruri de caractere în ordine alfabetică.
   ex. : S[][] = {{mike},{oana},{lili},{dana}}
   în urma apelului funcției S[][] = {{dana},{lili},{oana},{mike}}.
9. Fiind date următoarele declarații :
   static int a[2][3]={{-3,14,5},{1,-10,8}};
   static int *b[]={a[0],a[1]};
   int *p=b[1];
   Cum vor fi evaluate urm[toarele expresii :
   a) *b[1]
   b) *(++p)
   c) *9*(--p-2)
10. Care din următoarele expresii este echvalentă cu a[j] ? :
    1. *(a[j]+k)
    2. **(a[j+k])
    3. (*a+j))[k]
    4. (*(a+k))[j]
    5. *((*a+j))+k)
    6. **(a+j)+k
    7. *(&a[0][0]+j+k)

Set 6

1. Scrieți o funcție care afişează pe ecran din 20 în 20 de apeluri , de câte ori a fost apelata în total.
2. Să se scrie 2 funcții :

- conversie integer -> BCD
- conversie BCD -> integer

folosindu-se câmpurile de biți (bit field).

3. O stivă este o listă mai particulară în sensul că :
   - adăugarea se poate face doar la capătul listei (operație numită push)
   - ştergerea mai tot din coada listei (operație numită pop)

   Scrieți un program care :
   - crează o listă de acest tip folosind cele 2 funcții
     push pentru adăugare
     pop pentru ştergere.

   Trebuiesc implementate şi aceste funcții.

Set7

1. Să se scrie un program care să primească în linia de comandă o frază precum şi switch-urile –c sau –C funcție de acestea, programul vă oferă fraza din linia de intrare cu litere mici sau mari. Dacă switch nu este precizat se va considera implicit –c adică afişarea cu litere mici.
2. Scrieți o versiune recursivă a funcției strlen. Este recursivitatea mai bună ca versiunea normală. Argumentați.
3. Scrieți o funcție recursivă care calculează cel mai mare divizor comun a două nr. pozitive întregi.
4. Scrieți şi o versiune iterativă pentru punctul 3.
5. Scrieți o funcție recursivă care acceptă un pointer la string ca argument, transformă respectivul şir într-o listă înlăanțuită de caractere şi întoarce pointerul către primul element din listă.
6. Scrieți o funcție recursivă care numără elementele dintr-o listă înlănțuită. Argumentul trebuie să fie pointerul către primul element din listă. Evident că valoarea returnată va fi întreagă.
7. Scrieți o funcție recursivă care afişează valorile fiecărui element dintr-o listă înlănțuită.
8. Verificați legalitatea următoarelor declarații:
   a. *(*( )) [ ]
   b. * (**) [ ]
   c. (*(*(*x( )) [ ]) ( ))
   d. **x[ ] ( )
   e. * (x[ ]) [ ]
   f. * (* (x ( )) ( ))

Set8

1. Explicați efectele (şi traducerea) următoarelor macrouri:

1.1.
```
#define BS 1024
int buf [BS+1]
```

1.2. #define a[b] b+1
…
a(1) + 1

1.3. # define a(b) b+1
…
a(1) +1

1.4. #define cos(x) *cos(x)
…
cos(x) + cos (cos(y) + 1)

1.5. #define min(x,y) ((x) >= (y) ? x:y)
min (1,min (a,b))

Set9

1. Scrieți un program care să implementeze comanada
   # include “nume_fişier”
2. Scrieți o funcție care elimină toate spațiile suplimentare dintr-o sursă C
3. Scrieți un program care să verifice inchiderea corectă a parantezelor într-o sursă C. Indicație: stiva.
4. Scrieți un program care numără caracterele, cuvintele şi liniile unui fişier.
5. Scrieți un program care primeşte ca argumente mai multe fişiere şi le afişează pe ecran (un fel de type cu mai multe inturi)