Programarea calculatoarelor

Curs 2

Continut curs

- Constante
- Atribuire
- Tipul enumerare
- Decizie
- Ciclu
- Operatori
- Precedenta operatorilor
- Operatori
  - Relationali
  - Aritmetici
  - Pe biti
  - Unari
- Conversii de tip

Constante

- constante simbolice

#define NUME TEXT_CE_INLOCUIESTE_NUMELE

- se inlocuieste ad-literam NUME in program cu textul de dupa

#define TRUE 1

#define FALSE 0

- conventie – litere mari

Constante

- 12, 0; asimilate int
- 12345678900, 12L – asimilate long
- ‘x’, ‘\n’– char
- “sir de caractere”,”” – char* (string)
- 123u – unsigned
- 12.5, 1.5e-2 – double
- 12.4F - float

Declaratii

- toate variabilele trebuie declarate inainte de utilizare
- declaratia contine tipul si una sau mai multe variabile de acelasi tip ex: int x,y,z;
- se poate efectua initializare la declarare
  - ex: char sir[30]="un sir"; int x=4;
- const poate specifica faptul ca o variabila nu va fi schimbata
  - ex: const x=5;
- o variabila neinitializata va avea o valoare aleatoare

Expresie

- Expresie primara = variabila sau constanta sau sir de caractere sau (expresie)
- ex:
  - x sau 5 sau “sir de caractere” sau (x+5)
- expresie = expresii primare carora li se aplica operatori
- ex:
  - x+5
  - 5%2
  - (x+5)-(x-5)
- http://www.csci.csusb.edu/dick/samples/c.syntax.html#string_literal

Atribuire

- variabila = expresie;
- se evalueaza expresia si valoarea acesteia devine noua valoare a variabilei
- x= y; //atribuie variabilei x valoarea lui y
- Varianta din Pascal x:=y;

```c
#include <stdio.h>

int main()
{
        int x=5;
        printf("x= %d; adresa lui x=%p\n",x,&x);
        x=x+1;
        printf("x= %d; adresa lui x=%p\n",x,&x);
        return 0;


}
```

```
vlad@hoth:~/programare/c2$ ./c2p1
x= 5; adresa lui x=0x7fff667651fc
x= 6; adresa lui x=0x7fff667651fc
```

Tipul enumerare

- defineste un set de constante numite de tip intreg
- sintaxa: enum nume_tip {

identificator,

(sau) identificator=valoare

}

- ex:

enum bool {false,true};

enum bool a; //declara un element a de tip bool

Tipul enumerare exemple

```
enum bool{false, true};//false are valoarea 0, true are valoarea 1
enum bool a=false;
printf("a=%d\n",a);
a+=1;
printf("a=%d\n",a);
a=true;
printf("a=%d\n",a);
```

```
vlad@hoth:~/programare/c2$ ./c2p4
a=0
a=1
a=1
```

```
enum calificativ{ns=4,s=5,b=8,fb=10};
int notaStud1=ns,notaStud2=s, notaStud3=b, notaStud4=fb;

printf("nota Student1:%d\n",notaStud1);
printf("nota Student2:%d\n",notaStud2);
printf("nota Student3:%d\n",notaStud3);
printf("nota Student4:%d\n",notaStud4);
```

```
nota Student1:4
nota Student2:5
nota Student3:8
nota Student4:10
```

Instructiune

- ; (instructiunea vida)
- expresie;
  - x+5; y=x;
- apel de functie;
  - printf(“%d\n”,x);
- bloc de instructiuni
  - {}

Bloc de instructiuni

- bloc de instructiuni = mai multe instructiuni grupate intre { }
- bloc de instructiuni este echivalent cu o singura instructiune
- nu se pune ; dupa }
- exemplu:

```
int main()
{ //inceputul blocului de instructiuni
        printf(“hello world\n”);
        return 0;
} //sfarsitul blocului de instructiuni
```

Decizie

- if (expresie1)
  - instructiune1
- else
  - instructiune2
- else este optional

- instructiune1 se executa daca expresie1 este evaluata la o valoare diferita de 0
- instructiune2 se executa daca expresie1 este evaluata la 0

Decizie (2)

- A programmer is heading out to the grocery store, so his wife tells him "get a gallon of milk. If they have eggs, get a dozen." He returns with 12 gallons of milk.

Decizie (2)

- A programmer is heading out to the grocery store, so his wife tells him "get a gallon of milk. If they have eggs, get a dozen." He returns with 12 gallons of milk.

```
int main()
{
        int haveEggs;
        srand(time(NULL));
        haveEggs=rand()%2;
        if(haveEggs!=0)
                printf("i bought  12 gallons of milk\n");
        else
                printf("i bought 1 gallon of milk\n");
        return 0;

}
```

Decizie (3) – if … else if

- structura folosita cand se pot lua mai multe decizii diferite in functie de valoarea expresiei

```
if (delta<0)
        printf(“ecuatia nu are radacini reale”);
else if (delta==0)
        printf (“ecuatia are 1 radacina reala egala cu %f“, …);
else //cazul default – acopera toate celelalte p
        printf(“ecuatia are 2 radacini reale egale cu %f si %f”, …);
```

Operatori aritmetici

- +,-,*,/,%
- / => impartire intreaga (returneaza catul) daca operanzii sunt intregi
- / => impartire daca macar unul dintre operanzi este real
- % - restul impartirii
- / intreaga si % nu au rezultate bine definite in standard pentru operanzi negativi

Exemple operatori aritmetici

```
int x=5, y=2, z=-3;
printf("x/y (intreg)=%d, x%%y=%d, x/z(intreg)=%d, x%%z=%d\n", x/y,x%y,x/z,x%z);
printf("x/y=%f, x/z=%f\n", x*1.0/y,x*1.0/z);
```

```
x/y (intreg)=2, x%y=1, x/z(intreg)=-1, x%z=2
x/y=2.500000, x/z=-1.666667
```

Operatori relationali

- operatori relationali: >, <, >=,<=
- operatori egalitate: ==, !=
- observatie:
  - x==0 verifica daca x are valoarea 0 si intoarce 1 daca acest lucru se intampla
  - x=0 atribuie lui x valoarea 0 si intoarce aceasta valoare

```
printf("x=5: %d x==0:%d  x =0:%d x==0:%d\n",x=5,x==0,x=0,x==0);
```

```
x=5: 5 x==0:1  x =0:0 x==0:0
```

Operatori logici

- && - si
- || - sau
- ! – not (operator unar)

e1	e2	e1&&e2	e1||e2
0	0	0	0
0	1 (!=0)	0	1
1 (!=0)	0	0	1
1 (!=0)	1 (!=0)	1	1

- Observatie
  - **prin e1=1 intelegem e1 diferit de 0**

Operatorul ternar

- Conditie?valoare intoarsa in caz true: valoare rezultata in caz false
- c?x:y
- Ex:
  - a%2?”impar”:”par”
  - x>0?”strict pozitiv”:”negativ sau 0”
- echivalent cu

```
if(c)
    result=x;
else
    result=y ;
```

Exemplu operatori logici + operator ternar

- "Are you going to sit and type in front of that thing all day or are you going out with me?" -- programmer's girlfriend.
- "Yes" -- programmer

Exemplu operatori logici + operator ternar

- "Are you going to sit and type in front of that thing all day or are you going out with me?" -- programmer's girlfriend. "Yes" -- programmer

```c
int main()
{
        int play, goOut;
        srand(time(NULL));
        play=rand()%2;
        goOut=1-play;
        printf("go out=%s, play=%s, go out or play=%s\n",goOut==1?"yes":"no",
                        play==1?"yes":"no",(goOut||play)==1?"yes":"no");

        return 0;
}
```

```
[@web316:W,10:19 AM,Mon Oct 14]>gcc c2p6.c -o c2p6
[@web316:W,10:20 AM,Mon Oct 14]>./c2p6
go out=no, play=yes, go out or play=yes
[@web316:W,10:20 AM,Mon Oct 14]>./c2p6
go out=no, play=yes, go out or play=yes
[@web316:W,10:20 AM,Mon Oct 14]>./c2p6
go out=no, play=yes, go out or play=yes
[@web316:W,10:20 AM,Mon Oct 14]>./c2p6
go out=no, play=yes, go out or play=yes
[@web316:W,10:20 AM,Mon Oct 14]>./c2p6
go out=no, play=yes, go out or play=yes
[@web316:W,10:20 AM,Mon Oct 14]>./c2p6
go out=yes, play=no, go out or play=yes
```

Operatori pe biti

- pot fi aplicati doar pe variabilele de tip intreg(char, short, int, long)
- & - and pe biti
- | - sau pe biti
- ^ - xor
- << - shift stanga (inmultire cu 2)
- >> - shift dreapta (impartire intreaga la 2)

Operatori pe biti exemple

```
        unsigned char a=0, mask=0xff;

        printf("a&mask:%d\na|mask:%d\na^mask:%d\na<<1:%d\na>>1:%d\nmask<<1:%d\nm
ask>>1:%d\n",a&mask,a|mask,a^mask,a<<1,a>>1,mask<<1, mask>>1);
        return 0;
```

```
[@web316:W,10:59 AM,Mon Oct 14]>./c2p
a&mask:0
a|mask:255
a^mask:255
a<<1:0
a>>1:0
mask<<1:510
mask>>1:127
```

```
/* verifica daca al 4-lea bit dintr-un octet este 0 sau 1*/
int main()
{
        //mask va fi 00001000
        unsigned char mask=8, randomNumber;
        srand(time(NULL));
        randomNumber=rand()%255;
        printf("mask=%d, randomNumber=%d, al 4-lea bit este %s (%d)\n",mask,random
Number,((mask&randomNumber)==mask)?"1":"0",mask&randomNumber);
        return 0;

}
```

```
[@web316:W,11:27 AM,Mon Oct 14]>./a.out
mask=8, randomNumber=170, al 4-lea bit este 1 (8)
[@web316:W,11:27 AM,Mon Oct 14]>./a.out
mask=8, randomNumber=50, al 4-lea bit este 0 (0)
[@web316:W,11:27 AM,Mon Oct 14]>./a.out
mask=8, randomNumber=219, al 4-lea bit este 1 (8)
[@web316:W,11:27 AM,Mon Oct 14]>./a.out
mask=8, randomNumber=153, al 4-lea bit este 1 (8)
[@web316:W,11:27 AM,Mon Oct 14]>./a.out
mask=8, randomNumber=153, al 4-lea bit este 1 (8)
[@web316:W,11:27 AM,Mon Oct 14]>./a.out
mask=8, randomNumber=198, al 4-lea bit este 0 (0)
```

Operatori unari

- - => -x;
- incrementare/decrementare
  - i++; i--;
- i++ vs ++i
  - int a, b=3;
  - a=b++;
  - =>a=3;b=4

  int a, b=3

  a=++b;

  =>a=4, b=4
- ++i; // Fetch i, increment it, and return it
- i++; // Fetch i, copy it, increment i, return copy

Precedenta operatorilor

- tabelul din Kernighan&Ritchie p. 50

Operators	Associativity
`() [] -> .`	left to right
`! ~ ++ -- + - * (type) sizeof`	right to left
`* / %`	left to right
`+ -`	left to right
`<< >>`	left to right
`< <= > >=`	left to right
`== !=`	left to right
`&`	left to right
`^`	left to right
`|`	left to right
`&&`	left to right
`||`	left to right
`?:`	right to left
`= += -= *= /= %= &= ^= |= <<= >>=`	right to left
`,`	left to right

nary & +, -, and * have higher precedence than the binary forms.

a op b op c = (a op b) op c daca op are asociativitate stanga

a op b op c = a op (b op c) daca op are asociativitate dreapta

Instrucțiuni repetitive – for (;;)

- for([initializare];[conditie continuare];[actualizari]) instructiune
- initializare – expresie care se executa o singura data inainte de a se executa instructiunea pentru prima data
- conditie continuare – este evaluata **inainte** de fiecare iteratie
  - daca are valoarea true (diferit de 0) se executa instructiunea
  - daca lipseste conditia se considera ca este adevarata
  - daca are valoarea 0 se incheie iteratia, nu se mai executa instructiunea si se trece la instructiunea urmatoare
- instructiune – se executa daca se evalueaza conditie continuare la true
- actualizari – expresie care se executa dupa instructiune.

for - exemplu

- A programmer heads out to the store. His wife says "while you're out, get some milk.“ He never came home.

```
int main()
{
        int out=1;

        for(;out==1;)
        {
                printf("i'm still buying milk\n");
        }
        return 0;
}
~
```

```
i'm still buying milk
i'm still buying milk
i'm still buying milk
i'm still buying milk
i'm still buying milk
i'm still buying milk
i'm still buying milk
i'm still buying milk
i'm still buying milk
i'm still buying milk
i'm still buying milk
i'm still buying milk
i'm still buying milk
i'm still buying milk
i'm still buying milk
i'm still buying milk
i'm still buying milk
`Cvlad@hoth:~/programare/c2$
```

- important – sa ne asiguram ca se ajunge sa se iasa din for

Instructiuni repetitive while

- while (expresie) instructiune
- se evalueaza expresie
  - daca este false nu se executa instructiunea
  - daca este true
    - se executa instructiunea si se reia procesul

exemple while

- ex1:

int i=1;

while (i>5) printf(“%d”,i); //nu se executa niciodata printf

- ex2:

int i=3;

while (i>0){ printf(“%d\n”,i); i--;}

ce afiseaza?

instructiuni repetitive do… while

- do instructiune while (expresie);
- singurul tip de instructiune repetitiva dupa care se pune ;
- expresia este evaluata dupa ce se executa instructiunea (test final)

TEST 1

```
#include<stdio.h>
int main()
{
	int i=5;
	for(;scanf("%s", &i); printf("%d\n", i));
	return 0;
}
```

TEST 2

```
#include<stdio.h>
int main()
{
    int i=0;
    for(; i<=5; i++);
        printf("%d", i);
    return 0;
}
```