Facultatea de Inginerie Electrică

UNIVERSITATEA TEHNICĂ DIN CLUJ-NAPOCA

EUROPEAN UNIVERSITY OF TECHNOLOGY

Facultatea de Inginerie Electrica

PCLP 2

Programarea calculatoarelor si limbaje de programare 2

PCLP2

An I semestrul II

"Coding is easy when you C it in action."

CUPRINS

Responsabil curs:
Laura.Grindei@ethm.utcluj.ro

Responsabili laboratoare:
Claudia.Constantinescu@ethm.utcluj.ro
Angela.Lungu@ethm.utcluj.ro
Laszlo.Rapolti@ethm.utcluj.ro

Pagina web curs:
http://www.et.utcluj.ro/CursPCLP2.htm

Teams PCLP 2 Seria 1 2024
Cod: z1o9hou

Teams PCLP 2 Seria 2 2024
Cod: z1o9hou

LABORATOARE

Nu aveti teme la laborator , dar prezenta e obligatorie+ teste Teams Assignments

Laborator 1
Pointeri . Operatii cu pointeri

În acest capitol sunt prezentate consideraţii teoretice privind definirea şi utilizarea pointerilor fiind prezentate câteva probleme rezolvate cu pointeri la date de tip întreg, la tipul caracter şi la şir de caractere.

CONSIDERAŢII TEORETICE

Variabila de tip **pointer** este o variabilă care are ca şi valoare o adresă de memorie.

Pointerii se clasifică astfel:

- **Pointer la date** = conţine adresa unei variabile sau a unei constante
- **Pointer la funcţii** = conţine adresa codului executabil al unei funcţii
- **Pointer la obiecte** = conţine adresa unui obiect în memorie = adrese de date şi funcţii

Pointerii conţin adrese de memorie şi nu valori ca alte tipuri de variabile sau constante.

Dacă la o adresă de memorie se află altă adresă acest lucru are semnificaţia de indirectare (“pointare”) (o variabilă indică spre alta).

În Fig.1.1 este prezentat un exemplu de adrese de memorie în care sunt memorate date sau alte adrese ale altor date .

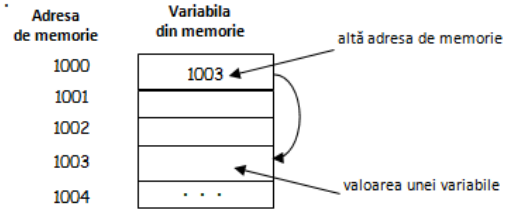


Fig.1.1. Reprezentarea variabilelor în memorie

PROBLEME REZOLVATE

Ex.1: Programul este un exemplu de declarare a 2 pointeri la date de tip int si respectiv real (float). Programul citeste de la tastatura o valoare intreaga x si o valoare de tip float corespunzatoare lui y si apoi afiseaza adresele variabilelor x si y si valorile lor si ale patratelor lor prin pointeri.

Varianta in C

```
#include <stdio.h>
#include <stdlib.h>
int main()
{int x, *p1;float y,*p2;
p1=&x; p2=&y;
printf("Introduceti o valoare intreaga x=");
scanf("%d", &x);
printf("Introduceti o valoare reala y="); scanf("%f", &y);
printf("\nAdresa lui x este:\n");
printf("p1=%p\n", p1);printf("\nValoarea lui x
este:\n");printf("x=*p=%d\n",*p1);
printf("\nValoarea lui x^2 este:\n");
printf("*p1^2=%d\n", *p1* *p1);
printf("\nAdresa lui y este:\n");
printf("p1=%p\n", p2);
printf("\nValoarea lui y este:\n");
printf("x=*p2=%.2f\n",*p2);
printf("\nValoarea lui y^2 este:\n");
printf("*p1^2=%.2f\n", *p2* *p2);
return 0;}
```

Varianta in C++

```
#include <iostream>
using namespace std;
int main()
{int x, *p1;float y,*p2;
p1=&x; p2=&y;
cout<<"Introduceti o valoare intreaga x=";
cin>>x ;
cout <<"\nIntroduceti o valoare reala y=";
cin>>y;
cout<<"\nAdresa lui x este:"<<p1;
cout<<"\nValoarea lui x este:"<<*p1;
cout<<"\nValoarea lui x^2 este:"<<*p1**p1;
cout<<"\nAdresa lui y este:"<<p2;
cout<<"\nValoarea lui y este:"<<*p2;
cout<<"\nValoarea lui y^2 este:"<<*p2* *p2;
return 0;}
```

Rezultate:

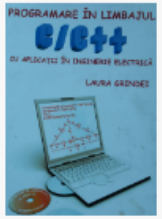
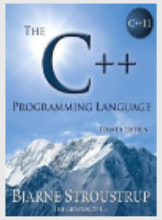
```
Introduceti o valoare intreaga x=2
Introduceti o valoare reala y=5.5

Adresa lui x este:
p1=0060FF04
Valoarea lui x este:
x=*p=2
Valoarea lui x^2 este:
*p1^2=4
Adresa lui y este:
p1=0060FF00
Valoarea lui y este:
x=*p2=5.50
Valoarea lui y^2 este:
*p1^2=30.25
```

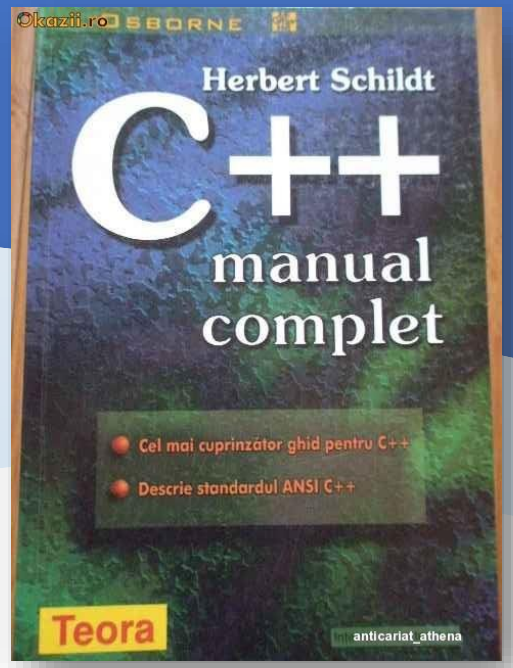
Aplicatie:

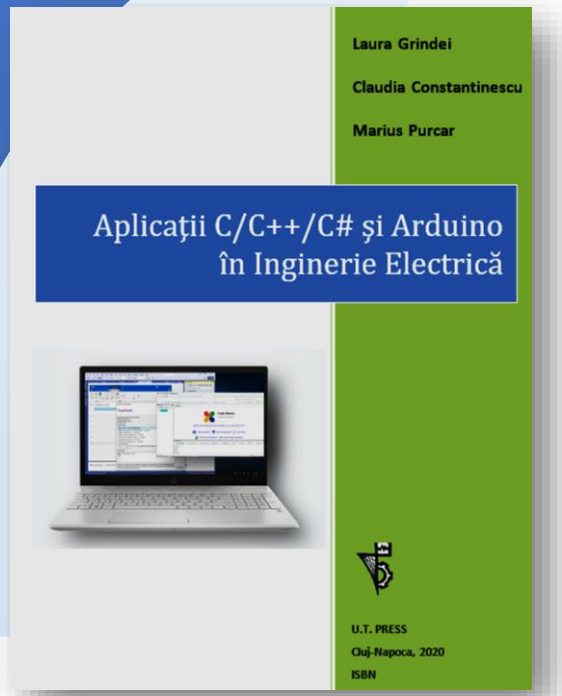
Să se modifice programul de mai sus astfel încât, utilizând pointeri, să se afişeze rezultatul calculului expresiei: x^2+3y^2 .

BIBLIOGRAFIE

Carte	Titlu, Autor, Editura, An publicare
	**Aplicatii C/C++/C# si Arduino in Inginerie Electrica** Laura Grindei, Claudia Constantinescu Marius Purcar Editura UTPress, On Line 2020
	**Programare in Limbajul C/C++ cu aplicatii in inginerie electrica** Laura Grindei, Casa Cartii de Stiinta, 2010
	**C++ Programming Language** Bjarne Stroustrup ADDISON WESLEY PUB CO INC 2013 download pdf
	**Aplicatii in limbajele C/C++ si Java** Mircea Vaida Casa Cartii de Stiinta 2002
	**Limbajele C si C++ pentru incepatori vol 1 - Liviu Negrescu** Mircea Vaida Casa Cartii de Stiinta 2002 download pdf
	**Programarea aplicatiilor folosind limbajul C# si platforma .NET** Daniela Alexandra Crisan Pro Universitaria 2015

BIBLIOGRAFIE

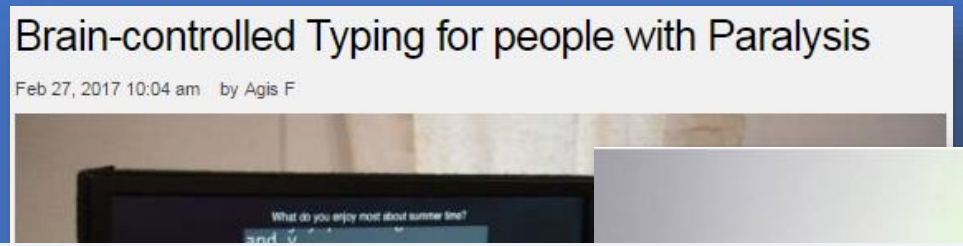


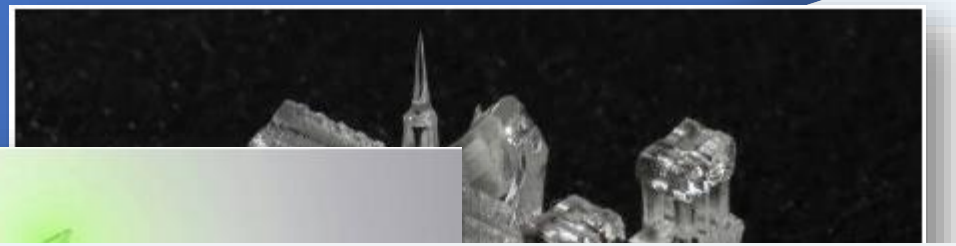


EXAMEN

Nota examen= (Nota Activitate laborator + Nota test teorie)/2	
Nota activitate laborator:	Media testelor din Assignments sau proiect laborator
Recuperari laborator:	**Prezenta la laborator este obligatorie** **Recuperari** : • **maxim 40%** in timpul semestrului, altfel rcontractare disciplina • **oricand in timpul semestrului** cu orice semigrupa
Nota test teorie:	Test grila in sesiune +program C/C++ +1p proiect IT prezentat la curs (optional)

PROIECT CURS

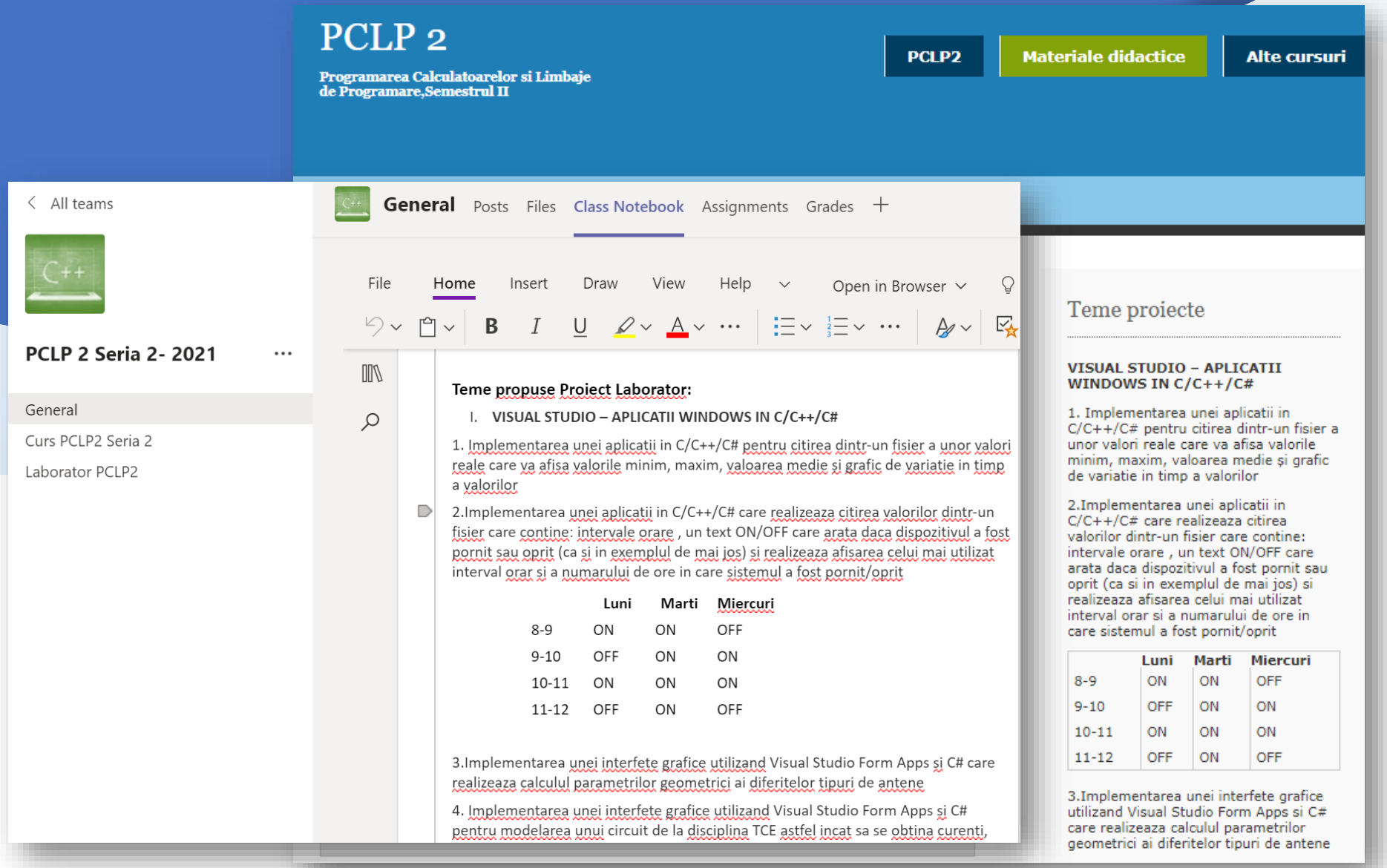




Top 10 Programming Languages

	Python	C	Java	C++	C#	R	JavaScript	PHP	Go	Swift
Paradigm	Multi-paradigm: object-oriented, imperative, functional, procedural, reflective	Imperative (procedural), structured	Multi-paradigm: object-oriented (class-based), structured, imperative, generic, reflective, concurrent	Multi-paradigm: procedural, functional, object-oriented, generic	Multi-paradigm: structured, imperative, object-oriented, event-driven, task-driven, functional, generic, reflective, concurrent	Multi-paradigm: array, object-oriented, imperative, functional, procedural, reflective	Multi-paradigm: object-oriented (prototype-based), imperative, functional, event-driven	Imperative, object-oriented, procedural, reflective	Compiled, concurrent, imperative, structured	Multi-paradigm: protocol-oriented, object-oriented, functional, imperative, block-structured
Designed by	Guido van Rossum	Dennis Ritchie	James Gosling	Bjarne Stroustrup	Microsoft	Ross Ihaka and Robert Gentleman	Brendan Eich	Rasmus Lerdorf	Robert Griesemer, Rob Pike, Ken Thompson	Chris Lattner and Apple Inc
Developer	Python Software Foundation	Dennis Ritchie & Bell Labs (creators), ANSI X3J11 (ANSI C), ISO/IEC	Sun Microsystems (now owned by Oracle corporation)	Bell Labs	Microsoft	R Core Team	Netscape Communications Corporation, Mozilla Foundation, Ecma International	The PHP Development Team, Zend Technologies	Google Inc.	Apple Inc
First appeared	20 February 1991 (26 years ago)	1972 (45 years ago)	May 23 1995 (22 years ago)	1983 (34 years ago)	2000 (17 years ago)	August 1993 (24 years ago)	December 4, 1995 (21 years ago)	June 8, 1995 (22 years ago)	November 10, 2009 (7 years ago)	June 2, 2014 (3 years ago)
Typing discipline	Duck, dynamic, strong	Static, weak, manifest, nominal	Static, strong, safe, nominative, manifest	Static, nominative, partially inferred	Static, dynamic, strong, safe, nominative, partially inferred	Dynamic	Dynamic, duck	Dynamic, weak, gradual (as for PHP 7.0.0)	Strong, static, inferred, structural	Static, strong, inferred
Platform	Cross-platform	Cross-platform	Windows, Solaris, Linux, OS X	Linux, MacOS, Solaris	Common Language Infrastructure	UNIX platforms, Windows, MacOS	Cross-platform	Unix-like, Windows	Linux, macOS, FreeBSD, NetBSD, OpenBSD, Windows, Plan 9, DragonFly BSD, Solaris	Darwin, Linux, FreeBSD
Filename extensions	.py, .pyc, .pyo (prior to 3.5), .pyw, .pyz (since 3.5)	.c, .h	.java, .class, .jar	.cc, .cpp, .C, c++, .h, .hh, .hpp, .hxx, .h++	.cs	.r, .R, .RData, .rds, .rda	.js	.php, .phtml, .php3, .php4, .php5, .php7, .phps	.go	.swift

PROIECT LABORATOR



De ce C/C++ ?

Conform statisticii TIOBE:

https://www.tiobe.com/tiobe-index

Feb 2023	Feb 2022	Change	Programming Language	Ratings	Change
1	1		Python	15.49%	+0.16%
2	2		C	15.39%	+1.31%
3	4	▲	C++	13.94%	+5.93%
4	3	▼	Java	13.21%	+1.07%
5	5		C#	6.38%	+1.01%
6	6		Visual Basic	4.14%	-1.09%
7	7		JavaScript	2.52%	+0.70%
8	10	▲	SQL	2.12%	+0.58%
9	9		Assembly language	1.38%	-0.21%
10	8	▼	PHP	1.29%	-0.49%

De ce C/C++ ?

Certificare internationala C++ Institute:

fiecare nivel 300 USD

- pentru formarea unei gandiri logice, algoritmice.
- foarte multe alte limbaje de programare se bazeaza pe sintaxa si pe conceptele folosite in C si C++. Exemple: Python, JavaScript, C#, PHP, etc.

Aplicatii C/C++/C#:

- implementari de algoritmi
- aplicatii desktop;
- programe sistem, compilatoare;
- masini virtuale; drivere;
- interfete grafice
- aplicatii in timp real
- jocuri, etc.
- apps pentru dispozitive mobile

PCLP 1

- Obiective
- Introducere
- Algoritmi. Limbajul C
- Functii de intrare/iesire.
- Operatori
- Instructiuni (1)
- Instructiuni (2)
- Functii
- Tablouri
- Algoritmi de sortare si cautare
- Recursivitate. Directive de preprocesare.
- Tablouri: exemple
- Stiva. Coada. Clase de memorie

PCLP 2

C (C++)

- Pointeri
- Alocare dinamica a memoriei
- Linia de comanda. Argumentele functiei main()
- Tipuri de date structurate: structuri, tablouri de structuri, enumerari , campuri de biti
- Fisiere
- Diferente C/C++

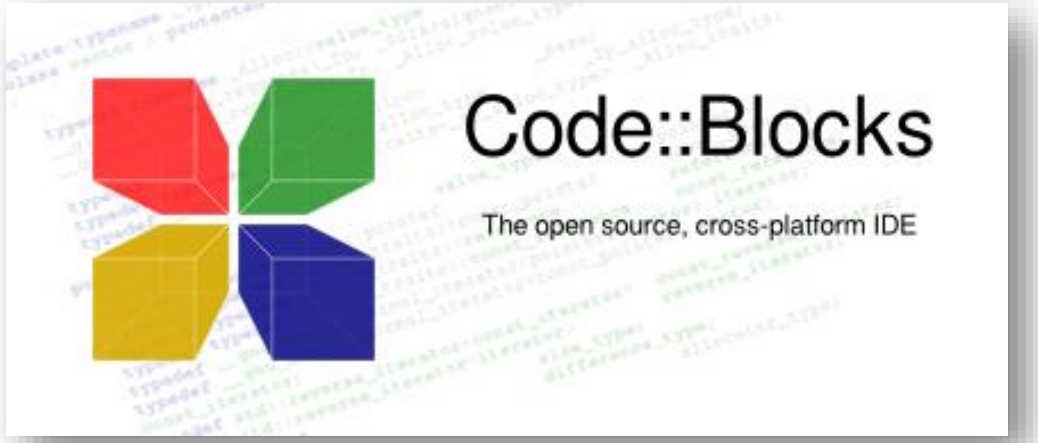
C++:

- STL
- Obiecte, clase si metode
- Mostenire

Aplicatii in inginerie: C/C++/Arduino/C#

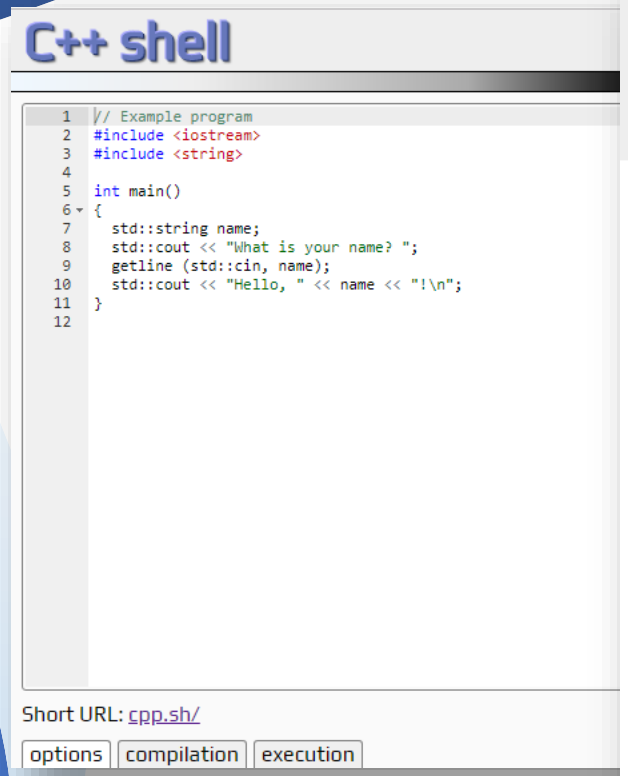
Medii de programare C/C++

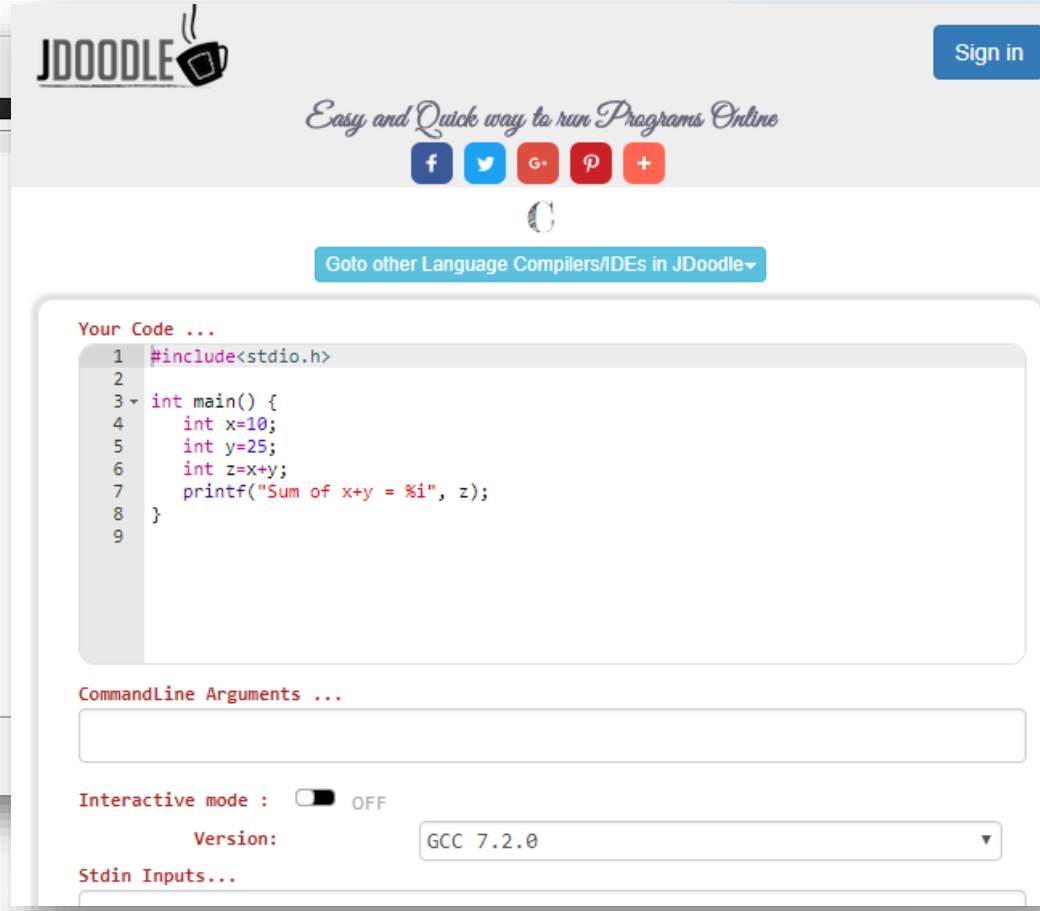
desktop



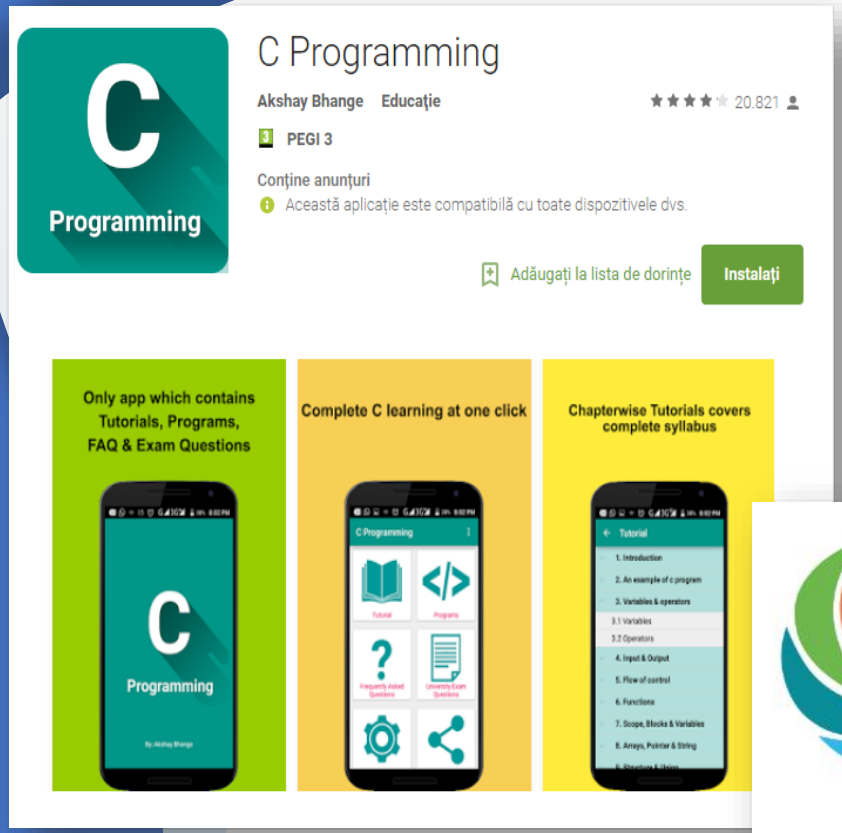
Medii de programare C/C++

On line





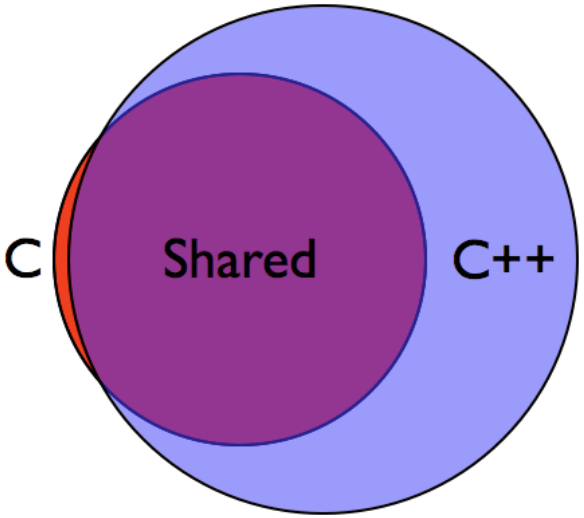
Apps Learn C/C++



Cap. 1

Introducere

pointeri

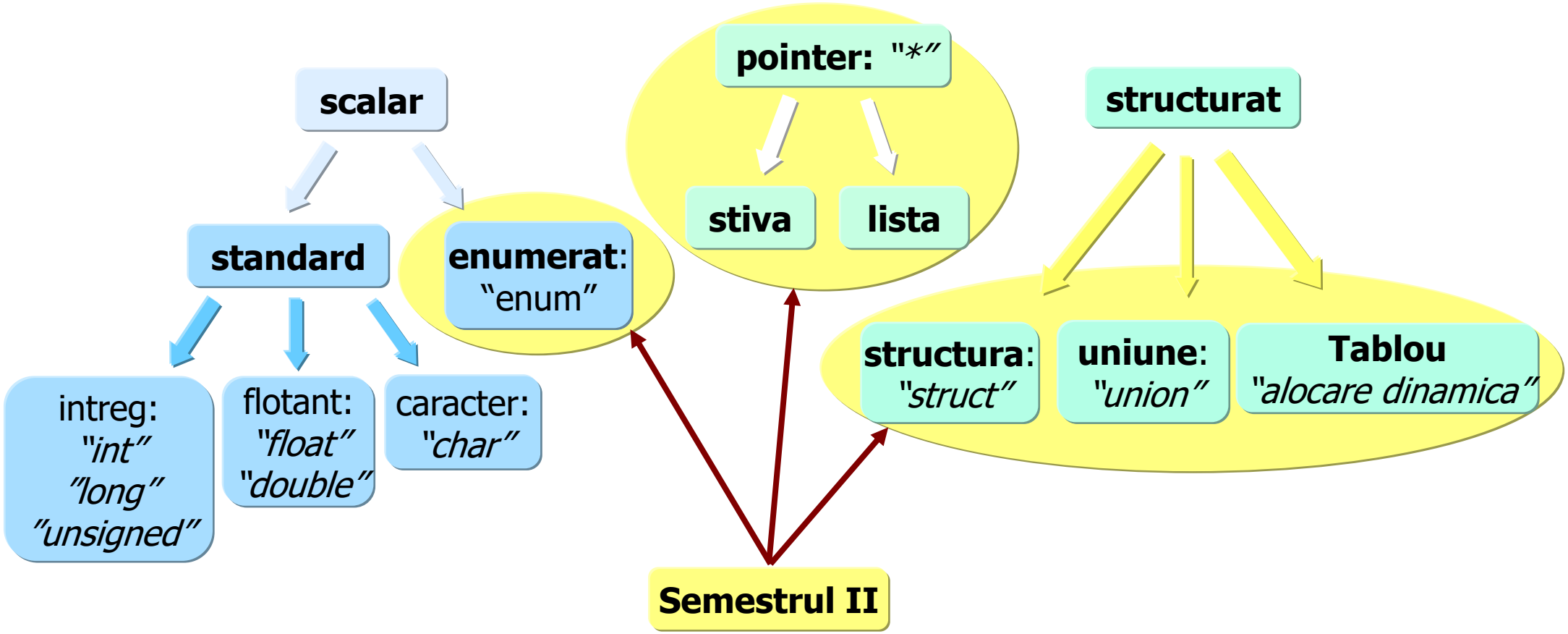


C++ integreaza aproape tot limbajulul C.

- C este un **limbaj procedural**
- C++ este un **limbaj orientat pe obiecte** si este o versiune mai complexa (updated version) a limbajului C.
- Majoritatea compilatoarelor C++ pot sa compileze programe in C (compilatoarele C nu pot compila programele in C++).

Cap 1: Introducere

Tipuri de date in C/C++



Cuprins Cap 1

1.1. Definirea pointerilor

Variabile si adrese

DEFINITII

“obiect” = o variabila sau un element dintr-o variabila ;

Fiecarui obiect ii corespund 2 “valori”:

- **rvalue (right value) = *valoarea*** asociata obiectului (trebuie specificata in dreapta semnului de atribuire)

- **lvalue (left value) = *adresa*** la care se gaseste obiectul

EXEMPLU

```
float k=4 ;
```

rvalue pentru variabila k este 4

lvalue este adresa la care este stocata variabila k in memorie pe lungime de 4 octeti (corespunzator tipului float)

1.1. Definirea pointerilor

Variabile si adrese

DEFINITII

“obiect” = o variabila sau un element dintr-o variabila ;

Fiecarui obiect ii corespund 2 “valori”:

- **rvalue (right value) = *valoarea*** asociata obiectului (trebuie specificata in dreapta semnului de atribuire)

- **lvalue (left value) = *adresa*** la care se gaseste obiectul

EXEMPLU

```
float k=4 ;
```

rvalue pentru variabila k este 4

lvalue este adresa la care este stocata variabila k in memorie pe lungime de 4 octeti (corespunzator tipului float)

TEST

Cati octeti sunt necesari pentru reprezentarea datelor?

Tip date
char
int
float
double

Cum putem verifica dimensiunea unei variabile x?

R: utilizand functia sizeof(x)

Ex.

```
double a;
printf("Nr octeti pentru tipul double: %d octeti \n",sizeof(a));
…
```

Output: 8

1.1. Definirea pointerilor

Valori si adrese

DEFINITII

“obiect” = o variabila sau un element dintr-o variabila ;

Fiecarui obiect ii corespund 2 “valori”:

- **rvalue (right value) = *valoarea*** asociata obiectului (specificata in dreapta semnului de atribuire)

- **lvalue (left value) = *adresa*** la care se gaseste obiectul

EXEMPLU

```
float k=4 ;
```

rvalue pentru variabila k este 4

lvalue este adresa la care este stocata variabila k in memorie pe lungime de 4 octeti (corespunzator tipului float)

1.1. Definirea pointerilor

Alocarea memoriei pentru variabile

EXEMPLU

Ex.1 Declararea variabilelor standard

```
…
int i=10, k[5];
float x=2.0,  z[20],  y=5.0;
…
```

Efect: La executia programului se aloca spatiu in memoria RAM pentru fiecare variabila

Variabila	Adresa (hexa)	Octeti	Intervalul alocat
i	1000	4	1000÷1003(inclusiv)
k	1004	20	1004÷1017
x	1018	4	1018÷101B
z	101C	80	101C÷106B
y	106C	4	106C÷106F

TEST

Cati octeti sunt necesari pentru reprezentarea celor 2 variabile?

```
float x[100];
double mat[3][3];
```

Raspuns :

pentru x se aloca 4*100=400 octeti

pentru mat se aloca 3*3*8= 72 octeti

1.1. Definirea pointerilor

Definitie pointer

DEFINITII

pointer = o variabila care are ca si valoare o adresa de memorie

Clasificarea variabilelor pointer

Pointer la date = contine adresa unei variabile sau a unei constante
Pointer la functii = contine adresa codului executabil al unei functii
Pointer la obiecte= contine adresa unui obiect in memorie =adrese de date si functii

Pointerii contin adrese nu valori !!!

1.1. Definirea pointerilor

Declarare pointer

Sintaxa

Format declarare: **tip *nume ;**

unde **tip=** tipul de baza al pointerului, defineste tipul variabilei la care indica acesta; poate fi orice tip de variabila;

nume= numele variabilei pointer

EXEMPLU

Ex.

```
int  *np;     //np este pointer la intreg

float *a, r;     //a este pointer la tipul float , r este variabila de tip float

char c,*pc;    //variabila c este de tip caracter si  pc este pointer la tipul caracter

int *ap[10];
```

Ce tip variabila este declarata astfel?

1.1. Definirea pointerilor

Operatori specifici pentru pointeri: &, *

DEFINIRE

Operatorul de adresare (referentiere): & (“adresa”) = operator care asociat unei variabile sau obiect, returneaza adresa de memorie a acelei variabile sau obiect

Dupa declarare, pointerul trebuie initializat cu adresa unei variabile (obiect) altfel acesta nu poate fi utilizat

EXEMPLE

Ex. 1: declarare si initializare in 2 instructiuni

```
double nr,*m;     //declarare
m=&nr ;           //initializare
```

Ex.2: declarare si initializare intr-o singura instructiune

```
char c,*pc =&c;
```

Singura instructiune in care pot aparea ambii operatori simultan

1.1. Definirea pointerilor

Operatori specifici pentru pointeri: &, *

DEFINIRE

Operatorul de indirectare (dereferentiere): * (“de la adresa”) = operator unar, complementar lui &, returneaza valoarea de la adresa de memorie specificata

EXEMPLE

Ex. 1: vrem sa atribuim lui q valoarea lui nr utilizand pointerul *m

```
int nr=100,q,*m;
m=&nr ;  //variabilei m i se atribuie adresa lui nr
q=*m ;  //variabilei q i se atribuie valoarea lui nr prin pointerul *m ⇔q=nr;
```

Efect: atribuie lui q valoarea variabilei nr indirect prin pointerul *m

Ex.2: vrem sa schimbam valoarea lui i prin pointerul *p

```
int i=1,*p=&i;      //variabilei p i se atribuie adresa lui i
*p=2;     //variabilei i, i se  atribuie valoarea 2 ⇔i=2;
```

Efect: atribuire indirecta prin pointer

1.1. Definirea pointerilor

**Operatori specifici pentru pointeri: &, *

Operatorul * nu trebuie confundat cu operatorul aritmetic utilizat pentru inmultire
Operatorul & nu trebuie confundat cu operatorul AND pentru biti
Operatorii * si & au prioritate fata de toti operatorii aritmetici (exceptie –unar ,aceeasi precedenta)
Atentie la declararea tipului pointerului: vezi Ex.

EXEMPLU

Ex: Urmatoarea secventa de program se compileaza corect (eventual cu avertismente =warning) dar nu produce rezultatul corect, deci nu se atribuie valoarea lui x lui y datorita neconcordantei tipurilor de date

```
double x=10.5, y;        float *p;
p=&x;
y=*p; //instructiunea nu produce rezultatul asteptat, pentru ca float se
        //reprezinta pe 4 octeti iar  double pe 8 octeti
```

Efect: p este declarat de tip float, deci variabilei y i se vor transfera numai 4 octeti (float) din informatia stocata , nu 8 octeti (double)

TEST

Indicati secventa corecta pentru declararea unui pointer la un intreg, numit address**:**

a. int address;
b. address *int;
c. *int address;
d. int *address;

Raspuns corect

d

1.1. Definirea pointerilor

Initializarea variabilelor pointer

Dupa declararea pointerului acesta trebuie initializat

EXEMPLU

INCORECT: generare eroare !

```
int *ip;
*ip = 100;
```

CORECT: declararea si initializarea pointerului

```
int *ip,  x;
ip = &x;
*ip = 100;
```

Ex. Declarare pointer si initializare pointer prin **2 instructiuni**

```
int a=1, *ptoa;
ptoa = &a;          //pointerului ptoa i se atribuie adresa variabilei  a
```

Ex. Declarare pointer cu initializare **intr-o singura instructiune**

```
int a=1, *ptoa= &a;          //pointerului ptoa i se atribuie adresa variabilei  a
```

TEST

Ex: int *p1,x; p1=&x;

Cum tiparim un pointer p1 in C?

```
printf("%?", p1 );
```

adresa continuta de pointer

```
printf("%p", p1 );
printf("%x", p1 );
printf("%x", &x );
```

valoarea continuta la adresa indicata de pointer

```
printf("%d", *p1 ); //val continuta de pointer
```

Cum tiparim numere in alte baze de numeratie decat 10 (8 sau 16) in C?

```
int a=11;
printf ("%x, a); //baza 16
printf ("%o", a); //baza 8
```

Cum tiparim un pointer p1 in C++?

```
cout << ? ;
```

adresa continuta de pointer

```
cout<< p1;
cout<<&x;
```

valoarea continuta la adresa indicata de pointer

```
cout<< *p1;
```

Cum tiparim numere in alte baze de numeratie decat 10 (8 sau 16) in C++?

```
int a=11;
cout <<hex<< a; //baza 16
cout<<oct<<a; //baza 8
```

1.1. Definirea pointerilor

Expresii cu pointeri

EXEMPLE

Ex.1 Pointeri la intregi

```
#include <stdio.h>
int main(){
int i1, i2, *p1;
i1 = 5; p1 = &i1;
i2 = *p1 / 2 + 10;
printf("i1 = %d, i2 = %d", i1, i2);
return 0;}
```

Ce rezultat va fi afisat?

```
i1 = 5, i2 = 12
```

Ex.2 Pointeri la caractere

```
#include <stdio.h>
int main() { char c = 'Q';
char *pc = &c;  printf("%c %c\n", c, *pc);
c = '/'; printf("%c %c\n", c, *pc);
*pc = '('; printf("%c %c\n", c, *pc);
return 0;}
```

```
Q Q
/ /
( (
```

1.2. Operatii cu pointeri

Operatii permise

- Comparare
- Initializare. Atribuire
- Adunarea/Scaderea unui nr. intreg la/dintr-un pointer
- Scaderea a 2 pointeri
- Incrementare/Decrementare

Operatii nepermise

- Adunarea a 2 sau mai multi pointeri
- Adunarea/scaderea tipului float /double la/din pointeri
- Inmultirea a 2 sau mai multi pointeri, sau *cu constanta
- Impartirea a 2 sau mai multi pointeri sau / cu constanta
- Aplicarea operatorilor pe bit

1.2. Operatii cu pointeri

Operatii cu pointeri: comparare

Comparare 2 pointeri : operatorii relationali permit compararea a 2 pointeri intr-o expresie relationala numai daca acestia indica spre obiecte de acelasi tip.

EXEMPLU

```
int *p, *q, k=2,j=3;
p=&k;     q=&j;
if (p<q) printf (“p indica o adresa de memorie mai mica decit q”);
printf(“p=%p q=%p\n”,p,q );
```

Comparare pointeri cu NULL sau 0

Operatorii de egalitate = = si ! = permit compararea cu constanta NULL (definita in C in <stdio.h>)

EXEMPLU

```
#define NULL 0
void *p        //p=pointer generic (nu este asociat nici unui tip de date )
p==NULL;   p!=NULL //verifica daca p a fost initializat
```

In C++ se recomanda comparatia cu 0 : p==0 si p!=0

1.2. Operatii cu pointeri

Operatii cu pointeri: initializare/atribuire

DEFINIRE

- Format : tip *nume =const;
- unde tip = tipul de baza al pointerului , defineste tipul variabilei la care indica acesta;
- nume = numele variabilei pointer,
- const = adresa unei variabile, tablou, functie,etc.

EXEMPLE

Ex.1: initializare cu NULL: void p=NULL;
Ex.2: prin atribuire directa: int x,*a=&x;
Ex.3: prin declarare si atribuire: int x,*a; a=&x; *a=2

1.2. Operatii cu pointeri

Operatii cu pointeri: adunarea/scaderea unui nr. intreg la/dintr-un pointer

Format adunare: tip *p1; atunci p1+n si p1-n ⇔ adunarea/scaderea la adresa indicata de p a valorii n*sizeof(tip)

EXEMPLU

p1+=10⇔p1=p1+10; p1 va indica catre al 10-lea element de acelasi tip cu p1
p2=p1-1; p2 va indica catre elementul anterior (de acelasi tip) lui p1

Format scadere: tip *p1,*p2; atunci p1-p2 este permisa numai pentru elemente de acelasi tip, rezultatul reprezentand nr. de obiecte ce separa cei doi pointeri

EXEMPLU

Ex: int q; float j[3], *p1=&j[1],*p2=&j[3]; q=p2-p1;
Efect: variabilei q i se atribuie nr. de elemente dintre cele doua adrese =2

TEST

Se considera secventa de instructiuni:

```
float numere[]={0,1,2,3,5,8,13}, *p1, *p2;
p1=&numere[1];  p2=&numere[6];
```

Considerand ca tipul float este reprezentat pe 4 octeţi, care va fi valoarea lui (p2-p1) ?

a) 8;
b) 5;
c) 4.
d) operaţie nepermisă;

Raspuns corect

- **b**

1.2. Operatii cu pointeri

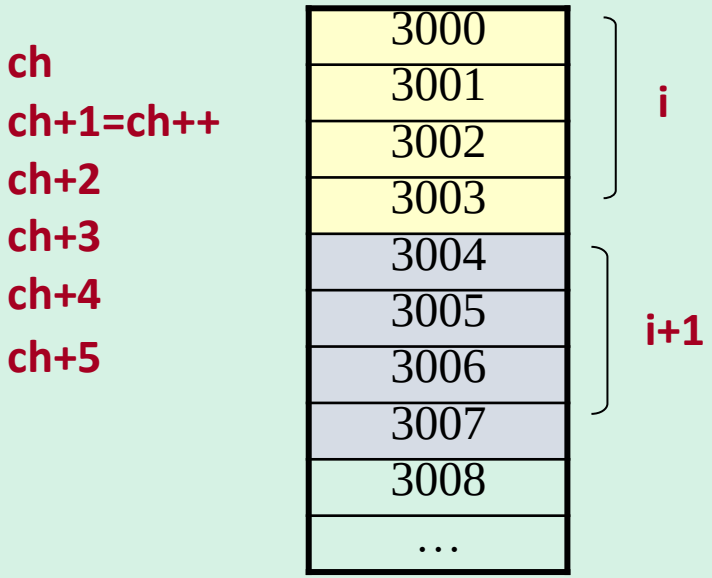
Operatii cu pointeri: incrementare/decrementare pointer

Format ++,--: tip *p1; atunci p++ si p-- ⇔ **adunarea /scaderea cu sizeof(tip).**

- incrementarea/decrementarea **nu reprezinta adunarea/scaderea propriuzisa cu 1** (nici a adresei nici a valorii spre care indica pointerul respectiv).
- se utilizeaza la tablouri, dupa incrementare/decrementare pointerul va indica spre elementul urmator/anterior de acelasi tip cu tipul sau de baza

EXEMPLE

Ex.3: char *ch=3000; int *i=3000;



1.2. Operatii cu pointeri

Operatii cu pointeri: exemplu

EXEMPLE

Ex.1:se considera 4 variabile de tip int, float, double si char carora le corespunde cate un pointer.
Programul realizeaza operatii de comparare,adunare,scadere,incrementare,etc.

```
#include <stdio.h>
void main()
{int i=10, *p1, *p2,*p3,*p4;
float f=0.98,*p5,*p6,*p7,*p8;
double d=-32.97,*p9,*p10,*p11,*p12;
char c='A', *p13,*p14,*p15,*p16;
p2=&i;  p1=p2-1; p3=p2+1; p4=p2+2;
printf("\ni=%d, p=&i=%p\n", i,p2);
printf("p-1=%p; p=%p\np+1=%p,p+2=%p\n", p1,p2,p3,p4);
printf("comparare p si p+1:\n"); if (p2<p3) printf("%p<%p\n",p2,p3);
p6=&f;  p5=p6-1;p7=p6+1;p8=p6+2;
printf("\nf=%f, p=&f=%p\n", f,p6);
printf("p-1=%p; p=%p\np+1=%p,p+2=%p\n", p5,p6,p7,p8);
p10=&d; p9=p10-1;p11=p10+1;p12=p10+2;
printf("\nd=%lf, p=&d=%p\n", d,p10);printf("p-1=%p; p=%p\np+1=%p,p+2=%p\n", p9,p10,p11,p12);
p14=&c; p13=p14-1;p15=p14+1;p16=p14+2;
printf("\nc=%c, p=&c=%p\n", c,p14);
printf("p-1=%p; p=%p\np+1=%p,p+2=%p\n", p13,p14,p15,p16);printf("incrementare p++=%p\n", p14++);}
```

```
i=10, p=&i=0060FEBC
p-1=0060FEB8; p=0060FEBC
p+1=0060FEC0,p+2=0060FEC4
comparare p si p+1:
0060FEBC<0060FEC0

f=0.980000, p=&f=0060FEB8
p-1=0060FEB4; p=0060FEB8
p+1=0060FEBC,p+2=0060FEC0

d=-32.970000, p=&d=0060FEB0
p-1=0060FEA8; p=0060FEB0
p+1=0060FEB8,p+2=0060FEC0

c=A, p=&c=0060FEAF
p-1=0060FEAE; p=0060FEAF
p+1=0060FEB0,p+2=0060FEB1
incrementare p++=0060FEAF
```

TEST kahoot

Pentru login, introduceti codul afisat pe ecran, in browser la adresa:

http://kahoot.it

1.3. Pointeri si tablouri

Pointeri la tablouri

Accesul la elementele tablourilor se realizeaza in C/C++ in 2 moduri:

- prin intermediul indicilor (Sem I)
- Prin pointeri (Sem II) . Avantaj:accesul la elemente mult mai rapid ⇒creste viteza de executie a programului

Nume tablou (sir/matrice) - scris fara indice **=** **pointer constant**, este de tipul elementelor tabloului si are ca valoare adresa primului element al tabloului

DEFINIRE

Tablou unidimensional=sir : tip a[i],*p; unde tip= tip de baza (int, long, float, double, char, etc) , i=0,n-1 si pointerul *p de acelasi tip cu elementele tabloului a[i]

NOTATII ECHIVALENTE (p pointer la tabloul a[]): p=a ⇔p=&a ⇔p=&a[0]

denumirea tabloului = pointerul la tablou= adresa primului element al tabloului (&a[0])

NOTATII ECHIVALENTE:

- Adresa tabloului a[]: a ⇔ &a ⇔ &a[0]
- Adresa variabilei a[i]: &a[i] ⇔ a+i ⇔ p+i
- Valoarea variabilei a[i]: a[i] ⇔ *(a+i) ⇔ *(p+i) ⇔ p[i] ⇔ *a+I ⇔ *p+i

1.3. Pointeri si tablouri

Pointeri la siruri

tip a[i],*p; a = constanta (pointer constant), pointerul la tablou trebuie declarat ca variabila pointer

EXEMPLU

Ex: p este o **variabila pointer,** iar a **este pointer constant**

```
char a[10], *p;
p=&a[0];    // sau p=a; sau p=&a
printf (a); // sau printf(p); //tipareste intreg sirul de caractere
```

Obs: doar pentru siruri de caractere

1.3. Pointeri si tablouri

Pointeri la siruri

EXEMPLE

Ex.1 : Se afiseaza elementele unui sir de numere intregi utilizand pointer=numele sirului

```
#include <stdio.h>
int main()
{int ary[]={1,2,3,4};
printf("%d ",*ary); //afiseaza primul element din sir : 1
printf("%d ",*ary+1); //al doilea: 2 ⇔ printf("%d ",*(ary+1));
printf("%d ",*ary+2); //al  3-lea : 3 ⇔ printf("%d ",*(ary+2));
printf("%d
return 0
```

1 2 3 4

Cum afisam elementele printr-o bucla?

Ex.2 : Aceeasi problema utilizand bucla for si notatiile echivalente:

```
#include <stdio.h>
int main()
{ int i;
int ary[]={1,2,3,4};
for (i=0;i<4;i++)
printf("%d ",*ary+i);
return 0;}
```

```
#include <stdio.h>
int main()
{ int i;
int ary[]={1,2,3,4}, *p=ary;
for (i=0;i<4;i++)
printf("%d ",*p+i);
return 0;}
```

```
#include <stdio.h>
int main()
{ int i;
int ary[]={1,2,3,4}, *p=ary;
for (i=0;i<4;i++)
printf("%d ",p[i]);
return 0;}
```

1.3. Pointeri si tablouri

Pointeri la siruri

Ce afiseaza programul din Ex.3?

EXEMPLE

Ex.3: afisarea unui sir de caractere

```
#include <stdio.h>
#include <string.h>
char *p=“Inginerie Electrica";
int main(void)
{  int i;
printf(p);  printf("\n");
for (i=strlen(p)-1;i>=0;i--)
    printf("%c", p[i]);
return 0;}
```

```
Inginerie Electrica
acirtcelE eirenignI
```

1.3. Pointeri si tablouri

Pointeri la siruri

EXEMPLU

Ex.4 : Programul calculeaza suma tuturor elementelor si produsul elementelor strict pozitive ale unui sir de nr. intregi, utilizand pointeri

```
#include <stdio.h>
int main ()
{int a[10]={-1,0,1,10,2,-2,0,-5,1,-4};
int *pa, s=0, p=1,i;
pa=a;
printf("tiparirea valorilor sirului cu a[i] si prin pointer");
for (i=0;i<10;i++)
{   printf("\na[%d]=%3d, *(a+%d)=%3d\n",i,a[i],i,*(pa+i));
    s+=*(pa+i); //s+=a[i];
    if(*(pa+i)>0) p*=*(pa+i);}  //sau if (a[i]>0) p*=a[i];
printf("suma tuturor elementelor sirului:\n");
printf("s+= *(pa+i)=%d\n",s);
printf("produsul elementelor strict pozitive ale sirului:\n");
printf("p*= *(pa+i)=%d\n", p);
return 0;}
```

```
tiparirea valorilor sirului cu a[i] si prin pointer
a[0]= -1, *(a+0)= -1

a[1]=  0, *(a+1)=  0

a[2]=  1, *(a+2)=  1

a[3]= 10, *(a+3)= 10

a[4]=  2, *(a+4)=  2

a[5]= -2, *(a+5)= -2

a[6]=  0, *(a+6)=  0

a[7]= -5, *(a+7)= -5

a[8]=  1, *(a+8)=  1

a[9]= -4, *(a+9)= -4
suma tuturor elementelor sirului:
s+= *(pa+i)=2
produsul elementelor strict pozitive ale sirului:
p*= *(pa+i)=20
```

1.3. Pointeri si tablouri

Pointeri la siruri

EXEMPLU

Care e varianta corecta?

Ex.5 : Programul initializeaza cu 0 elementele unui sir de nr. intregi

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

Ex.6 : Programul initializeaza cu 0 elementele unui sir de nr. intregi

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

Toate sunt corecte

```
0  0  0  0  0
```

Ex.7 : Programul initializeaza cu 0 elementele unui sir de nr. intregi

```
#include <stdio.h>
int main()
{int a = 0, b = 0, c = 0, d=0, e=0, i;
int array[5] = {a, b, c, d,e};
for (i=0;i<5;i++) printf(" %d ", array[i]);
return 0;}
```

Ex.8 : Programul initializeaza cu 0 elementele unui sir de nr. intregi, utilizand pointeri

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

1.3. Pointeri si tablouri

Pointeri la siruri

EXEMPLU

Ex.9 : Programul initializeaza elementele unui sir de nr. intregi cu valori de la 1 la 100, utilizand pointeri :

```
#include <stdio.h>
int main()
{ int i,array[100],*p;
p=array;
//for (i=1;i<=100;i++) {p[i]=i; printf(" %d ", p[i]);}//var 1
for (i=0;i<100;i++,p++) {*p=i+1; printf("%d ",*p);} //var 2
 return 0;
}
```

```
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23
24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43
44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63
64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83
84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99 100
```

1.3. Pointeri si tablouri

Pointeri la siruri

EXEMPLU

Ex.10: Programul afiseaza un sir de caractere utilizand un pointer cu index.

```
#include <stdio.h>
int main()
{char sir[]="Pointeri la siruri";
char *p; int i;
p=sir;
printf(p);printf("\n");//prima varianta –tot sirul , echivalent cu printf(sir);
for (i=0;p[i];i++) printf("%c", p[i]); printf("\n");//a 2-a varianta character cu caracter
for (i=0;p[i];i++) putchar(p[i]);printf("\n"); //a 3-a varianta character cu caracter
while(*p) putchar(*p++); //a 4-a varianta character cu character
return 0;}
```

```
Pointeri la siruri
Pointeri la siruri
Pointeri la siruri
Pointeri la siruri
```

Un pointer se poate indexa in mod similar cu un tablou daca este pointer la un tablou

1.3. Pointeri si tablouri

Pointeri la siruri

EXEMPLU

Ex.11: Functia **scriesir()** afiseaza un sir de caractere, cate un caracter

Solutia 1: accesul prin pointer

```
void scriesir(char *s)
{while(*s) putchar(*s++);}
```

Solutia 2: accesul prin pointer cu index

```
void scriesir(char *s)
{int i;
for (i=0;s[i];++i) putchar(s[i]);}
```

1.3. Pointeri si tablouri

Pointeri la siruri

EXEMPLU

Ce functie copiaza un sir in alt sir si in ce biblioteca se afla?

strcpy() in <string.h> (C)

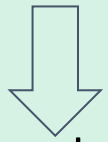
Ex.12 Functiile sircpy() copiaza un sir t intr-un alt sir s:

Solutia 1: versiunea cu siruri si index

```
void sircpy1(char s[],char t[])
{ int i;i=0;
  while ((s[i]=t[i])!='\0') i++;}
```

Solutia 2: versiunea 1-a cu pointeri

```
void sircpy2(char *s,char *t)
{ while ((*s=*t)!='\0')
  {s++; t++;}}
```



Solutia 3: versiunea a 2-a cu pointeri condensata

```
void sircpy3(char *s,char *t)
{ while (*s++=*t++) ;}
```

1.3. Pointeri si tablouri

Pointeri la matrici

DEFINIRE

Fie matricea: **a[i][j], i=0,m-1; j=0,n-1,** si pointerul p de acelasi tip cu tipul matricii,
Conventie: p=a denumirea matricii = adresa primului element al matricii (a[0][0])
NOTATII ECHIVALENTE:
Adresa matricii **:** **a ⇔ &a[0][0] ⇔ &a**
Adresa variabilei **a[i][j]:** **&a[i][j] ⇔ a[i]+j ⇔ *(a+i)+j**
Valoarea variabilei **a[i][j]:** **a[i][j] ⇔ *(a[i]+j) ⇔*(*(a+i)+j)**

Daca matricea e patratica (n=m)
Adresa a[i][j]: p+k, k=i*n+j
Valoarea a[i][j]: *(p+k) , k=i*n+j
Ex: Matrice de 9 elemente

1	**2**	**3**
4	**5**	**6**
7	**8**	**9**

a

1	2	3	4	5	6	7	8	9

p= a+i*n+j

a[i][j]

n=3, p=a+i*3+j

1.3. Pointeri si tablouri

Pointeri la matrici

Ce dimensiune are zona de memorie ocupata de tablou?

Tip date	**sir**	**matrice**
int (4 octeti)	int a[10]=10*4=40 octeti	int a[10][10]=10*10*4=400 octeti
float(4 octeti)	float a[10]=10*4=40 octeti	float a[10][10]=10*10*4=400 octeti
char(1 octet)	char a[10]=10*1=10 octeti	char a[10][10]=10*10*1=100 octeti
double(8 octeti)	double a[10]=10*8=80 octeti	double a[10][10]=10*10*8=800 octeti

1.3. Pointeri si tablouri

Pointeri la matrici

EXEMPLU

Ex.1:. Afisarea unei matrici utilizand pointeri

```
#include <stdio.h>
int main()
{int i, j, n,m,a[100][100];
printf ("n=");scanf("%d", &n);
printf ("m=");scanf("%d", &m);
//citire elemente matrice
for (i=0;i<n;i++)
for (j=0;j<m;j++)
{printf("  a[%d,%d]=", i,j); scanf("%d", *(a+i)+j); }
//afisare elemente matrice cu notatii echivalente
for (i=0;i<n;i++)
for (j=0;j<m;j++)
{printf("\na[%d][%d]=%3d", i,j,a[i][j]);
printf("  *a[%d]+%d=%3d", i,j,*(a[i]+j));
printf("  *(*(a+%d)+%2d)=%d", i,j,*(*(a+i)+j));}
printf("\n");
return 0;}
```

```
n=2
m=3
  a[0,0]=1
  a[0,1]=2
  a[0,2]=3
  a[1,0]=4
  a[1,1]=5
  a[1,2]=6

a[0][0]=  1  *a[0]+0=  1  *(*(a+0)+ 0)=1
a[0][1]=  2  *a[0]+1=  2  *(*(a+0)+ 1)=2
a[0][2]=  3  *a[0]+2=  3  *(*(a+0)+ 2)=3
a[1][0]=  4  *a[1]+0=  4  *(*(a+1)+ 0)=4
a[1][1]=  5  *a[1]+1=  5  *(*(a+1)+ 1)=5
a[1][2]=  6  *a[1]+2=  6  *(*(a+1)+ 2)=6
```

1.3. Pointeri si tablouri

Pointeri la matrici

EXEMPLU

Ex.2 Determinarea max. elementelor unei matrici utilizand pointeri

```
#include <stdio.h>
#include <stdlib.h>
int main()
{int max;
int a[10][10],i,j,n,m,*p;
p=&a[0][0];
printf("n=");scanf("%d", &n);
printf("m=");scanf("%d", &m);
for (i=0;i<n;i++)
for (j=0;j<m;j++)
{ printf("a[%d][%d]=",i,j); scanf("%d", *(a+i)+j);}
max=*p;
for (i=0; i < n; i++)
  for (j=0; j < m; j++)
     if (max < *(*(a+i)+j))  max = *(*(a+i)+j);
printf("\n max din matrice este %d\n",max);
return 0;}
```

```
n=2
m=2
a[0][0]=1
a[0][1]=2
a[1][0]=3
a[1][1]=4

max din matrice este 4
```

Cum calculam minimul elementelor matricii?

1.3. Pointeri si tablouri

Pointeri la matrici

EXEMPLU

Ex.2 Determinarea max. si a pozitiei acestuia dintre elementele unei matrici utilizand pointeri

```
#include <stdio.h>
#include <stdlib.h>
int main()
{int max, k,q;
int a[10][10],i,j,n,m,*p; p=&a[0][0];
printf("n=");scanf("%d", &n);
printf("m=");scanf("%d", &m);
for (i=0;i<n;i++)
for (j=0;j<m;j++)
{ printf("a[%d][%d]=",i,j); scanf("%d", *(a+i)+j);}
max=*p;
for (i=0; i < n; i++)
  for (j=0; j < m; j++)
     if (max < *(*(a+i)+j)) { max = *(*(a+i)+j); k=i;q=j;}
printf("\n max din matrice este %d\n",max);
printf("\n pozitia max din matrice:%d,%d\n",k,q);return 0;}
```

```
n=2
m=3
a[0][0]=1
a[0][1]=2
a[0][2]=3
a[1][0]=4
a[1][1]=5
a[1][2]=6

 max din matrice este 6

 pozitia max din matrice:1,2
```

Cum afisam pozitia maximului in sir?

TEST kahoot

Pentru login, introduceti codul afisat pe ecran, in browser la adresa:

http://kahoot.it

1.3. Pointeri si tablouri

Tablouri de pointeri

DEFINIRE

Format declarare :

- **Tablou unidimensional** (sir):
  tip *nume[max], max=dimensiunea maxima a sirului de pointeri
- **Tablou bidimensional** (matrice):
  tip *nume[max1][max2], max1/max2=nr maxim de elemente pe linie /coloana

EXEMPLE

Ex.1. Un tablou unidimensional de pointeri de tipul int cu 10 elemente:

```
int  *x[10], var=100;
x[2]=&var ; // se atribuie elementului al 3-lea adresa variabilei intregi var
printf("%d", *x[2]); //se tipareste valoarea indicata de cel de-al 3-lea pointer
                     //din tabloul de pointeri
```

Ex.2. O matrice de pointeri de tipul float cu marimea de 10x10 elemente:

```
float  *x[10][10], var=1.5;
x[0][2]=&var ; printf("%f", *x[0][2]) // se tipareste valoarea variabilei var
```

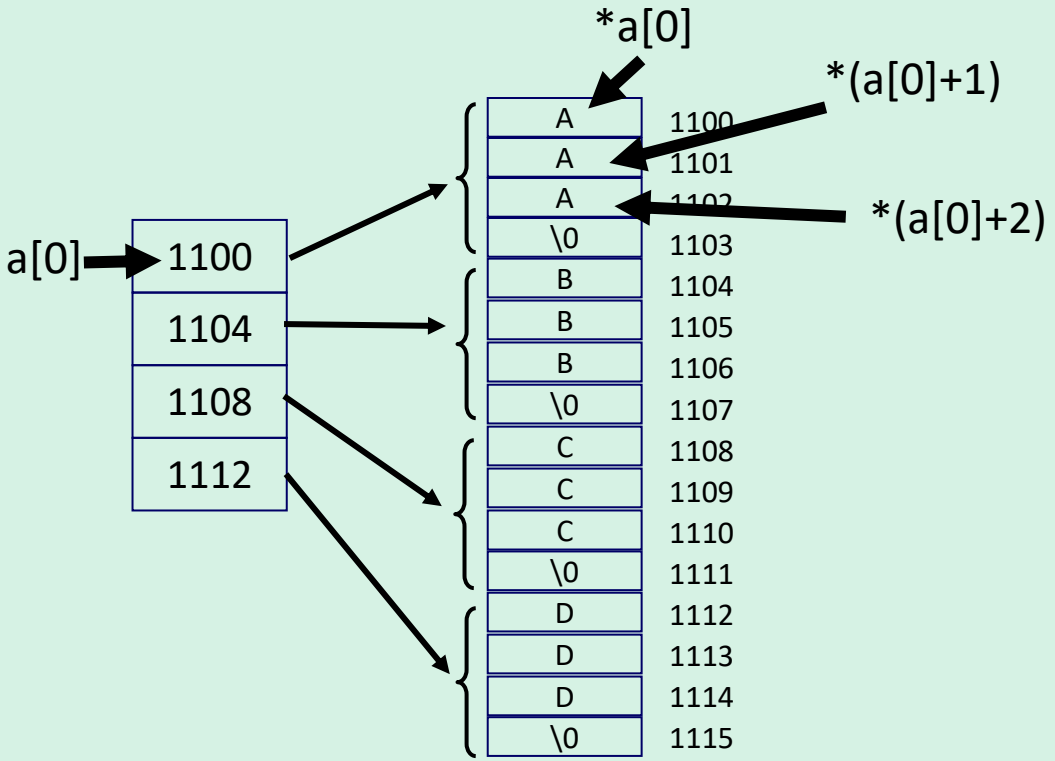
1.3. Pointeri si tablouri

Tablouri de pointeri

EXEMPLE

Ex.3. Definirea unui tablou de pointeri la siruri de caractere:

char *a[ ]={“AAA”,”BBB”,”CCC”,”DDD”};



1.3. Pointeri si tablouri

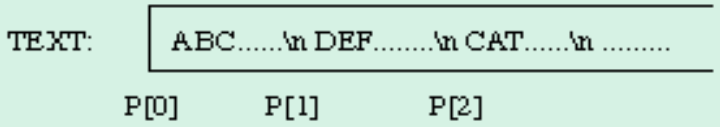
Tablouri de pointeri

Ce functie compara 2 siruri?

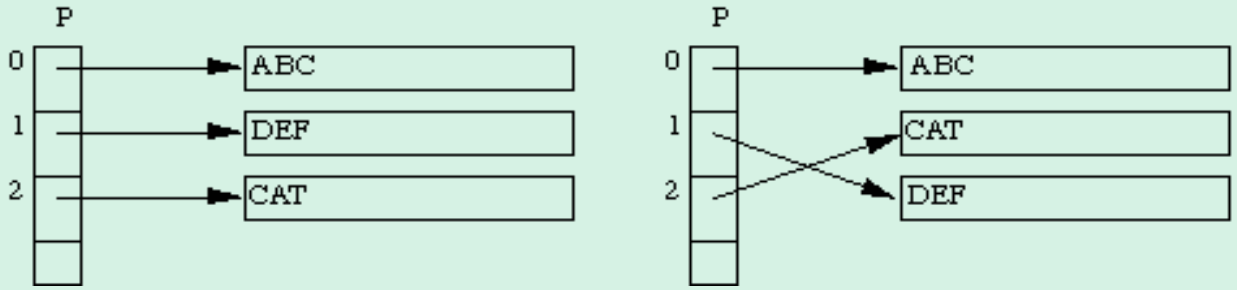
EXEMPLE

La ce sunt utile tablourile de pointeri ?

Ex.4. Sortarea in ordine alfabetica a unui tablou de siruri de caractere de diferite lungimi.



strcmp() in C: <string.h>



Obs. : sirurile de caractere nu pot fi comparate/interschimbate intr-o singura operatie.

Mod de lucru:

- se stocheaza fiecare linie de text terminata cu "\n" intr-un **sir de caractere**
- se creeaza un **tablou de pointeri**, fiecare pointer va contine adresa primului caracter din fiecare sir
- se compara 2 siruri de caractere utilizand functia strcmp() si pointerii
- daca nu sunt in ordinea ceruta atunci cele 2 siruri se schimba intre ele, prin **schimbarea pointerilor in tabloul de pointeri** (nu a sirurilor de text!)

1.3. Pointeri si tablouri

Tablouri de pointeri

EXEMPLE

Ex.4. Sortarea in ordine alfabetica a unor siruri de caractere de diferite lungimi (inclusiv nume si prenume) utilizand un tablou de pointeri la siruri de caractere.

```
#include <stdio.h>
#include <string.h>
int sort(char*a[],int n);
int main()
{char *sir[]={"Popescu","Morar","Adam","Zidaru","Marin"};
int i; printf("Sir nesortat\n");
for (i=0;i<5;i++) printf("%s,",sir[i]);
sort(sir,5); printf("\n Sir sortat\n");
for (i=0;i<5;i++) printf("%s,",sir[i]);
return 0;}
int sort(char*a[],int n)
{int i,j; char *aux;
for (i=0;i<n;i++)  for (j=0;j<n;j++)
  if(strcmp(a[i],a[j])<0)   {aux=a[i];   a[i]=a[j];   a[j]=aux;} return 0;}
```

```
Sir nesortat
Popescu,Morar,Adam,Zidaru,Marin,
Sir sortat
Adam,Marin,Morar,Popescu,Zidaru,
```

1.3. Pointeri si tablouri

Tablouri de pointeri

EXEMPLE

Ex.5. Se defineste un tablou de pointeri la siruri de caractere

```
#include<stdio.h>
int main()
{char *zile[]={"Luni", "Marti", "Miercuri", "Joi", "Vineri",
                "Sambata", "Duminica"};
int i;
for(i=0;i<7;i++)
     printf("*zile[%d]=%-10s;zile[%d]=%p\n",i,zile[i], i,zile[i]);
for(i=0;i<7;i++)
     printf("*(zile[5]++)=%c\n", *(zile[5]++));
 return 0;}
```

```
*zile[0]=Luni       ;zile[0]=00403024
*zile[1]=Marti      ;zile[1]=00403029
*zile[2]=Miercuri   ;zile[2]=0040302F
*zile[3]=Joi        ;zile[3]=00403038
*zile[4]=Vineri     ;zile[4]=0040303C
*zile[5]=Sambata    ;zile[5]=00403043
*zile[6]=Duminica   ;zile[6]=0040304B
*(zile[5]++)=S
*(zile[5]++)=a
*(zile[5]++)=m
*(zile[5]++)=b
*(zile[5]++)=a
*(zile[5]++)=t
*(zile[5]++)=a
```

1.3. Pointeri si tablouri

Tablouri de pointeri

EXEMPLE

```
#include <stdio.h>
int main()
{char *nume[]={"Alex","Ion","Mircea",NULL};
printf("%c\n", *nume[0]);        //A
printf("%c\n", *(nume[0]+1)); // l
printf("%c\n", *(nume[0]+2)); //e
printf("%c\n", *(nume[0]+3)); //x
printf("\n");
printf("%c\n", *nume[1]);        //I
printf("%c\n", *(nume[1]+1)); //o
printf("%c\n", *(nume[1]+2)); //n
printf("\n");
printf("%c\n", *nume[2]);        //M
printf("%c\n", *(nume[2]+1)); //i
printf("%c\n", *(nume[2]+2)); //r
printf("%c\n", *(nume[2]+3)); //c
printf("%c\n", *(nume[2]+4)); //e
printf("%c\n", *(nume[2]+5)); //a
printf("\n"); printf("%c\n", *(nume[1]+3) ); // linie goala
printf("%s\n", *nume);       // Alex sau printf("%s\n", nume[0]);
printf("%s\n", *nume+1);   //lex
printf("%s\n", *(nume+1)); //Ion sau printf("%s\n", nume[1]);
printf("%s\n", *(nume+2)); //Mircea
return 0;}
```

```
A
l
e
x

I
o
n

M
i
r
c
e
a


Alex
lex
Ion
Mircea
```

TEST

Se considera instructiunea :

char *Nume[ ]={"Diana","Georgiana","Raluca"};

Ce se va afisa cu instructiunea : printf("%c", *(Nume[1]+2));

a) “D” b) “G” c) “o” d) “Raluca”

Raspuns corect

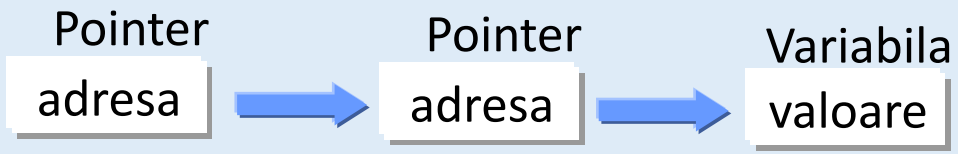
- **c**

1.4. Pointeri la pointeri

Tipuri de indirectare

a. Indirectare simpla

b. Indirectare multipla



1.4. Pointeri la pointeri

Declarare pointer la pointer Format: tip **nume;

EXEMPLE

Ex.1: definirea unui pointer la pointer
int **p; //p este un pointer catre un alt pointer de tipul int
Ex.2: definirea unui pointer la pointer
int a; //defineste o variabila de tip int – contine o o val.numerica intreaga
int *pa;//defineste un pointer la int – contine adresa unei variabile de tip int
int **ppa; //defineste un pointer la un pointer la int – contine adresa unei
//variabile de tip pointer la int
Initializam variabilele:
a=10; pa=&a; ppa=&pa;
Pentru a tipari **adresa variabilei a**:
printf("%x", &a);
printf("%p", pa);
printf("%p", *ppa);
Atunci urmatoarele instructiuni tiparesc **valoarea variabilei a** =10:
printf("%d", a);
printf("%d", *pa);
printf("%d", **ppa);

1.4. Pointeri la pointeri

Declarare pointer la pointer

EXEMPLE

Ex.3:

```
#include <stdio.h>
int main()
{int x,*p,**q;
x=10;
p=&x;
q=&p;
printf("x=%d\n", x);
printf("*p este pointer la variabila x\n");
printf("**q este pointer la variabila pointer *p\n");
printf("adresa lui x prin p: %p\n", p);
printf("adresa lui x prin q: %p\n", *q);
printf("valoarea lui x prin p:%d", *p);
printf("\nvaloarea lui x prin q:%d", **q);
return 0;}
```

```
x=10
*p este pointer la variabila x
**q este pointer la variabila pointer *p
adresa lui x prin p: 0060FF08
adresa lui x prin q: 0060FF08
valoarea lui x prin p:10
valoarea lui x prin q:10
```

TEST kahoot

Pentru login, introduceti codul afisat pe ecran, in browser la adresa:

http://kahoot.it

1.4. Pointeri si functii

I. Pointeri ca argumente de functii

EXEMPLU

Ex.1: Functia swap1() ar trebui sa faca schimbarea dintre x si y

```
#include <stdio.h>
void swap1(int a, int b);
int main()
{  int x,y;
   printf( "x="); scanf ("%d", &x);
   printf( "y="); scanf ("%d", &y);
   swap1(x,y);
   printf("x=%d, y=%d", x,y);
   return 0;}
void swap1(int a, int b)
{  int temp;
    temp=a; a=b;
    b=temp;}
```

```
x=3
y=5
x=3, y=5
```

- In C transferul parametrilor este efectuat implicit prin valoare = nu modifica parametri de apel (swap1()).
- In C daca se doreste modificarea unei variabile parametru atunci trebuie transmisa functiei adresa variabilei => **apel prin pointeri** Ex. functia swap2()
- In C++ schimbarea continutului a 2 variabile se poate face cu **apel prin referinta.** Ex.functia swap3().

1.4. Pointeri si functii

I. Pointeri ca argumente de functii

EXEMPLE

Ex.2: in C, parametri tip pointeri

```
#include <stdio.h>
void swap2(int *pa, int *pb);
int main()
{  int x,y;
   printf( "x="); scanf ("%d", &x);
   printf( "y="); scanf ("%d", &y);
   swap2(&x,&y);
   printf("x=%d, y=%d", x,y);
   return 0;}
void swap2(int *pa, int *pb)
{  int temp;
   temp=*pa;
   *pa=*pb;
   *pb=temp; }
```

```
x=3
y=5
x=5, y=3
```

Ex.3: in C++ , parametri tip referinta

```
#include <stdio.h>
void swap3(int &a, int &b);
int main()
{  int x,y;
   printf( "x="); scanf ("%d", &x);
   printf( "y="); scanf ("%d", &y);
   swap3(x,y);
   printf("x=%d, y=%d", x,y);
   return 0;}
void swap3(int &a, int &b)
{  int temp;
   temp=a;   a=b;
   b=temp;}
```

```
x=3
y=5
x=5, y=3
```

1.4. Pointeri si functii

II. Tablouri de pointeri ca argumente de functii

EXEMPLE

Ex.6. Un tablou de pointeri se poate transmite unei functii, in mod similar cu transmiterea numelui tabloului fara indici.

Functia afis_tab() afiseaza un sir de nr intregi prin intermediul unui tablou de pointeri.

```
void afis_tab(int *q[])
{int i;
for (i=0;i<10;i++)
      printf("%d", *q[i]);
}

// q nu este un pointer catre intregi! q este un tablou de pointeri catre intregi
```

1.4. Pointeri si functii

III. Functie care returneaza un pointer

Exista o diferenta intre functia care returneaza un pointer si un pointer la o functie

EXEMPLE

Ex.1: Definirea prototipului unei **functii** function cu un parametru de tip double care returneaza un pointer la double:

```
double * function(double);
```

Ex.2: Definirea unei variabile de tipul **pointer la o functie** cu un parametru de tip double si care returneaza o valoare tip double:

```
double (*pf)(double);
```

1.4. Pointeri si functii

IV. Pointeri la functii

DEFINIRE

Sintaxa: tip (*nume_pointer) (lista_param_formali); //declarare pointer

Initializare: nume_pointer=nume_functie; //initializare pointer

unde tip = tipul de baza al pointerului , poate fi void daca nu returneaza nici o valoare, sau unul din tipurile char, int, float, double, etc.

nume_pointer = numele pointerului la functie

nume_functie = numele functiei= pointer constant (similar pointer la tablouri)

Apel functie: (*nume_pointer) (lista_param_efectivi);

- **functie ≠ variabila**, dar are o localizare (adresa) in memorie ce poate fi atribuita unui pointer
- adresa unei functii se obtine utilizand numele functiei fara paranteze si argumente (vezi Ex.)

1.4. Pointeri si functii

IV. Pointeri la functii

EXEMPLE

Ex.4: Pointer la o functie de afisare de tip void

```
#include <stdio.h>
void fun(int a)
{    printf("Valoarea lui a este %d\n", a);}

int main()
{    // p este pointer la functia fun()
void (*p)(int) = &fun; //declarare si initializare pointer la functie
     // echivalent  cu  void (*p)(int); p = &fun;
     // apel functie fun prin pointer
(*p)(10);
return 0;}
```

```
Valoarea lui a este 10
```

1.4. Pointeri si functii

IV. Pointeri la functii

EXEMPLE

Ex.5: Afisarea valorilor functiilor sin(x), cos(x) pentru Pi/6, utilizand pointerii *ps, *pc la aceste functii

```
#include <stdio.h>
#include <math.h>
#define PI 3.1415926
int main()
{double (*ps)(double), (*pc)(double);
ps=sin; pc=cos;
printf("\n\t ps=adresa functiei sin = %p", ps);
printf("\n\t pc=adresa functiei cos = %p", pc);
printf("\n\napelul functiilor trigonometrice\nprin pointeri la functii\n");
printf("\n\t\t sin(pi/6) = %lf", (*ps)(PI/6) );
printf("\n\t\t cos(pi/6) = %lf", (*pc)(PI/6) );return 0;}
```

```
        adresa functiei sin = 00401C00
        adresa functiei cos = 00401C08

apelul functiilor trigonometrice
prin pointeri la functii

                sin(pi/6) = 0.500000
                cos(pi/6) = 0.866025
```

Cum se calculeaza tangenta (pi/6) ?

```
printf("\n\t\t tan(pi/6) = %lf", (*ps)(PI/6)/(*pc)(PI/6) );
```

sau: `double (*pt)(double);pt=tan; printf("\n\t\t tan(pi/6)=%lf", (*pt)(PI/6) );`

1.4. Pointeri si functii

IV. Pointeri la functii

Cum au fost create functiile trigonometrice predefinite? Metoda : descompunere in serii Taylor

$$sin(x) = \frac{x}{1!} - \frac{x^3}{3!} + \frac{x^5}{5!} - \ldots = \sum_{i=1}^{\infty} (-1)^{i+1} \frac{x^{2i+1}}{(2i+1)!}$$

$$\cos(x) = \sum_{k=0}^{\infty} \frac{(-1)^k x^{2k}}{(2k)!}$$

$$\sin^2(x) = -\sum_{k=1}^{\infty} \frac{(-1)^k 2^{-1+2k} x^{2k}}{(2k)!}$$

$$\tan^{-1}(x) = \sum_{k=0}^{\infty} \frac{(-1)^k x^{1+2k}}{1+2k} \quad \text{for } |x| < 1$$

$$\log(1+x) = -\sum_{k=1}^{\infty} \frac{(-1)^k x^k}{k} \quad \text{for } |x| < 1$$

$$e^x = \sum_{k=0}^{\infty} \frac{x^k}{k!}$$

$$\cosh(x) = \sum_{k=0}^{\infty} \frac{x^{2k}}{(2k)!}$$

1.4. Pointeri si functii

IV. Pointeri la functii

Descompunere in serii Taylor: sin(x), x radiani

EXEMPLE

Ex.6: Exemplu de calcul al functiei sin() si comparare cu sin() din biblioteca math.h

```
#include <stdio.h>
#include <math.h>
double fact(int j)
{double t; int k; t=1;
for (k=1;k<=j;k++)t=t*k;return t;}
int main()
{int i,n; double x,fx; fx=x;
double (*p)(double, double); p=pow;
double (*q) (int); q=fact;
printf("x=");scanf("%lf",&x);
printf("n=");scanf("%d",&n);
for (i=1;i<=n;i++)
fx+=(*p)(-1,i)*(*p)(x,(2*i+1))/(*q)(2*i+1); //fx+=pow(-1,i)*pow(x,(2*i+1))/fact(2*i+1);
printf("Val functiei sin(x) calculata\ncu descompunerea in serie este:\n");
printf("sin1(%lf)=%.20lf\n", x, fx);
printf("\nval functiei sin(x) calculata\ncu functia sin din <math.h> este:\n");
printf("sin2(%lf)=%.20lf\n", x ,sin(x));return 0;}
```

$$sin(x) = \frac{x}{1!} - \frac{x^3}{3!} + \frac{x^5}{5!} - \ldots = \sum_{i=1}^{\infty} (-1)^{i+1} \frac{x^{2i+1}}{(2i+1)!}$$

```
x=0.5
n=5
Val functiei sin(x) calculata
cu descompunerea in serie este:
sin1(0.500000)=0.47942553860418341000

val functiei sin(x) calculata
cu functia sin din <math.h> este:
sin2(0.500000)=0.47942553860420301000
```

Obs: Daca x este in radiani =ok, daca e in grade trebuie convertit =>

1.4. Pointeri si functii

IV. Pointeri la functii

Descompunere in serii Taylor: sin(x), x grade

EXEMPLE

Ex.7: Exemplu de calcul al functiei sin() si comparare cu sin() din biblioteca math.h cu **x citit in grade si convertit in radiani**

```
#include <stdio.h>
#include <math.h>
double fact(int j) {double t; int k; t=1;
for (k=1;k<=j;k++)t=t*k;return t;}
int main()
{int i,n; double x,fx;
double (*p)(double, double); p=pow;
double (*q) (int); q=fact;
printf("introduceti x in grade=");scanf("%lf",&x);
printf("n=");scanf("%d",&n);
x = x * (3.142/180.0);printf("x in radiani=%lf\n", x);fx=x;
for (i=1;i<=n;i++)
    fx+=(*p)(-1,i)*(*p)(x,(2*i+1))/(*q)(2*i+1);
printf("Val functiei sin(x) calculata\ncu descompunerea in serie este:\n");
printf("sin1(%lf)=%.20lf\n", x, fx);
printf("\nval functiei sin(x) calculata\ncu functia sin din <math.h> este:\n");
printf("sin2(%lf)=%.20lf\n", x ,sin(x));return 0;}
```

```
introduceti x in grade=30
n=20
x in radiani=0.523667
Val functiei sin(x) calculata
cu descompunerea in serie este:
sin1(0.523667)=0.50005879423755217000

val functiei sin(x) calculata
cu functia sin din <math.h> este:
sin2(0.523667)=0.50005879423755206000
```

1.4. Pointeri si functii

V. Tablou de pointeri la functii

DEFINIRE

Sintaxa declarare cu initializare:

tip (*nume_tabpointeri[]) (lista_param_formali) = {lista nume functii};

unde tip = tipul de baza al pointerului , poate fi void daca nu returneaza nici o valoare, sau unul din tipurile char, int, float, double

nume_tabpointeri = **numele tabloului de pointeri la functii**

lista nume_functii = **numele functiilor**

Apel functie: (*nume_tabpointeri[i]) (lista_param_efectivi);

- **Functiile din lista trebuie sa aiba acelasi numar de parametri si acelasi tip returnat**

1.4. Pointeri si functii

V. Tablou de pointeri la functii

EXEMPLE

Ex.4: Afisarea valorilor functiilor sin(x), cos(x) , exp(x), log(x) pentru anumite valori ale argumentelor, utilizand tabloul de pointeri *tabfun la aceste functii

```
#define pi 3.1415926
…
double (*tabfun[])(double) = { sin, cos, exp, log, sqrt } ,y,z;
//pentru a calcula e^3 = exp(3) :
y = (*tabfun[2])(3);
//pentru  a calcula sin (pi/6) :
z= (*tabfun[0])(pi/6);
```

*tabfun[0] *tabfun[1] *tabfun[2] *tabfun[3] *tabfun[4]

Cum calculam
- sqrt(4)?
- cos (2*pi/3)?
- log(10)?
- pow(10,24)?

```
z1= (*tabfun[4])(4);
z2= (*tabfun[1])(2*pi/3);
 z3= (*tabfun[3])(10);
```

Nu se poate cu acest tablou de pointeri pow() are 2 parametri

TEST

Care este forma corecta de declarare si initializare a unui pointer la functia sqrt()?

a) double *p(double); p=sqrt(double);

b) double *sqrt(double); sqrt=*p;

c) double (*p) (sqrt)(double); p=sqrt;

d) double (*p)(double); p=sqrt;

Raspuns corect

- **d)**

TEST

Daca double x, (*p)(double);p=exp;

Cum atribui x=e^5 prin pointer la functie?

a) x=(*p) exp(5);

b) x=*p(5);

c) x=(*p)(5);

d) x=exp(*p);

Raspuns corect

- **c)**

1.4. Pointeri si functii

Exemple functii predefinite citire /scriere caractere in C

Funcţie	Efect	Biblioteca
getchar()	permite citirea cu ecou a caracterelor introduse de la tastatură până la acţionarea tastei "Enter";	**<stdio.h>**
getche()	citeşte, un caracter şi are ecou (afişarea imediată a caracterului pe monitor); nu necesită caracter de linie noua; nu este definit de standardul ANSI C, dar este o extindere uzuală;	**<conio.h>**
getch()	citeşte, un caracter fără ecou; nu necesită caracter de linie noua; nu este definit de standardul ANSI C;	**<conio.h>**
putchar()	scrie un caracter pe ecran şi Returnează codul caracterului afişat sau -1 în caz de eroare; Format: putchar(expr), unde expr reprezintă codul ASCII al caracterului ce se afiseaza;	**<stdio.h>**
putch()	scrie un caracter pe ecran; Format: putch(expr), unde expr reprezintă codul ASCII al caracterului ce se afiseaza;	**<conio.h>**
gets()	citeşte un şir de la tastatură;	**<stdio.h>**
puts()	scrie un şir pe ecran;	**<stdio.h>**

1.4. Pointeri si functii

Exemple functii predefinite pentru siruri de caractere in C

Funcţie	Efect	Biblioteca
isalnum(c)	Returnează true (adevarat) dacă c este litera sau cifra	<ctype.h>
isalpha(c)	Returnează true (adevarat) dacă c este litera	<ctype.h>
isblank(c)	Returnează true (adevarat) dacă c este blank sau tab	<ctype.h>
isdigit(c)	Returnează true (adevarat) dacă c este cifra	<ctype.h>
islower(c)	Returnează true (adevarat) dacă c este litera mica	<ctype.h>
isupper(c)	Returnează true (adevarat) dacă c este litera majuscula	<ctype.h>
isspace(c)	Returnează true (adevarat) dacă c este caracter alb (spatiu,tab, carriage return, linie noua)	<ctype.h>
tolower(c)	Returnează litera mica corespunzatoare lui c dacă există, altfel returnează caracterul neschimbat	<stdlib.h> <ctype.h>
toupper(c)	Returnează litera majuscula corespunzatoare lui c dacă există, altfel returnează caracterul neschimbat	<stdlib.h> <ctype.h>

1.4. Pointeri si functii

Exemplu utilizare isdigit(), strlen()

EXEMPLE

Ex.: Afisarea nr. de cifre dintr-un text utilizand functia isdigit() care returneaza o valoare !=0 daca parametrul este o cifra

```
#include <stdio.h>
#include <string.h>
#include <ctype.h>
int main()
{
int i,k=0;
char *p="sir de 4 cuvinte";
for (i=0;i<strlen(p);i++)
{if ( isdigit(p[i]) ) k++;}
printf("%d",k); return 0;}
```

Afiseaza : 1

Ce afiseaza daca char *p="1 sir de 40 cuvinte " ?

Afiseaza : 3

Se afiseaza nr de cifre

1.4. Pointeri si functii

Exemplu utilizare pointer la isdigit()

EXEMPLE

Ex.4: Afisarea nr de cifre dintr-un text utilizand un pointer la isdigit(), si pointer la strlen()

```
#include <stdio.h>
#include <string.h>
#include <ctype.h>
int main()
{int i,k=0;
int (*pisdg) (int); pisdg=isdigit;
size_t (*pstrl)(const char *); pstrl=strlen;
char *p="1 sir de 40 cuvinte";
for (i=0;i<(*pstrl)(p);i++) //echivalent i<strlen(p)
{if ((*pisdg)(p[i])) k++;}  //echivalent isdigit(p[i]) !=0)
printf("%d",k);
 return 0;}
```

Afiseaza : 3
Se afiseaza nr de cifre

Obs Tipul functiilor:

```
size_t strlen(const char *str);
int isdigit(int c);
```

1.4. Pointeri si functii

Exemple functii predefinite siruri de caractere in C : <string.h>

Funcţie	Efect
strcpy(s1,s2)	copiază s1 în s2
strcat(s1,s2)	concatenează s2 la sfirsitul lui s1
strlen(s1)	returneaza lungimea lui s1
strcmp(s1,s2)	comparare şiruri; returnează 0 dacă s1 şi s2 sunt identice, <0 dacă s1<s2, >0 dacă s1>s2
strchr(s1,ch)	Se caută caracterul ch în şirul s1; funcţia returnează un pointer la prima apariţie a lui ch în s1
strstr(s1,s2)	Se caută un şir s2 în şirul s1; funcţia returneaza un pointer la prima apariţie a lui s2 în s1

1.4. Pointeri si functii

Functii predefinite de conversie in C: <stdlib.h>

Funcţie	Efect
atoi(const char *string)	converteşte un şir de caractere ASCII într-un întreg
atof(const char *string)	converteşte un şir de caractere ASCII intr-un nr. real
itoa(int value, char *string, int base)	converteşte un intreg (int) in sir ascii , din baza 10 în baza de numeratie specificata de base
ltoa(long value, char *string, int base)	converteşte un intreg (long int) in sir ascii din baza 10 în baza de numeratie specificata de base

1.4. Pointeri si functii

Exemple functii predefinite in C: <math.h>

Funcţie	Efect
sin(v)	Returnează valoarea reală aproximativă a sinusului trigonometric a unghiului specificat **v** (double) în radiani.
cos(v)	Returnează valoarea reală aproximativă a cosinusului trigonometric a unghiului specificat **v** (double) în radiani.
tan(v)	Returnează valoarea reală aproximativă a tangentei trigono-metrice a unghiului specificat **v** (double) în radiani.
asin(v)	Returnează valoarea reală aproximativă a arcsin trigonometric a unghiului specificat **v** (double) în radiani.
acos(v)	Returnează valoarea reală aproximativă a arcos trigonometric a unghiului specificat **v** (double) în radiani.
atan(v)	Returnează valoarea reală aproximativă a arctangentei trigonometrice a unghiului specificat **v** (double) în radiani.
atan2(y,x)	Returnează valoarea reală aproximativă a arctangentei trigonometrice a unghiului specificat prin **y/x** (y double, x double) în radiani.
cosh(v)	Returnează valoarea sinusului hiperbolic din **v** (double)
sinh(v)	Returnează valoarea cosinusului hiperbolic din **v** (double)

1.4. Pointeri si functii

Alte functii predefinite din <math.h>

Funcţie	Efect
pow(x,y)	Returnează valoarea ridicarii la puterea **y** a expresiei numerice specificate prin **x. y** poate fi de tipul: int, long, float,double,etc. Rezultatul returnat are acelaşi tip cu tipul **x**.
sqrt(x)	Returnează valoarea rădăcinii pătrate din expresia numerică specificată. Rezultatul returnat are acelaşi tip cu tipul lui **x**.
abs(x)	Returnează valoarea absoluta a expresiei numerice specificate prin **x**. Rezultatul are acelaşi tip ca **x**.
round(x,l)	Valoarea expresiei numerice specificată prin **x** este rotunjita cu precizia specificata de **l** (**l** este de tip int). Rezultatul returnat va avea acelaşi tip ca şi **x**.
rand(seed)	Generează valori numerice reale aleatoare. (se poate specifica o expresie de tip int numita **seed**.)
exp(x)	Returnează valoarea exponentială a expresiei numerice specificate prin x (**x** double).
log(x)	Returnează valoarea logaritmului natural din valoarea expresiei numerice reale specificate prin **x** (**x** de tip double).
log10(x)	Returnează valoarea logaritmului în baza 10 din valoarea expresiei numerice reale specificate prin **x** (**x** de tip double).

TEST kahoot

Pentru login, introduceti codul afisat pe ecran, in browser la adresa:

http://kahoot.it