Tehnici de programare

Alocarea dinamică a memoriei. Pointeri

ovidiu.banias@aut.upt.ro

Cuprins

Pointeri. Utilitate

- Alocare dinamică a memoriei

  Compile time vs. Run time

  Optimizare

- Transmiterea parametrilor prin referință

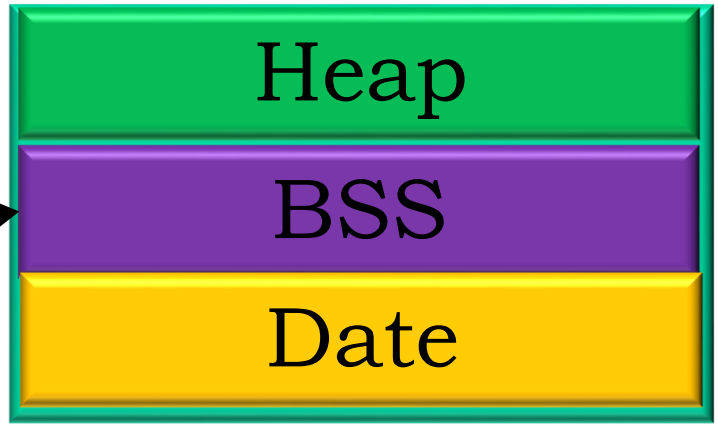
- Vectori și șiruri de caractere

Memoria

Segmentul de STIVĂ

- Variabile locale
- Apeluri de funcții

Segmentul de DATE



- Variabile globale, statice
- Alocare dinamică
- Variabile inițializate/neinițializate
- BSS= block starting with symbol

Segmentul de COD/TEXT

- Conține instrucțiunile executabile

Adrese de memorie

- Adresă = numărul de ordine al unui octet(cuvânt) în memorie
- Compilator pe 16/32/64 de biți
- O variabilă ocupă x octeți succesivi. Adresa primului octet=adresa variabilei

  char → 1 octet

  int → 2 octeți (sau 4)
- adresa unei variabile ≠ valoarea variabilei
- adresă = pointer

Variabile de tip pointer

- Sintaxa C:

```
tip    *numeVariabila
```

  - **tip** : *tip de dată standard* (char, int,...)

    sau *tip de dată compus*(struct, union,..)

  - ***** : operator de dereferențiere (indirectare)

  - **numeVariabila** : nume variabilă

```
tip    *numeVar
char   *adresa1;
int    *adresa2, *adresa3;
float  *adresa4;
```

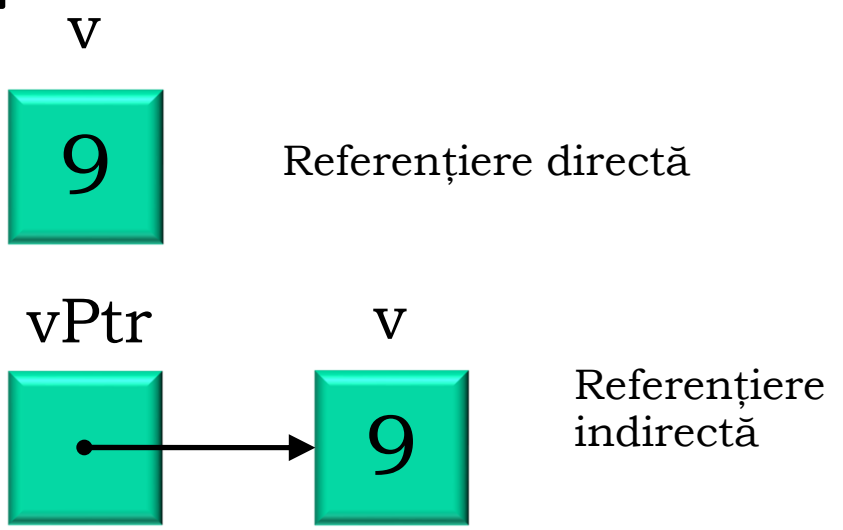
Pointer = variabilă ce are ca valoare o adresă de memorie

O variabilă de tip pointer, pointează către o anumită zonă din memorie (adresează o anumită zonă de memorie)

Variabile de tip pointer

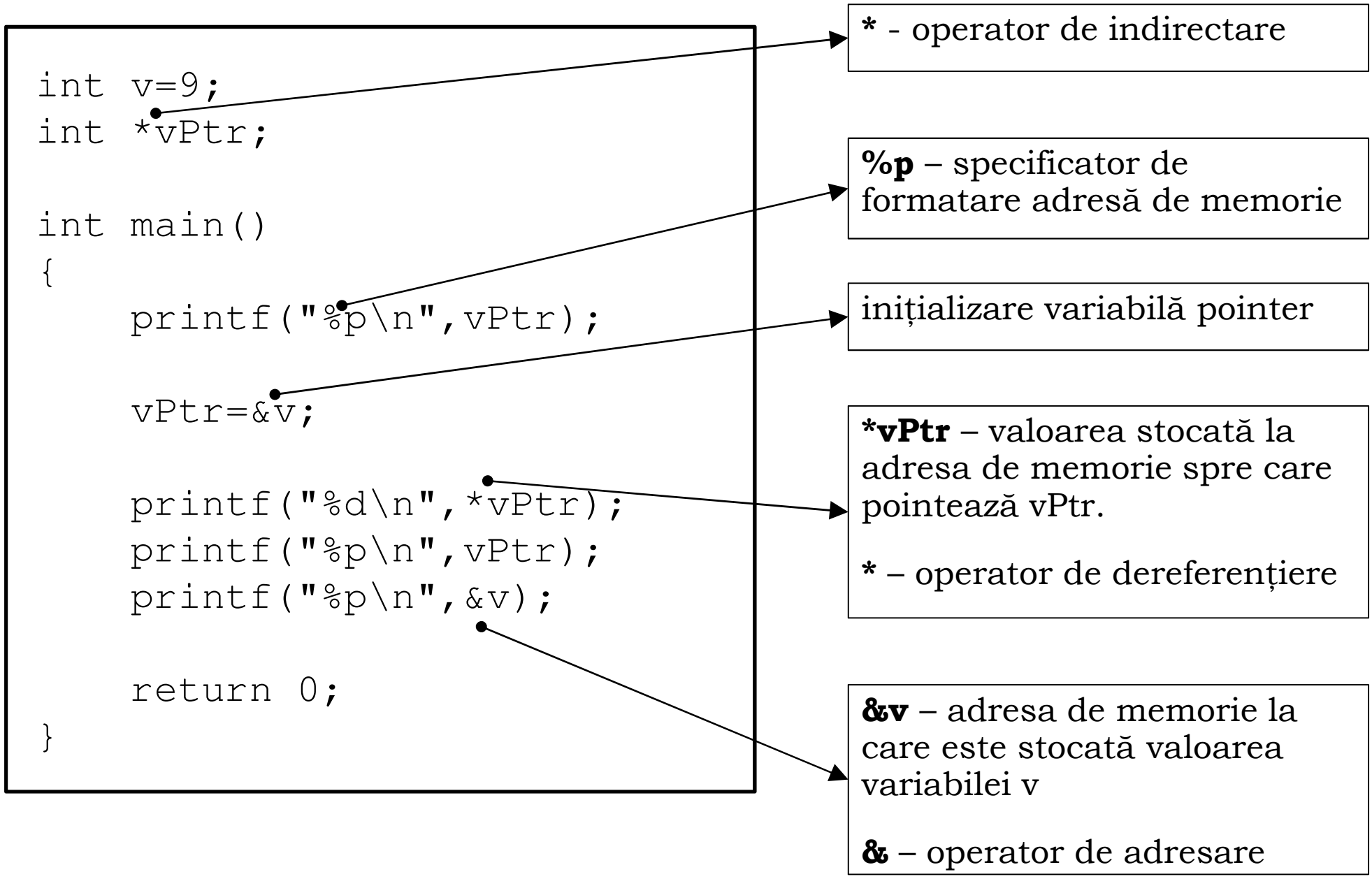
```
int v=7;

int *vPtr;

…

vPtr=&v;

…
```

Pointerul ***vPtr*** pointează către variabila ***v***



- ***&*** - operator de adresare (referențiere)
- ***0, NULL*** – inițializare pointer

Variabile de tip pointer. Dereferențiere



Operații cu pointeri. Exemplificare

```
int v=9;
int *vPtr, *vPtr2;

int main()
{
    vPtr=&v;

    *vPtr = *vPtr + 5; printf("%d\n",*vPtr);

    (*vPtr) ++; printf("%d\n",*vPtr);

    vPtr2=vPtr; (*vPtr) ++; printf("%d\n",*vPtr2);

    printf("%p\n",vPtr);
    printf("%p\n",vPtr2);

    return 0;
}
```

Alocarea dinamică a memoriei

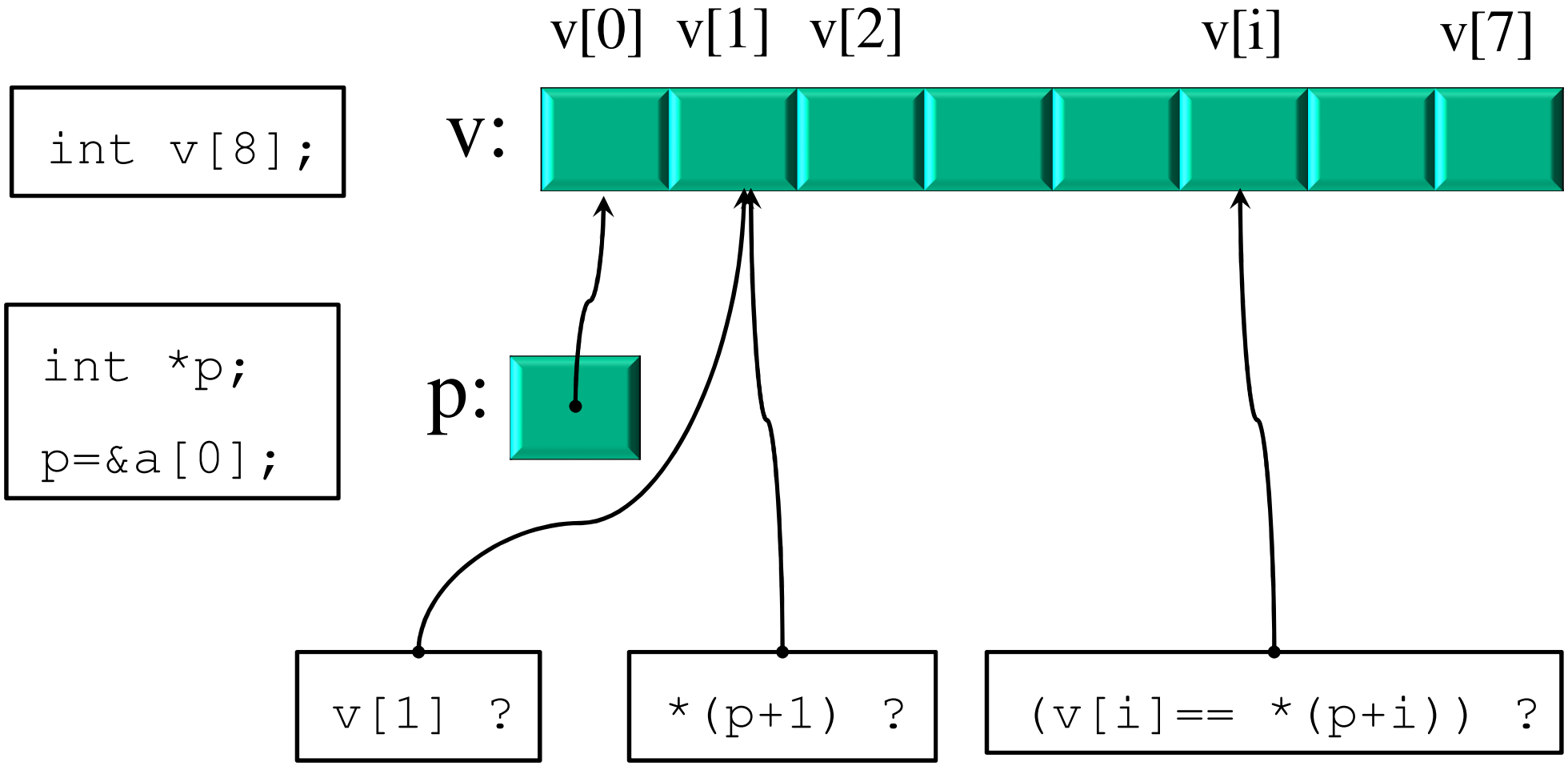
```
#include <stdio.h>
#include <stdlib.h>

#define tip char

tip *vPtr, *vPtr2;

int main()
{
    vPtr=(tip *) malloc(sizeof(tip));
    vPtr2=(tip *) malloc(sizeof(tip));

    printf("%p\n",vPtr);
    printf("%p\n",vPtr2);
    printf("%p\n",vPtr2+1);

    printf("%d\n",(vPtr2-vPtr));
    return 0;
}
```

Transmiterea parametrilor prin referință

```
void Swap1(int p, int q){ // prin valoare
    int k;
    k=p;p=q;q=k;
}
void Swap2(int &p, int &q){ // prin referinta (C++)
    int k;
    k=p;p=q;q=k;
}
void Swap3(int *p, int *q){ // prin referinta
    int k;
    k=*p;*p=*q;*q=k;
}
int main(){
    a=2;b=5;
    Swap1(a,b);printf("Swap1 -> %d %d\n",a,b);
    Swap2(a,b);printf("Swap2 -> %d %d\n",a,b);
    Swap3(&a,&b);printf("Swap3 -> %d %d\n",a,b);
   return 0;
}
```

Vectori și pointeri

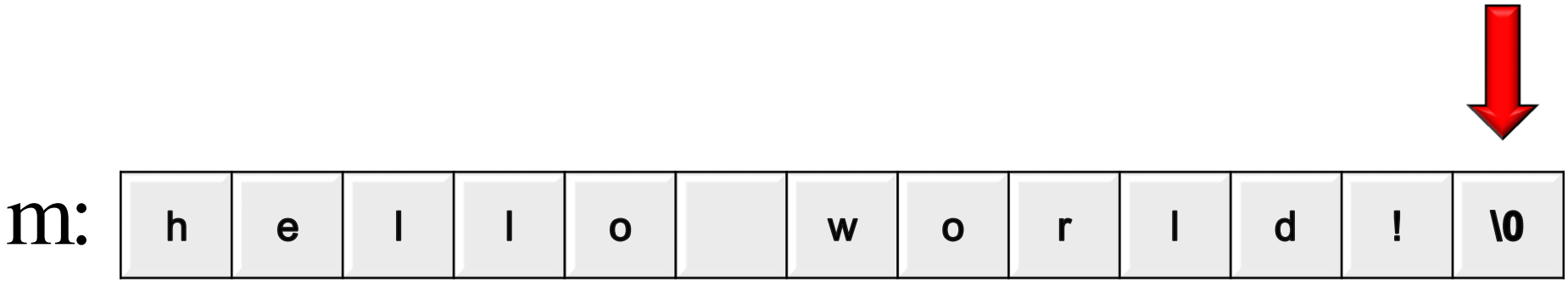
- Strânsă legătură între vectori şi pointeri
- Orice operație cu indecși şi vectori poate fi realizată cu pointeri



Vectori (șiruri de caractere) și pointeri

```
char  m[]="hello world!"; //vector
```

```
char *m="hello world!"; //pointer
```



- + strcpy, strcmp, strcat, strlen,...
- + studiu bibliografic

Vectori multidimensionali şi pointeri

```
char a[3][10]={"Luni","Marti","Miercuri"};
```

SAU

```
char *a[3]={"Luni","Marti","Miercuri"};
```

- avantaje
- dezavantaje
- exemplificare