Дефинисање показивача

Процесорска и RAM меморија се називају оперативне меморије пошто се оне најчешће користе у стандардном раду са промењивима.
Оперативна меморија је низ меморијских локација.
Свака меморијска локација је нумерисана целим позитивним бројевима од 1 до броја пуног капацитета меморије. Бројеви који означавају меморијску локацију називају се адресе.
Подаци који се смештају у меморијске локације заузимају различит број бајтова:
char заузима 1 бајт, int заузима 2 или 4 бајта, float заузима 4 бајта, double заузима 8 бајтова.

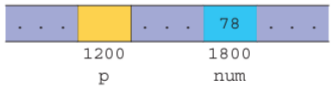
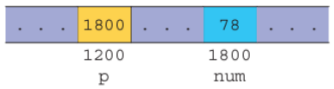
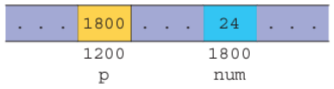
Декларација показивача

Показивач (pointer) је податак у кои се може сместити адреса неке меморијске локације.
Показивачи могу заузимати 2 или 4 бајта меморије.
Ова количина меморије зависи од опсега адреса на појединачном рачунару.
Показивач се дефинише као: tip_pokazivaca * ime_pokazivača
Тип показивача мора бити истог типа као и промењива на чију адресу указује.
Пример декларације једног показивача: int *a; //пример указује да је показивач *a целобројног типа, тј да промењива а садржи меморијску адресу на којој се налази неки целобројни податак.
Пример декларације више показивача: int *a, *b;
Ови примери су идентични: int *x; int* x; int * x; int*x;

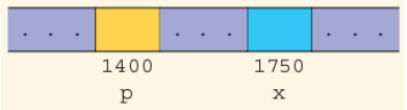
Оператори показивача

Оператор * се назива оператор диференцирања и користи се када треба приступити податку уписаном на адреси на коју указује показивач, тј вредност промењиве на тој адреси.
Оператор & (амперстанд) је унарни оператор и назива се оператор адресе и користи се када треба сазнати на којој меморијској локацији је уписан податак.
Пример:
int x = 100, y; y = x; // x и y су целобројне промењиве, y има вредност 100
int *p; // *p је показивач
p = &x; cout << *p // p добија адресу меморијске локације x, каже се да p указује на x, исписује садржај меморијске локације x пошто показивач *p указује на x
Објашњење: претпостављамо да је p на адреси 1200 и да је num на адреси 1800

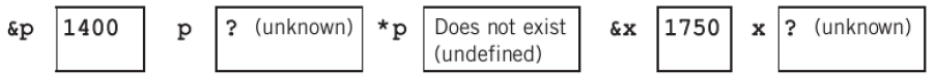
```
        int *p, num;
1       num = 78;
2       p = &num;
3       *p = 24;
```

After Statement	Values of the Variables	Explanation
1		The statement **num = 78;** stores **78** into **num**.
2		The statement **p = #** stores the address of **num**, which is **1800**, into **p**.
3		The statement ***p = 24;** stores **24** into the memory location to which **p** points. Because the value of **p** is **1800**, statement 3 stores **24** into memory location **1800**. Note that the value of **num** is also changed.

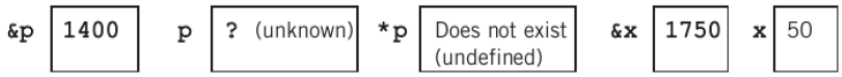
Пример: int *p, x;



после прве линије кода:

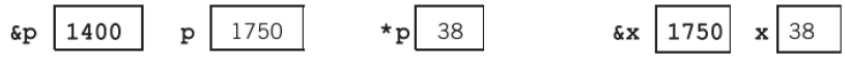


x = 50;



p = &x;

*p = 38;



Пример: Декларисати три промењиве, једну целобројну, једну кратку децималну и једну дугу децималну па одредити њихове адресе.

```
#include<iostream>
using namespace std;

int main()
{
        int x=5, *px=&x;
        float y=2.1, *py=&y;
        double z=1.22, *pz=&z;
        cout << “x ima vrednost ” << x << “ smestenu na adresi ” << px << endl;
        cout << “y ima vrednost ” << y << “ smestenu na adresi ” << py << endl;
        cout << “z ima vrednost ” << z << “ smestenu na adresi ” << pz << endl;
        return 0;
}
```

x ima vrednost 5 smestenu na adresi 0x7fff7892bed4
y ima vrednost 2.1 smestenu na adresi 0x7fff7892bed0
z ima vrednost 1.22 smestenu na adresi 0x7fff7892bec8

Домаћи 13: Целобројној промењивој доделити показивач, па преко показивача променити вредност промењиве. Приказати вредност промењиве, показивача као и колико простора заузимају у меморији.
Домаћи 14: Декл две целобројне промењиве, па одредити њихове адресе и одредити која мем локација је већа.

Иницијализација показивача

Показивачу се не може директно доделити нека бројчана вредност.
Једини изузетак је поинтер са чија је вредност 0.
Могуће је користити и предефинисану константу NULL уместо броја 0.
NULL је дефинисан у неколико хедер фајлова (и у iostream).
int *px = 0;

Аритметичке операције над показивачима

Показивачи су прости целобројни подаци и над њима је могуће вршити следеће аритметичке операције:

- Додељивање вредности једног показивача другом
- Сабирање и одузимање два показивача
- Сабирање и одузимање показивача и целог броја
- Поређење два показивача
- Поређење показивача са нулом

Показивачи и низови

Показивачи се највише примењују код низова. Они показују адресу неког елемента низа. Први елемент низа је у ствари, показивач на низ, јер је само име низа управо адреса првог елемента низа.

Пример: коришћење показивача за приказ следећег елемената низа

```
#include <iostream>
using namespace std;

int main ()
{
	int x[5] = {4, 6, 14, 3, 6};	// inicijalizujemo niz, ime niza „x“ – ukazuje na adresu prvog elementa niza
	int *px;			// deklarisemo pokazivac
	px = x + 1;			// pokazivacu dodeljujemo adresu drugog elementa niza
	cout << "vrednost drugog elementa niza je " << *px << endl;
	return 0;
}
```

Пример: коришћење показивача за приказ свих елемената у низу

```
#include <iostream>
using namespace std;
int main ()
{
	int a[10] = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10};			// inicijalizacija niza
	int *pa;							// deklaracija pokazivaca
	pa = a;								// dodela adrese niza pokazivacu
	for (int i = 0; i < 10; i++)
	{
		cout << "vrednost " << i + 1 << "-tog elementa niza je " // u telu petlje pomeramo pokazivac za po
		<< *(pa + i) << endl;					// jednu lokaciju udesno
	}								// i prikazujemo podatak na toj lokaciji
	return 0;
}
```

Могуће је доделити адресу било ког елемента у низу показивачу на низ (коришћењем &).

Домаћи 15: Приказати све елементе низа a = {2, 0, 35, 10} помоћу показивача.

Домаћи 16: Доделити адресу петог елемента у низу показивачу *px. Низ је a = {3, 6, 10, 230, 35, 1000}.

Пример: Показивачу px доделити адресу петог елемента низа, показивачу py адресу првог елемента низа 1,2,3, 4,5,6,7,8,9,10.

```
#include <iostream>
using namespace std;

int main ()
{
	int a[10] = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10};
	int *px, *py;
	px = &a[0];
	cout << “Vrednost 5-tog elementa niza je ” << *py << endl;
	py = &a[4];
	cout << “Vrednost 1-og elementa niza je ” << *px << endl;
	return 0;
}
```

Као резултат се добија:

Vrednost 5-tog elementa niza je 5

Vrednost 1-og elementa niza je 1

Пример: Заменити места првом и последњем елементу низа a[10]={1,2,3, 4,5,6,7,8,9,10}.

```
#include <iostream>
using namespace std;
int main ()
{
	int a[10] = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10};
	int z, *px, *py;
	py = &a[9];
	px = &a[0];
	cout << “Pre zamene mesta: ” << endl;
	cout << “Vrednost prvog elementa niza je ” << a[0] << endl;
	cout << “Vrednost poslednjeg elementa niza je ” << a[9] << endl;
	z = *px;
	*px = *py;
	*py = z;
	cout << “Posle zamene mesta: ” << endl;
	cout << “Vrednost prvog elementa niza je ” << a[0] << endl;
	cout << “Vrednost poslednjeg elementa niza je ” << a[9] << endl;
	return 0;
}
```

Као резултат се добија:

```
Pre zamene mesta:
Vrednost prvog elementa niza je 1
Vrednost poslednjeg elementa niza je 10
Posle zamene mesta:
Vrednost prvog elementa niza je 10
Vrednost poslednjeg elementa niza je 1
```

<u>Показивачи и функције</u>

Пример: Израчунати у оквиру функције отпорност и снагу потрошача када су познати напон и струја.

```
#include<iostream>
void elektricno_kolo(double, double, double, double);
using namespace std;

int main()
{
	double u, i, P, R;
	cout << “Unesi jacinu struje I: ”;
	cin >> i;
	cout << “Unesi napon U: ”;
	cin >> u;
	elektricno_kolo(u, i, &R, &P);
	cout << “Otpornost potrosaca je: ” << R << endl;
	cout << “Snaga potrosaca je: ” << P << endl;
	return 0;
}
void elektricno_kolo(double napon, double struja, double *pR, double *pP)
{
	*pR = napon / struja;
	*pP = napon * struja;
}
```

Пример: Написати функцију која одређује збир елемената низа.

```
#include<iostream>
int suma(int);
using namespace std;

int main()
{
	int x[10] = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10};
	cout << “Zbir elemenata niza: ” << suma(x) << endl;
	return 0;
}
int suma(int a[])		// niz moze biti parametar funkcije, navodi se tip i ime niza sa []
{
	int S = 0;
	for (int i = 0; i < 10; i++) S += a[i];
	return S;
}
```

Пример: Написати функцију која поставља елементе низа на 0.

```
#include<iostream>
void nula(int);
using namespace std;

int main()
{
	int x[10] = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10};
	nula(x);
	for (int i = 0; i < 10; i++) cout << x[i] << “ ”;
	cout << endl;
	return 0;
}
void nula(int a[])		// niz moze biti parametar funkcije, navodi se tip i ime niza sa []
{
	for (int i = 0; i < 10; i++) a[i] = 0;
}
```