

Pointers in C and C++

Βασικές γνώσεις για τους δείκτες στη C++

Νοέμβριος 2019

Τμήμα Πληροφορικής και Τηλεπικοινωνιών

Πανεπιστήμιο Ιωαννίνων

Γκόγκος Χρήστος

https://github.com/chgogos/ceteiep_dsa/tree/master/appendix_pointers

Απλό παράδειγμα με δείκτη (τελεστές &, *)

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

int main()
{
    int x = 5;
    int *px; // δήλωση χωρίς αρχικοποίηση δείκτη
    px = &x; // αρχικοποίηση δείκτη
    *px = 7; // αποαναφορά (dereference) δείκτη
    printf("x=%i &x=%x\n", x, &x);
    printf("px=%x *px=%i, &px=%x\n", px, *px, &px);
}
```

```
x=7 &x=61fe1c
px=61fe1c *px=7, &px=61fe10
```

Συνηθισμένο λάθος με δείκτη

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

int main()
{
    int *p1;
    int *p2 = NULL;
    int *p3 = nullptr; // C++ guidelines
    printf("p1=%x, p2=%x, p3=%x\n", p1, p2, p3);

    *p1 = 7; // μη προβλέψιμη συμπεριφορά
    printf("bye");
}
```

p1=6, p2=0, p3=0

Δείκτης σε struct

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

struct point {
    int x;
    int y;
};

int main() {
    point a = {3, 2};
    point *p = &a;
    p->x = 4; // ή (*p).x = 4;
    p->y = 5; // ή (*p).y = 5;
}
```

Δείκτης σε union

```
#include <cstdio>
using namespace std;
union my_union {
    int i;
    double d;
    char c;
};

int main() {
    my_union mu;
    my_union *p = &mu;
    printf("sizeof union is %d\n", sizeof(mu));

    p->i = 1;
    printf("%d %.2f %c\n", p->i, p->d, p->c);

    p->d = 3.14;
    printf("%d %.2f %c\n", p->i, p->d, p->c);

    p->c = 'a';
    printf("%d %.2f %c\n", p->i, p->d, p->c);
}
```

```
1 0.00
1374389535 3.14
1374389601 3.14 a
```

Κλήση με αναφορά (call by reference)

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

// κλήση με αναφορά για την παράμετρο y (σε C και C++)
void fun1(int x, int *y) {
    x++;
    (*y)++;
}

// κλήση με αναφορά για την παράμετρο y (C++)
void fun2(int x, int &y) {
    x++;
    y++;
}

int main() {
    int a=5,b=5;
    fun1(a,&b);
    cout << a << " " << b << endl;
    fun2(a,b);
    cout << a << " " << b << endl;
}
```

5 6
5 7

Δείκτες και πίνακες (αριθμητική δεικτών)

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

int main() {
    int a[] = {1, 2, 3, 4, 5};
    int *p;
    p = a; // ή p=&a[0];
    *p = 7; // {7, 2, 3, 4, 5};
    p++;
    *p = 7; // {7, 7, 3, 4, 5};
    p += 2;
    *p = 7; // {7, 7, 3, 7, 5};
    int *p1 = &a[0];
    int *p2 = &a[4];
    cout << p2 - p1 << endl; // εμφανίζει 4
}
```

4

Διαφορά πίνακα και δείκτη

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

int main() {
    int a[] = {1, 2, 3, 4, 5};
    int n = sizeof(a) / sizeof(a[0]);
    int *p = a;
    for(int i=0;i<n;i++) {
        cout << a[i] << "-" << p[i] << "-" << *(p+i) << endl;
    }
    cout << "size of a: " << sizeof(a) << endl;
    cout << "size of p: " << sizeof(p) << endl;
    p++;
    // a++; // error: lvalue required as increment operand
}
```

```
1-1-1
2-2-2
3-3-3
4-4-4
5-5-5
size of a: 20
size of p: 8
```


Δυναμική δέσμευση μνήμης (C)

```
#include <stdio>
#include <stdlib>

int main() {
    int *p = (int *)malloc(sizeof(int));
    *p = 5;
    printf("p=%x *p=%i\n", p, *p);
    free(p);

    p = (int *)calloc(sizeof(int), 1);
    printf("p=%x *p=%i\n", p, *p);
    free(p);

    p = (int *)malloc(sizeof(int)*2);
    p[0]=1;
    p[1]=2;
    printf("p=%x p[0]=%i p[1]=%i\n", p, p[0], p[1]);

    p = (int*)realloc(p,3);
    p[2]=3;
    printf("p=%x p[0]=%i p[1]=%i, p[2]=%i\n", p, p[0], p[1], p[2]);
    free (p);
}
```

```
p=cd1360 *p=5
p=cd1360 *p=0
p=cd1360 p[0]=1 p[1]=2
p=cd1360 p[0]=1 p[1]=2, p[2]=3
```

Δυναμική δέσμευση μνήμης για struct (C)

```
#include <stdlib>
#include <stdio>
using namespace std;

struct point {
    int x;
    int y;
};

int main() {
    point *p = (point *)malloc(sizeof(point));
    p->x = 1;
    p->y = 1;
    printf("%i %i\n", p->x, p->y);
    free(p);
}
```

11

Δυναμική δέσμευση μνήμης (C++)

```
#include <iostream>
#include <cstdio>

int main() {
    int *p = new int;
    *p = 5;
    printf("p=%x *p=%i\n", p, *p);
    delete p;

    p = new int[3];
    printf("p=%x *p[0]=%i *p[1]=%i *p[2]=%i\n", p, p[0], p[1], p[2]);
    delete[] p;

    p = new int[3](); // αρχικοποίηση με 0
    printf("p=%x *p[0]=%i *p[1]=%i *p[2]=%i\n", p, p[0], p[1], p[2]);
    delete[] p;
}
```

```
p=6d1690 *p=5
p=6d1690 *p[0]=7154192 *p[1]=0 *p[2]=7143760
p=6d1690 *p[0]=0 *p[1]=0 *p[2]=0
```

Δυναμική δέσμευση μνήμης για struct (C++)

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

struct point {
    int x;
    int y;
};

int main() {
    point *p = new point;
    p->x = 1;
    p->y = 1;
    cout << p->x << " " << p->y << endl;
    delete p;
}
```

11

Η περίπτωση του *p++

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

int main() {
    int a[] = {10, 20, 30, 40, 50};
    int *p = a;
    int sum = 0;
    for (int i = 0; i < 5; i++) {
        cout << *p << endl;
        sum += *p++;
    }
    cout << "sum=" << sum << endl;
}
```

```
10
20
30
40
50
sum=150
```

Δείκτης προς δυναμικά δεσμευμένο πίνακα εγγραφών

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

struct point {
    int x;
    int y;
};

int main() {
    point *p1 = new point[5];
    p1[0].x = 1;
    p1[0].y = 1;
    cout << "POINTER TO DYNAMIC ARRAY: " << p1[0].x << " " << p1[0].y << endl;
    delete[] p1;
}
```

POINTER TO DYNAMIC ARRAY: 1 1

Αυτόματος πίνακας δεικτών προς εγγραφές

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

struct point {
    int x;
    int y;
};

int main() {
    point *p2[5]; // automatic array of pointers
    for (int i = 0; i < 5; i++) {
        p2[i] = new point;
    }
    p2[0]->x = 1;
    p2[0]->y = 1;
    cout << "AUTOMATIC ARRAY OF POINTERS: " << p2[0]->x << " " << p2[0]->y << endl;
    for (int i = 0; i < 5; i++) {
        delete p2[i];
    }
}
```

AUTOMATIC ARRAY OF POINTERS: 1 1

Δυναμικός πίνακας δεικτών προς εγγραφές

```
#include <iostream>
using namespace std;
struct point {
    int x;
    int y;
};
int main() {
    point **p3 = new point *[5]; // dynamic array of pointers
    for (int i = 0; i < 5; i++) {
        p3[i] = new point;
    }
    p3[0]->x = 1;
    p3[0]->y = 1;
    cout << "DYNAMIC ARRAY OF POINTERS: " << p3[0]->x << " " << p3[0]->y << endl;
    for (int i = 0; i < 5; i++) {
        delete p3[i];
    }
    delete[] p3;
}
```

DYNAMIC ARRAY OF POINTERS: 1 1

Δείκτες σε void

```
#include <cstdio>
using namespace std;
void fun1(int *p) {
    printf("fun(int*) >>> &p=%x p=%x *p=%i\n", &p, p, *p);
}
void fun2(double *p) {
    printf("fun(double*) >>> &p=%x p=%x *p=%.2f\n", &p, p, *p);
}
int main() {
    int *ip = new int(7);
    double *dp = new double(7.7);
    void *vp = nullptr;
    printf("1. ip=%x dp=%x vp=%x *ip=%i *dp=%.2f\n", ip, dp, vp, *ip, *dp);

    vp = ip;
    *(int *)vp = 1;
    printf("2. ip=%x dp=%x vp=%x *ip=%i *dp=%.2f\n", ip, dp, vp, *ip, *dp);
    fun1((int *)vp);

    vp = dp;
    *(double *)vp = 1.5;
    printf("3. ip=%x dp=%x vp=%x *ip=%i *dp=%.2f\n", ip, dp, vp, *ip, *dp);
    fun2((double *)vp);
}
```

```
1. ip=1b1690 dp=1b16b0 vp=0 *ip=7 *dp=7.70
2. ip=1b1690 dp=1b16b0 vp=1b1690 *ip=1 *dp=7.70
fun(int*) >>> &p=61fdd0 p=1b1690 *p=1
3. ip=1b1690 dp=1b16b0 vp=1b16b0 *ip=1 *dp=1.50
fun(double*) >>> &p=61fdd0 p=1b16b0 *p=1.50
```

Δείκτες συναρτήσεων (function pointers)

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

int fun1(int a, int b){
    return a + b;
}

int fun2(int a, int b){
    return a*b;
}

int fun(int (*fn)(int, int), int a, int b){
    return fn(a,b);
}

int main(){
    cout << fun(fun1, 4,5) << endl;
    cout << fun(fun2, 4,5) << endl;
}
```

9
20

Δείκτες συναρτήσεων

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

void calc(double (*fn)(double), double from, double to, int points) {
    double step = (to - from) / points;
    for (int i = 0; i <= points; i++) {
        double x = from + i * step;
        double y = fn(x);
        cout << "f(" << x << ")=" << y << endl;
    }
}

int main() {
    double PI = 3.1415;
    calc(sin, -PI, PI, 10);
    calc(cos, -PI, PI, 10);
    calc(sqrt, 0, 100, 10);
}
```

```
f(-3.1415)=-9.26536e-05
f(-2.5132)=-0.587845
f(-1.8849)=-0.951074
f(-1.2566)=-0.951045
f(-0.6283)=-0.58777
f(4.44089e-16)=4.44089e-16
f(0.6283)=0.58777
f(1.2566)=0.951045
f(1.8849)=0.951074
f(2.5132)=0.587845
f(3.1415)=9.26536e-05
f(-3.1415)=-1
f(-2.5132)=-0.808973
f(-1.8849)=-0.308964
f(-1.2566)=0.309052
f(-0.6283)=0.809028
f(4.44089e-16)=1
f(0.6283)=0.809028
f(1.2566)=0.309052
f(1.8849)=-0.308964
f(2.5132)=-0.808973
f(3.1415)=-1
f(0)=0
f(10)=3.16228
f(20)=4.47214
f(30)=5.47723
f(40)=6.32456
f(50)=7.07107
f(60)=7.74597
f(70)=8.3666
f(80)=8.94427
f(90)=9.48683
f(100)=10
```

Δείκτες συναρτήσεων

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

int compare_string_by_length(string s1, string s2) {
    return s1.length() - s2.length();
}

int main() {
    vector<string> v = {"chris", "maria", "nikos", "aristea"};
    sort(v.begin(), v.end());
    for (string s : v) {
        cout << s << " ";
    }
    cout << endl;

    // ταξινόμηση με βάση το μήκος λεκτικών (αύξουσα)
    int (*cfs)(string, string) = compare_string_by_length;
    sort(v.begin(), v.end(), cfs);
    for (string s : v) {
        cout << s << " ";
    }
    cout << endl;
}
```

```
aristea chris maria nikos
chris maria nikos aristea
```

```
sort(v.begin(), v.end(), compare_string_by_length);
ή
int (*cfs)(string, string) = compare_string_by_length;
sort(v.begin(), v.end(), cfs);
ή
auto cfs2 = compare_string_by_length;
sort(v.begin(), v.end(), cfs2);
```

Δείκτης σε δείκτη: **

```
#include <iostream>
using namespace std;
struct node {
    int value;
    node *next;
};
void push_front(node **head, int x) {
    node *new_node = new node;
    new_node->value = x;
    new_node->next = *head;
    *head = new_node;
}
void print_list(node *head) {
    node *cur = head;
    while (cur != NULL) {
        cout << cur->value << " "; cur = cur->next;
    }
    cout << endl;
}
int main() {
    node **head = new node*;
    push_front(head, 5); push_front(head, 7); push_front(head, 9);
    print_list(*head);
}
```

975

Αναφορά σε δείκτη: *&

```
#include <iostream>
using namespace std;
struct node {
    int value;
    node *next;
};
void push_front(node *&head, int x) {
    node *new_node = new node;
    new_node->value = x;
    new_node->next = head;
    head = new_node;
}
void print_list(node *head) {
    node *cur = head;
    while (cur != NULL) {
        cout << cur->value << " "; cur = cur->next;
    }
    cout << endl;
}
int main() {
    node *head = NULL;
    push_front(head, 5); push_front(head, 7); push_front(head, 9);
    print_list(head);
}
```

975

Συνάρτηση που επιστρέφει δείκτη

```
#include <iostream>
using namespace std;
struct node {
    int value;
    node *next;
};
node *push_front(node *head, int x) {
    node *new_node = new node;
    new_node->value = x;
    new_node->next = head;
    return new_node;
}
void print_list(node *head) {
    node *cur = head;
    while (cur != NULL)
        cout << cur->value << " "; cur = cur->next;
    cout << endl;
}
int main() {
    node *head = NULL;
    head = push_front(head, 5); head = push_front(head, 7); head = push_front(head, 9);
    print_list(head);
}
```

975

Οι iterators των vectors είναι δείκτες

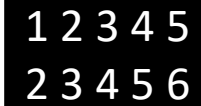
```
#include <iostream>
#include <vector>

using namespace std;

int main() {
    vector<int> v = {1, 2, 3, 4, 5};

    vector<int>::iterator itr = v.begin();
    while (itr != v.end()) {
        cout << *itr << " ";
        (*itr)++;
        itr++;
    }
    cout << endl;

    for (auto it = v.begin(); it != v.end(); it++) {
        cout << *it << " ";
    }
    cout << endl;
}
```



1 2 3 4 5
2 3 4 5 6

Οι iterators των maps είναι δείκτες

```
#include <iostream>
#include <map>
using namespace std;
int main() {
    std::map<std::string, int> months = {
        {"Jan", 31}, {"Feb", 28}, {"Mar", 31}, {"Apr", 30}
    };
    map<string, int>::iterator itr = months.begin();
    while (itr != months.end()) {
        if (itr->second == 31)
            itr = months.erase(itr);
        else
            ++itr;
    }
    for (auto itr = months.cbegin(); itr != months.cend(); ++itr)
        cout << itr->first << " " << itr->second << endl;
}
```

Apr 30
Feb 28

Απλός δείκτης σε int (ο ρόλος του const)

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

int main() {
    int a = 5, b = 10;
    const int c = 20;
    int *p1; // pointer to int

    p1 = &a;
    (*p1)++;
    p1 = &b;
    (*p1)++;
    // p1 = &c; // error: assigning to 'int *' from incompatible type 'const int *'
    printf("pointer to int: a=%i b=%i c=%i *p1=%i\n", a, b, c, *p1);
}
```

pointer to int: a=6 b=11 c=20 *p1=11

Δείκτης σε const int

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

int main()
{
    int a = 5, b = 10;
    const int c = 20;
    int const *p2; // pointer to const int

    p2 = &a;
    p2 = &b;
    p2 = &c;
    printf("pointer to const int: a=%i b=%i c=%i *p2=%i\n", a, b, c, *p2);
    // p2 = &a;
    // (*p2)++; // error: read-only variable is not assignable
}
```

pointer to const int: a=5 b=10 c=20 *p2=20

const δείκτης σε int

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

int main()
{
    int a = 5, b = 10;
    const int c = 20;
    int *const p3 = &a; // const pointer to int

    (*p3)++;
    // p3 = &b; // error: cannot assign to variable 'p3' with const-qualified type 'int *const'
    printf("const pointer to int: a=%i b=%i c=%i *p3=%i\n", a, b, c, *p3);
}
```

const pointer to int: a=6 b=10 c=20 *p3=6

const δείκτης σε const int

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

int main()
{
    int a = 5, b = 10;
    const int c = 20;
    int const *const p4 = &a; // const pointer to const int

    // (*p4)++; // error: read-only variable is not assignable
    // p4 = &b; // error: cannot assign to variable 'p4' with const-qualified type 'const int *const'
    // p4 = &c; // error: cannot assign to variable 'p4' with const-qualified type 'const int *const'
    printf("const pointer to const int: a=%i b=%i c=%i *p4=%i\n", a, b, c, *p4);
}
```

const pointer to const int: a=5 b=10 c=20 *p4=5

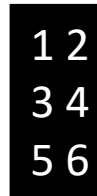
Ο δείκτης this

```
#include <iostream>
using namespace std;
class point {
private:
    int x, y;
public:
    point() {}
    void set_xy(int x, int y) {
        this->x = x;
        this->y = y;
    }
    void display() {
        cout << this << "(x=" << x << ", y=" << y << ")" << endl;
    }
};
int main() {
    point a_point;
    a_point.set_xy(1, 2);
    a_point.display();
}
```

0x62fe18(x=1, y=2)

Δυναμικός δισδιάστατος πίνακας (δείκτης σε δείκτη)

```
#include <iostream>
using namespace std;
void print(int **a, int m, int n) {
    for (int i = 0; i < m; i++) {
        for (int j = 0; j < n; j++)
            cout << a[i][j] << " ";
        cout << endl;
    }
}
int main() {
    int m = 3, n = 2, c = 0;
    int **mat = new int *[m];
    for (int i = 0; i < m; i++) {
        mat[i] = new int[n];
        for (int j = 0; j < n; j++)
            mat[i][j] = ++c;
    }
    print(mat, m, n);
    for (int i = 0; i < m; i++)
        delete[] mat[i];
    delete[] mat;
}
```



```
12
34
56
```