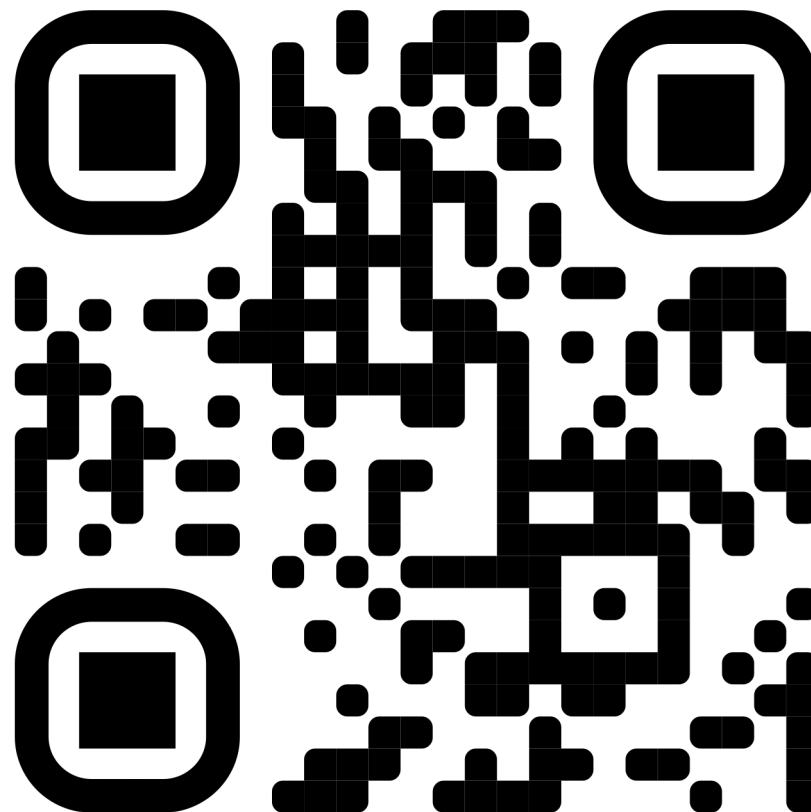


# Указатели-2

## Консоль

### u.to/72D\_Gg

Лекция 3, 19 февраля, 2021



Лектор:

Дмитрий Северов, кафедра информатики 608 КПМ

[dseverov@mail.mipt.ru](mailto:dseverov@mail.mipt.ru)

### Обратная связь : u.to/7Wn7Gg

# Двумерный динамический массив

```
const int n=3, m=5;
```

```
int (*a)[m] = new int[n][m]; // *b[m]  
// a = (int(*)[m])malloc(n*m*4);
```

4 байта

20 байт

```

const int K=2, N=3, M=4;

int main() {
char (*a)[M] = new char[N][M], *pa = (char*)a;
    for(int n=0; n<N; n++)
        for(int m=0; m<M; m++) a[n][m] = 65+M*n+m;
    for(int i=0; i<N*M; i++) cout << *(pa+i) << ((i+1)%M?' ':'\n');
delete a; cout << endl;

char **c, *pc;
c = new char*[N]; for(int n=0; n<N; n++) c[n]=new char[M];
pc = c[0];
for(int n=0; n<N; n++)
    for(int m=0; m<M; m++) c[n][m] = 65+M*n+m;
for(int i=0; i<N*M; i++) cout << *(pc+i) << ((i+1)%M?' ':'\n');
for(int n=0; n<N; n++) { cout << (void*)c[n] << (n==N-1?'\\n':'\\t'); delete c[n]; }
delete c; cout << endl;

int (*b)[N][M] = new int[K][N][M];
for(int k=0; k<K; k++)
    for(int n=0; n<N; n++)
        for(int m=0; m<M; m++)
            cout << (b[k][n][m] = M*N*k + M*n + m) << (m==M-1?(n==N-1?"\\n\\n":"\\n"):"\\t");

return 0;
}

```

```

A B C D
E F G H
I J K L

A B C D
A A A A
A A A A
0x4f0f50      0x4f1240      0x4f1260

0      1      2      3
4      5      6      7
8      9      10     11

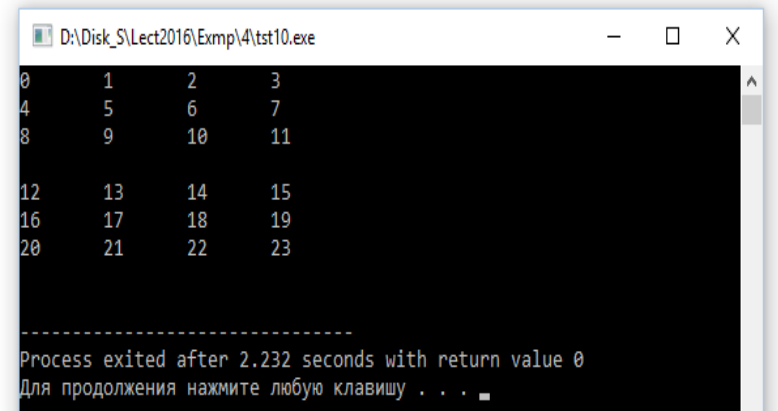
12     13     14     15
16     17     18     19
20     21     22     23

```

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

```
int main() {
    const int K=2, N=3, M=4;
    int (*b)[N][M] = (int(*)[N][M])calloc(K*N*M, sizeof(int)), k, n, m;
    for(k=0; k<K; k++)
        for(n=0; n<N; n++)
            for(m=0; m<M; m++)
                printf("%d%s", (b[k][n][m] = M*N*k + M*n + m), (m==M-1?(n==N-1?"\n\n":"\n"):"\t" ));

    return 0;
}
```



```
D:\Disk_S\Lect2016\Exmp\4\tst10.exe
0      1      2      3
4      5      6      7
8      9      10     11

12     13     14     15
16     17     18     19
20     21     22     23

-----
Process exited after 2.232 seconds with return value 0
Для продолжения нажмите любую клавишу . . .
```

# Операции Инкремента и Декремента

1. `*++Ptr // *(Ptr += 1)` – перевод указания на следующий элемент с использованием его значения при вычислении выражения
2. `--*Ptr // *Ptr -= 1` – уменьшение текущего элемента на единицу с использованием этого значения в вычислении выражения
3. `*Ptr++ // (t=Ptr, Ptr+=1, *t)` – использование значения текущего элемента и перевод указания на следующий элемент
4. `(*Ptr)-- // (t=*Ptr, *Ptr-=1, t)` – использование в вычислении текущего значения элемента с последующим его уменьшением на единицу

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

```
int f(int a) {  
    printf("f::a=%d\n",a);  
    return a;  
}
```

```
int main() {  
    int *a = (int[]){10,20,30}; int b;  
    b = f(*++a) + f(--*a) + f(*a--) + f((*a)++);  
    printf("b=%d a[]= %d %d %d\n",b,a[0],a[1],a[2]);  
  
    return 0;  
}
```

D:\A1\S\errors\tst3.exe

```
f::a=20  
f::a=19  
f::a=19  
f::a=10  
b=68 a[]= 11 19 30
```

-----  
Process exited after 0.07714 seconds with return value 0  
Для продолжения нажмите любую клавишу . . . \_

Журнал компиляции

Отладка

Результаты поиска



mpilation results...

```





#include <iostream>
using namespace std;

int f(int a) {
    cout << "f::a=" << a << endl;
    return a;
}

int main() {
    int a[]={10,20,30},b, *Pa=&a[0];
    b = f(++Pa) + f(--Pa) + f(*Pa++) + f((*Pa)--);
    cout << "b=" << b << " a[]=" << a[0] << ' ' << a[1] << ' ' << a[2] << endl;

    return 0;
}

```

 Журнал компиляции
  Отладка
  Результаты поиска
 

mpilation results...

```

D:\A1\S\errors\tst3.exe
f::a=20
f::a=19
f::a=19
f::a=30
b=88 a[]=10 19 29

```

## Пример 2: операции со строками

- Вычисление длины строки (`strlen(a)`)\*)

```
char *b=a; while(*b++); return b-a-1;
```

- Копирование (`strcpy(a,b)`)

```
while(*a++ = *b++);
```

- Сравнение (`strcmp(a,b)`)

```
while(*a++ == *b++) if(!*(a-1)) return 0;  
return *(a-1) - *(b-1);
```

- Конкатенация (`strcat(a,b)`)

```
char a[N]={..., '\0'}, *p=a;  
while(*p)p++; while(*p++=*b++);
```

\*) `#include <string.h>` - библиотечные функции Си



# Основные функции ввода/вывода

- `int getchar(void);`
- `int getche(void);`
- `int getch(void);`
- `int [f]getc(FILE*);`
- `int putchar(int);`
- `int putch(int);`
- `int [f]putc(int, FILE*);`
- `int printf("%s", char*);`
- `int puts(const char*);`
- `int fputs(char*, FILE*);`
- `int scanf("%s", char*);`
- `char* gets(char*);`
- `char* fgets(char*, int, FILE*);`

**stdin**  
**stdout**  
**stderr**

# Вывод на экран и в поток

```
#include <stdio.h>
#include <conio.h>
#include <string.h>

int main() {const char *s1="This is
            output into the stdout stream",
            *s="to the console";
char *p, s2[]="This is output into
            the stdout stream";
//  *s1='a';
while(*s1) putchar(*s1++);
    putchar('\n');
```

```
    p=strstr(s2,"stdout");
    *(p+3)='e'; *(p+5)=*(p+4)='r';
    p=s2;
    while(*p) fputc(*p++,stderr);
    fputc('\n',stderr);

    p=strstr(s2,"into");
    while(*p++=*s++); p=s2;
    while(*p) putchar(*p++);
    putchar('\n');

    return 0;
}
```

# Вывод

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

```
#include <conio.h>
```

```
#include <string.h>
```

```
int main() {const char
```

```
output into the std
```

```
*s="to the console";
```

```
char *p, s2[]="This is output in  
the stdout stream";
```

```
// *s1='a';
```

```
while(*s1) putchar(*s1++);  
putchar('\n');
```

```
*(p+3)='e'; *(p+5)=*(p+4)='r';  
p=s2;
```

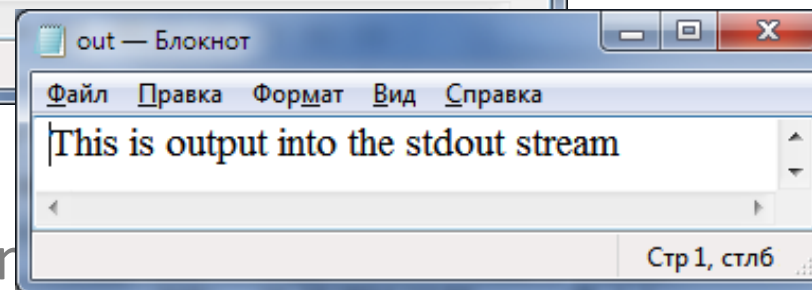
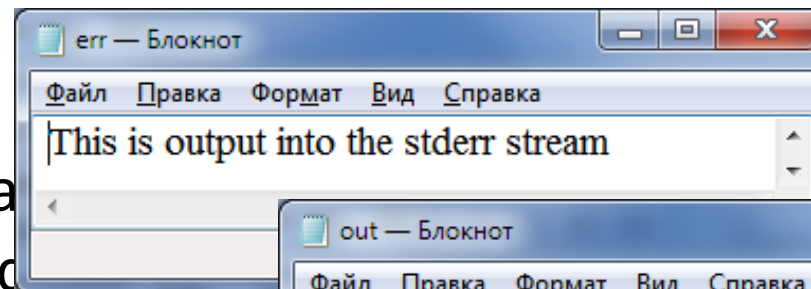
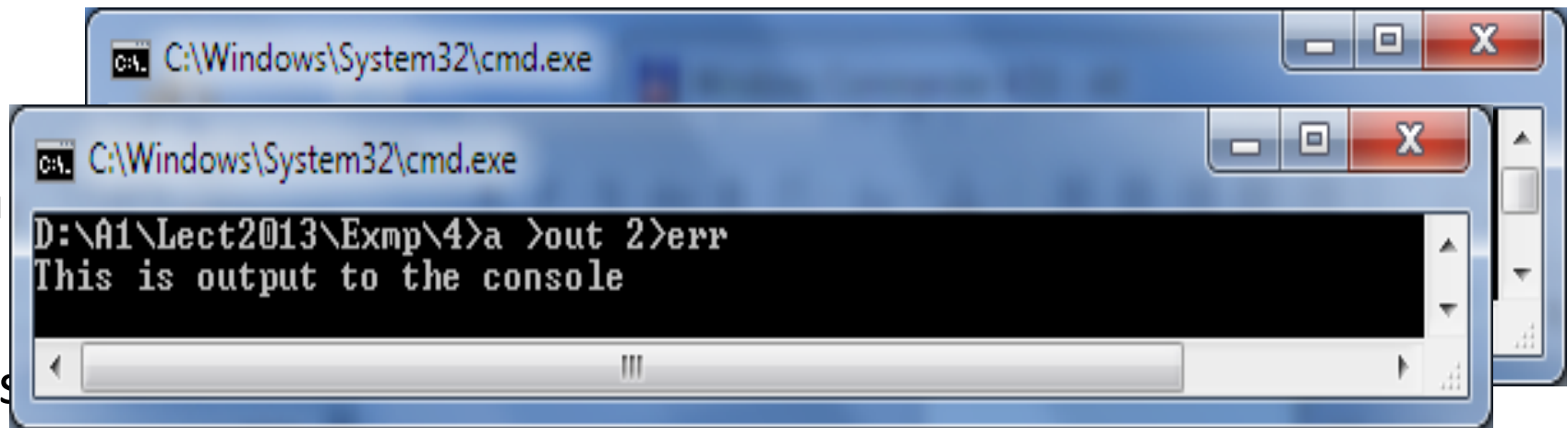
```
putc(*p++,stderr);  
derr);
```

```
s2;
```

```
putch('\n');
```

```
return 0;
```

```
}
```



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

```
char* getstr(char* s, FILE* a) {  
    int b=1, l;  
    char c[10];  
    if (feof(a)) return NULL;  
    s=(char*) calloc(1,1);  
    while (b && fgets(c,10,a)) {  
        if (c[(l=strlen(c))-1]=='\n') { b=0; c[l-1]='\0'; }  
        s = (char*) realloc(s, strlen(s)+(b?l:l-1)+1);  
        strcat(s,c); }  
    return s;  
}
```

```
int main() {  
    char* s;  
    while (s=getstr(s,stdin)) {  
        puts(s); printf(" %d\n", strlen(s));  
        free(s); }  
    return 0;  
}
```



```

#include <iostream>
using namespace std;

istream& operator>> (istream& b, char* &a) {
    int l=0,la=0;
    const int n=10;
    char s[n]; s[n-1]='\0'; *(a = new char)='\0';
    if(b.peek()==EOF) return b;
    while(b.get(s[l]),s[l++]!='\n')
        if(l==n-1) { l=0; a=(char*)realloc(a,la+=n); strcat(a,s); }
    if(l>1) { s[l-1]='\0'; a=(char*)realloc(a,la+l); strcat(a,s); }
    return b;
}

int main() {
    char *s;
    while(cin >> s) {
        cout << s << ' ' << strlen(s) << endl;
        delete s; }
    return 0;
}

```

```

#include <iostream>
using namespace std;

istream& operator>> (istream& b, char* &a) {
    int l=0,la=0;
    const int n=10;
    char s[n]; s[n-1]='\0'; *(a = new char)='\0';
    if(b.peek()==EOF) return b;

```

```

qwqwqwqwqwqw qwqwqwqw qwqwqwqwqw qwqwqwqwqw qwqwqwqwqw
qwqwqwqwqwqw qwqwqwqw qwqwqwqwqw qwqwqwqwqw qwqwqwqwqw 51
asasas          dddddd
asasas          dddddd 21
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa 63

```

```

0
^Z

```

```

        cout << s << " " << strlen(s) << endl;
        delete s; }
    return 0;
}

```

## Пример 3: потеря быстродействия

- Для заданных строк `s1` и `s2` найти номер символа, с которого начинается вхождение строки `s2` в строку `s1`.
- `int strncmp(char* s1, char* s2, int n)` – производит сравнение не более `n` символов из двух строк и возвращает значение 0 – только в случае если строки совпадают.
- `int main(int argn, char* argv[ ])`
  - `argn` – количество слов в командной строке
  - `argv` – параметры, переданные программе



```

#include <stdio.h>
#include <string.h>

unsigned long long start, end;
unsigned long long access_counter() { asm("rdtsc"); }

int main(int argc, char *argv[]) {
int i,l1,l2;
    if(argc<3) { printf("Error parameter list\n"); return 1; }
    start=access_counter();
    for(i=0; i<strlen(argv[1]); i++)
        if(!strncmp(argv[1]+i,argv[2],strlen(argv[2])))
            printf("%s %s %d\n",argv[1],argv[2],i);
    end=access_counter();
    printf("%u\n", (unsigned) (end-start));
    start=access_counter();
    l1=strlen(argv[1]); l2=strlen(argv[2]);
    for(i=0; i<l1; i++)
        if(!strncmp(argv[1]+i,argv[2],l2))
            printf("%s %s %d\n",argv[1],argv[2],i);
    end=access_counter();
    printf("%u\n", (unsigned) (end-start));
    return 0;
}

```

D:\A1\Lect2013\Exmp\4\tst4.exe

```

12345678901234567890*12*1234567890 *12* 20
335447
12345678901234567890*12*1234567890 *12* 20
121051

```

## «ЧТО ПОЗВОЛЕНО ЮПИТЕРУ ...»

```
#include <stdio.h>
#define N ...
struct { int a[N];
        } a,b;
int main() {
    ...
    a=b;    //разрешено
    ...
    return 0; }
```

```
#include <stdio.h>
#define N ...
int a[N],b[N];
int main() {
    ...
    a=b;    // ошибка
    ...
    return 0; }
```

```
#include <iostream>

const int N=30;
struct { int a[N]; } a,b;

int main() {
    for(int i=0; i<N; i++) { b.a[i]=i; a.a[i]=0; }
    a=b;
    for(int i=0; i<N; i++) std::cout << a.a[i] << ' '; std::cout << std::endl;

    return 0;
}
```

d:\A1\Lect2013\Exmp\4\tst9.exe

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

## Пример 4: односвязный список

- Определение формата узла списка

```
struct node {  
    My_type item;  
    struct node *next;  
};
```

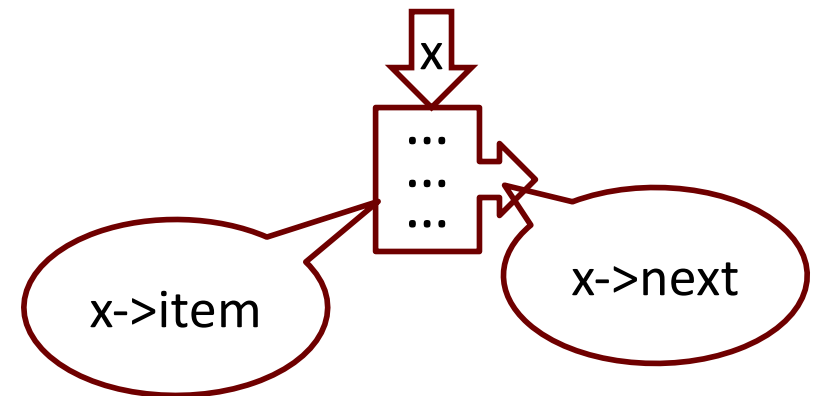
```
typedef struct node *link;
```

- Выделение памяти для узла СПИСКА

```
link x = (link)malloc(sizeof(struct node));
```

- Инициализация элементов узла

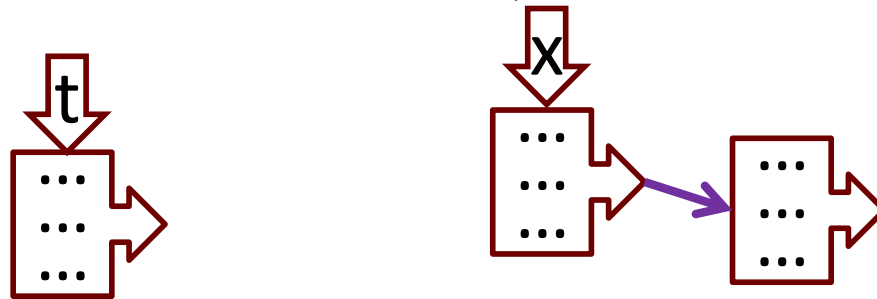
```
(*x).item = ... , x->next = NULL;
```



# Объединение элементов в список (1)

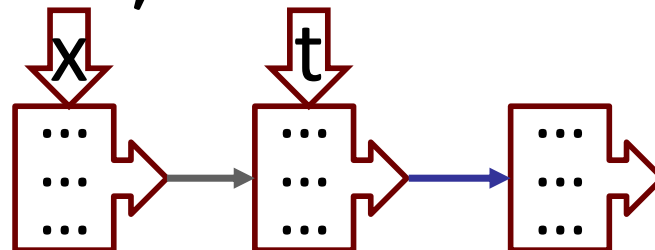
## ■ Вставка узла $t$ за узлом $x$

$t \rightarrow \text{next} = x \rightarrow \text{next}, x \rightarrow \text{next} = t;$



## ■ Удаление узла $t$ за узлом $x$

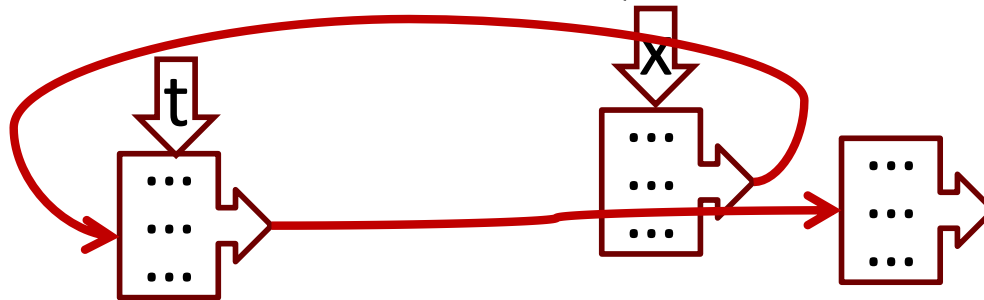
$t = x \rightarrow \text{next}; x \rightarrow \text{next} = t \rightarrow \text{next}; \text{free}(t);$



# Объединение элементов в список (1)

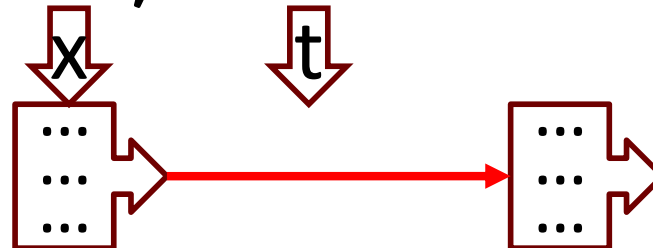
- Вставка узла  $t$  за текущим узлом  $x$

$t \rightarrow \text{next} = x \rightarrow \text{next}, x \rightarrow \text{next} = t;$



- Удаление узла, следующего за  $x$

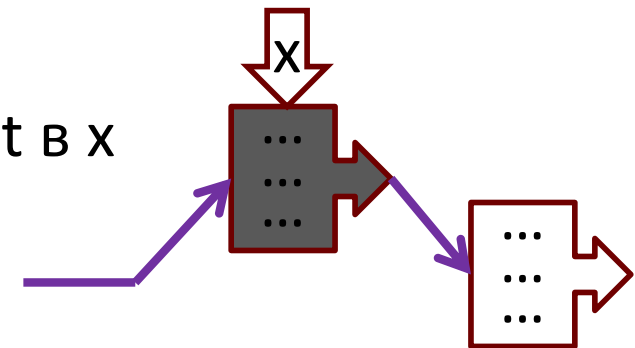
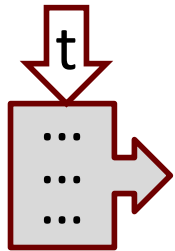
$t = x \rightarrow \text{next}, x \rightarrow \text{next} = t \rightarrow \text{next}, \text{free}(t);$



# Объединение элементов в список (2)

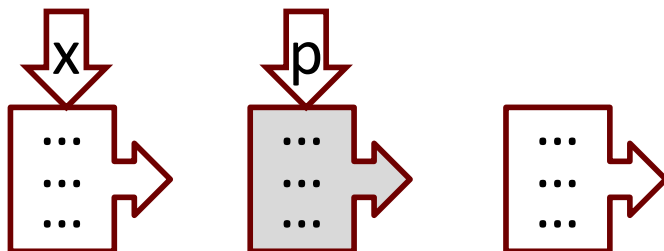
- Вставка узла  $t$  перед текущим узлом  $x$  со сменой текущего

```
link g = new node;    // новый узел g
* $g$  = * $x$ ;            //  $g$  дубль  $x$ , перед  $y$ 
 $x \rightarrow \text{next} = g$ ;    //  $g$  после  $x$ 
 $x \rightarrow \text{item} = t \rightarrow \text{item};$  // содержимое  $t$  в  $x$ 
```



- Удаление узла  $x$  со сменой текущего

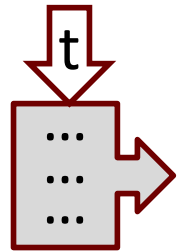
```
link  $p = x \rightarrow \text{next};$     // к удалению
* $x = *x \rightarrow \text{next};$     // перенос значения
delete  $p;$                 // удаление
```



# Объединение элементов в список (2)

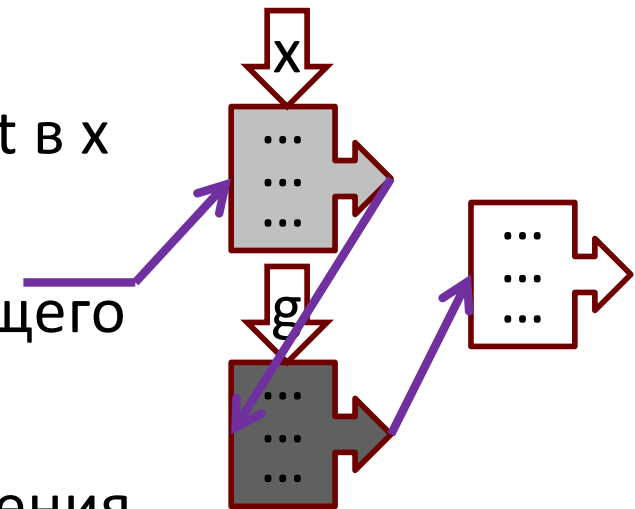
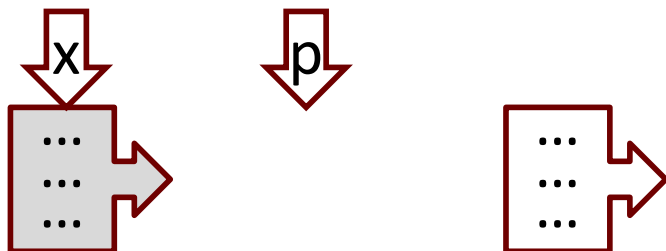
- Вставка узла  $t$  перед текущим узлом  $x$  со сменой текущего

```
link g = new node;    // новый узел g
* $g$  = * $x$ ;            //  $g$  дубль  $x$ , перед  $y$ 
 $x \rightarrow \text{next} = g$ ;    //  $g$  после  $x$ 
 $x \rightarrow \text{item} = t \rightarrow \text{item};$  // содержимое  $t$  в  $x$ 
```



- Удаление текущего узла  $x$  со сменой текущего

```
link  $p = x \rightarrow \text{next};$     // к удалению
* $x = *x \rightarrow \text{next};$     // перенос значения
delete  $p;$                 // удаление
```





```

#include <stdio.h>
#include <malloc.h>
#define new(x) (x*)malloc(sizeof(x))

typedef struct node *link;    // связка - указатель на узел
typedef struct node { int Num; link Next; } item;    // узел или элемент списка

int main() {
    int n,m;
    link top, p;    // вершина списка и безунок

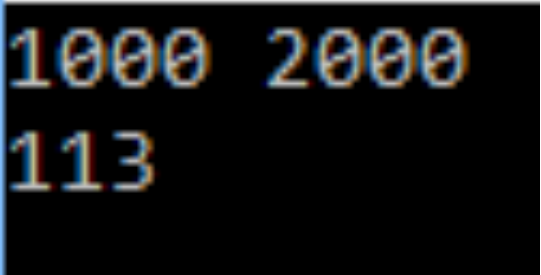
    scanf("%d%d",&n,&m);
    top = p = new(item);
    for(int i=1;i<n;i++) { p->Num = i; p = (p->Next = new(item)); }
    p->Num = n; p = (p->Next = top);

    for (int k=m%n;n>1;k=m%--n)
        if(k==1) *p = *p->Next;
        else { if(!k) k=n;
                for(int i=1;i<k-1;i++) p=p->Next;
                p = (p->Next = p->Next->Next); }

    printf("%d\n",p->Num);

    return 0;
}

```



```

1000 2000
113

```