Структуры данных. Массивы указателей

Структура данных (СД) - совокупность взаимосвязанных переменных и их значений:

- СД – целостное представление физической или программной (логической) сущности
- **Соглашения по данным** – правила соответствия связей и значений в различных переменных СД
- Функции, изменяющие СД, должны сохранять соглашения по данным

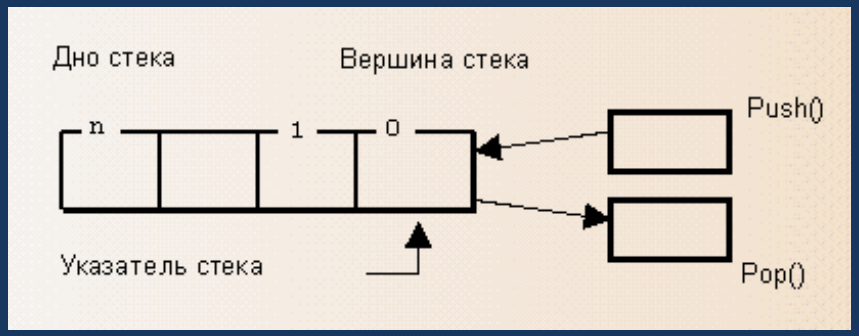
Физический – реальный, имеющий место на самом деле

Логический – образное представление (метафора), создаваемая программными средствами, логическое представление – интерфейс, создающий представление, отличное от реального.

Структуры данных. Массивы указателей

Пример СД: стек - последовательность, включение и исключение элементов в которую производится только с одного конца:

- Последовательность записи и извлечения взаимно обратны
  PUSH (A), PUSH(B), PUSH(C), POP()->C, POP()->B, POP()->A
- Моделирует принцип вложенности:
  - Вложенность конструкций при синтаксическом анализе – стековый автомат
  - Вложенный вызов функций main – A – B – return – A – return – main, стек – история вызовов в обратном порядке
  - Вложенные прерывания процессора (Interrupt)
- Представление – область памяти (массив) с указателем (индексом) элемента в вершине стека (последний записанный). **SP – Stack Pointer**
- В аппаратном сегменте стека (SS) – стек «кверху дном»

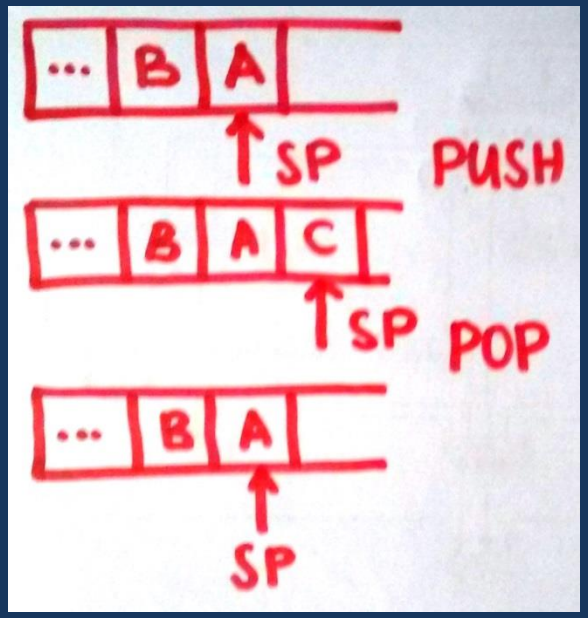


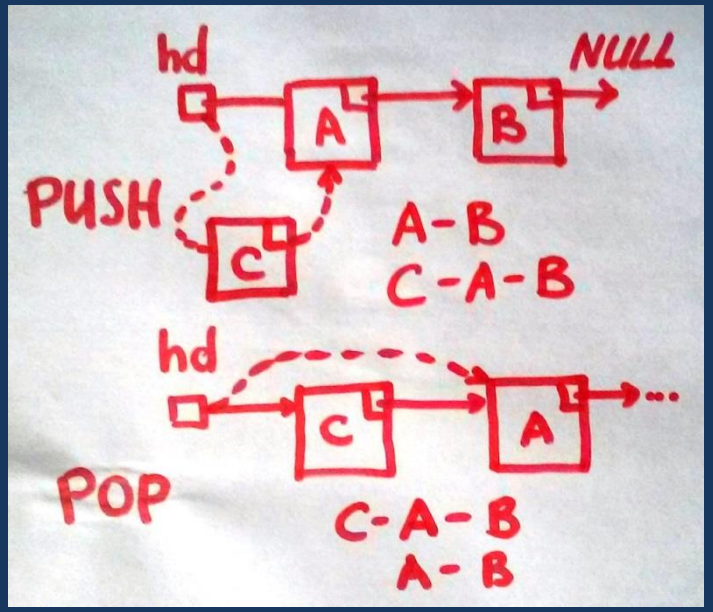

```
int a[100], int sp=-1;

void PUSH(int v){ a[++sp]=v; }

int POP(){ return sp == -1 ? 0 : a[sp++]; }

int GET(int n) { return sp == -1 ? 0 : a[sp-n]; }
```

Структуры данных. Массивы указателей

Физическая СД, логическая СД

Переменные: массив (размерность = const)
Логическая СД: последовательность, стек, очередь
Физическая СД: последовательность в массиве, стек в массиве, стек на односвязном списке





Линейные структуры данных

Последовательность: линейная логическая структура данных
Стек – последовательность с добавлением/извлечением данных с одного конца
Очередь – последовательность с добавлением/извлечением данных с разных концов

Логический номер (ЛН) - номер элемента в порядке обхода СД

Последовательность на массиве – трудоемкость операций:

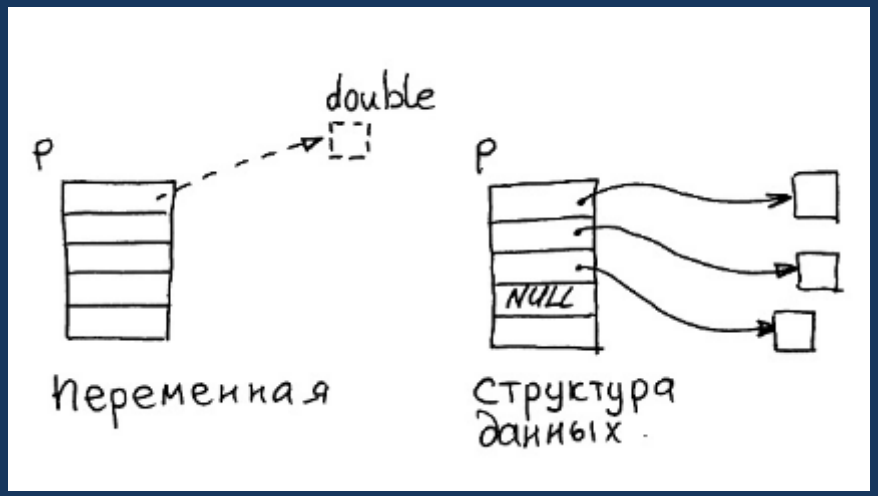
- Append - O(1)
- Insert - **O(N)**
- Remove - **O(N)**
- Get - O(1) – **прямой доступ**
- Search - O(N) – неупорядоченная
- Search - $O(\log_2(N))$ – упорядоченная
- С учетом перераспределение памяти: Append - O(1)…**O(N)**

Массив указателей. Синтаксис

Массив указателей – структура данных:

- МУ – переменная
- Указуемые переменные (массивы)
- Значения указателей (ссылки)

Каждая компонента может быть определена статически и динамически



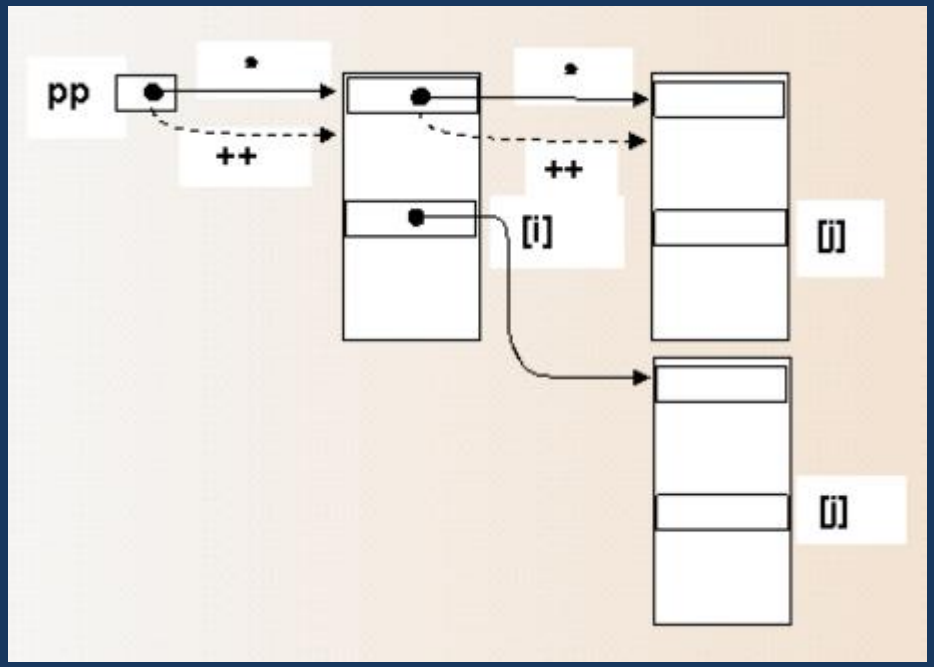
Используемые ТД:

- int *pp[n] - массив указателей
- int **pp - указатель на указатель

Массив указателей. Синтаксис

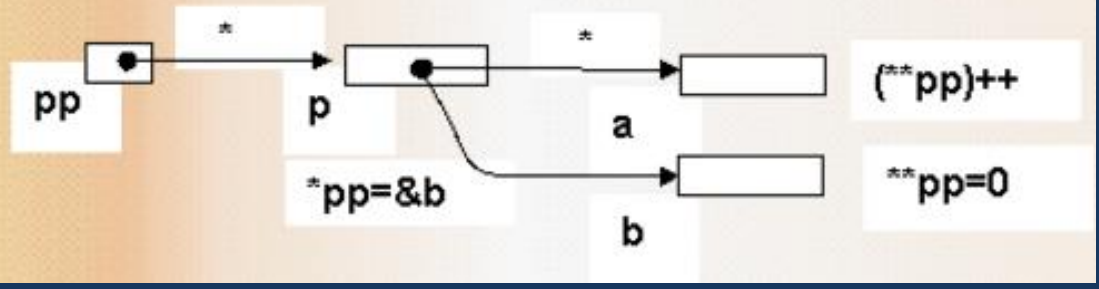
Многообразие вариаций int:** указатель на отдельную переменную и на массив на каждом уровне

Технологический аспект: указуемые объекты могут быть «собственностью» структуры данных, либо МУ ссылается на них как на сторонние объекты.

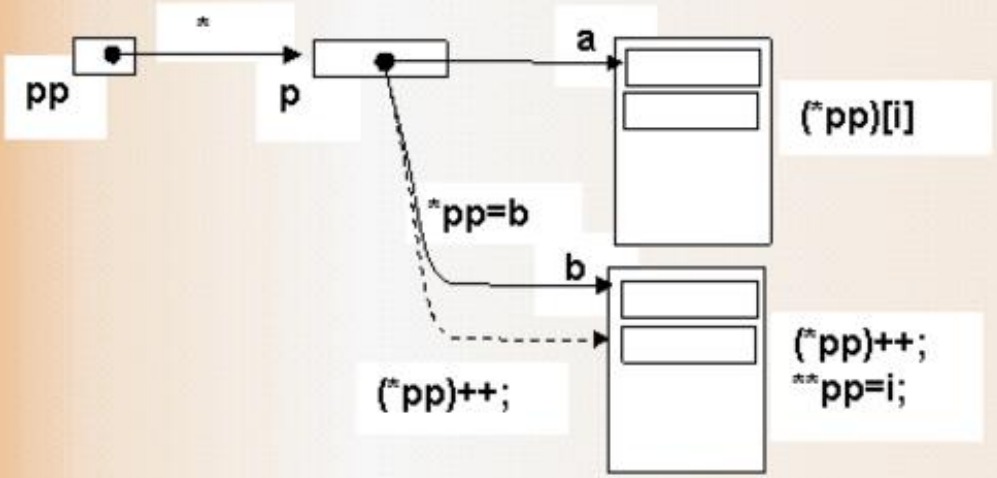


Массив указателей. Синтаксис

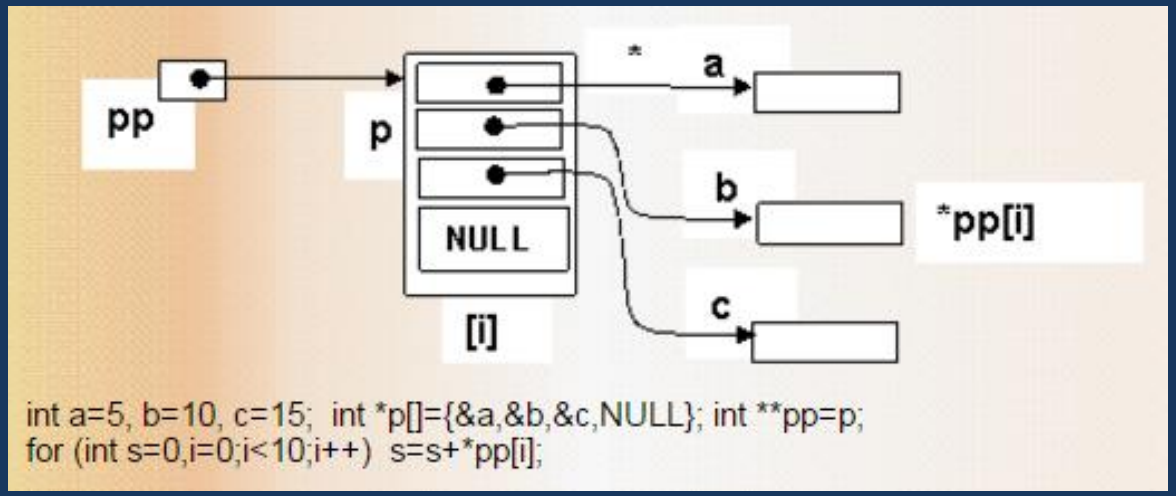
```
int a=5, b=10;  int *p=&a; int **pp=&p;
(**pp)++;  *pp=&b; **pp=0;
```




```
int a[10]=5, b[10]=10;  int *p=a; int **pp=&p;
for (int i=0;i<10;i++) (*pp)[i]=0;
*pp=b;
for (int i=0;i<10;i++) { (*pp)++; **pp=i; }
```

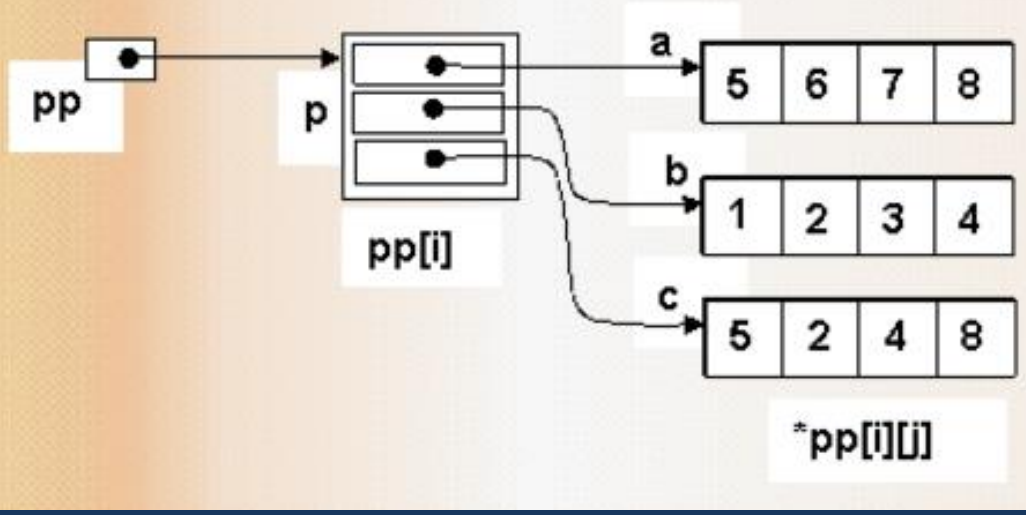


Массив указателей. Синтаксис




```
int a=5, b=10, c=15;  int *p[]={&a,&b,&c,NULL}; int **pp=p;
for (int s=0,i=0;i<10;i++)  s=s+*pp[i];
```

```
int a[]={5,6,7,8}, b[]={1,2,3,4}, c[]={5,2,4,8};  int *p[]={a,b,c}; int **pp=p;
for (int s=0,i=0;i<3;i++)
        for (int j=0;j<4;j++)  s=s+pp[i][j];
```



Статические и динамические МУ

```
int a1,a2,a3, *pd[] = { &a1, &a2, &a3, NULL};
```

```
int d[19], *pd[20];
for (i=0; i<19; i++) pd[i] = &d[i];
pd[i] = NULL;
```

```
int *p, *pd[20];
for (i=0; i<19; i++){ p = new int;  *p = i;  pd[i] = p; }
pd[i] = NULL;
```

```
int **pp, *p;
pp = new int *[20];        // память под массив указателей
for (i=0; i<19; i++){      // из 20 указателей типа int*
        p = new int;
        *p = i;
         pp[i] = p;        // можно pp[i]=new int; *pp[i]=i;
       }
pp[i] = NULL;
```

Массив указателей. Техника работы

- МУ устанавливает собственный **логический порядок** следования элементов
- Упорядоченная последовательность: МУ на исходные элементы + сортировка МУ (сравнение элементов и перестановка указателей)
- МУ как коллекция найденных фрагментов или элементов
- МУ как логическое представление двумерности (МУ на линейные массивы)

```
//-----------------------------------------------------62-01.cpp
//--- Сортировка массива и массива указателей
void sort1 (double d[],int sz){
        int i,k;
        do {
       for ( k=0, i=0; i<sz-1; i++)
                if (d[i] > d[i+1])
                { double c; c = d[i]; d[i] = d[i+1]; d[i+1] = c; k=1;}
        } while (k); }

 void sort2 (double *pd[]){
        int i,k;
       do {
       for ( k=0, i=0; pd[i+1]!=NULL;i++)
                if (*pd[i] > *pd[i+1])              // Сравнение указуемых переменных
                {double *c;                         // Перестановка указателей
                c = pd[i]; pd[i] = pd[i+1];pd[i+1] = c; k = 1; }
        } while (k);}
```

Массив указателей. Техника работы

```
void sortTable(int mode){
        int i,j,k;
        for (i=0;i<nn;i++){
            for(j=k=i; j<nn;j++)
                if (TBL[j].cmpUser(TBL[k],mode)<0)
                    k=j;
            user cc=TBL[i]; TBL[i]=TBL[k]; TBL[k]=cc;
            }
        }
```

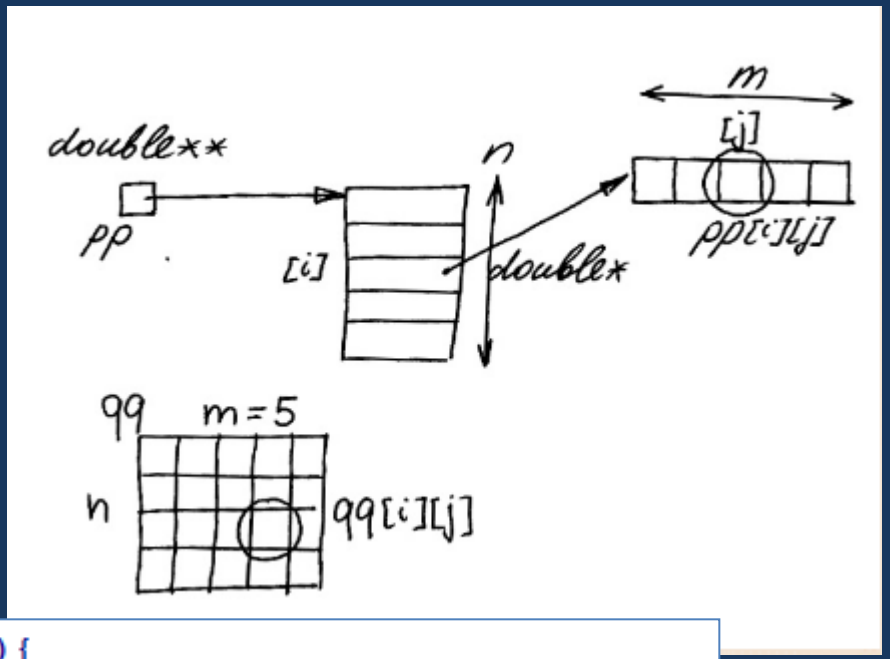
```
user **createSortedTable(int mode){
    user **pp = new user*[nn];
    for (int i=0;i<nn;i++)
        pp[i] = &TBL[i];
    int i,j,k;
    for (i=0;i<nn;i++){
        for(j=k=i; j<nn;j++)
            if (pp[j]->cmpUser(*pp[k],mode)<0)
                k=j;
            user *cc=pp[i]; pp[i]=pp[k]; pp[k]=cc;
            }
        return pp;
        }
};
```

```
        int cmpUser(user &T,int mode){
                switch (mode){
case 0: return strcmp(name,T.name);
case 1: return birth.cmpDate(T.birth);
case 2: return np-T.np;
                }}
        };
```

Двумерный массив. Массивы указателей

- int *pp[] или int **pp - синтаксис определения
- pp[i] – указатель на i-ую строку
- pp[i][j]

Решение для двумерного
Динамического массива




```
double **load(char nm[], int &n, int &m){
    FILE *fd=fopen(nm,"r");
    if (fd==NULL) return NULL;
    fscanf(fd,"%d%d",&n,&m);          // Чтение размерностей
    double **pp=new double*[n];       // Создание ДМУ по первой размерности
    for(int i=0;i<n;i++){
        pp[i]=new double[m];              // Создание линейных ДМ (строк)
        for(int j=0;j<m;j++) fscanf(fd,"%lf",&pp[i][j]);
        }
    fclose(fd);
    return pp;}
```

Двумерный массив. Массивы указателей

- int *pp[] или int **pp - синтаксис определения
- pp[i] – указатель на i-ую строку
- pp[i][j]

Решение для двумерного динамического массива

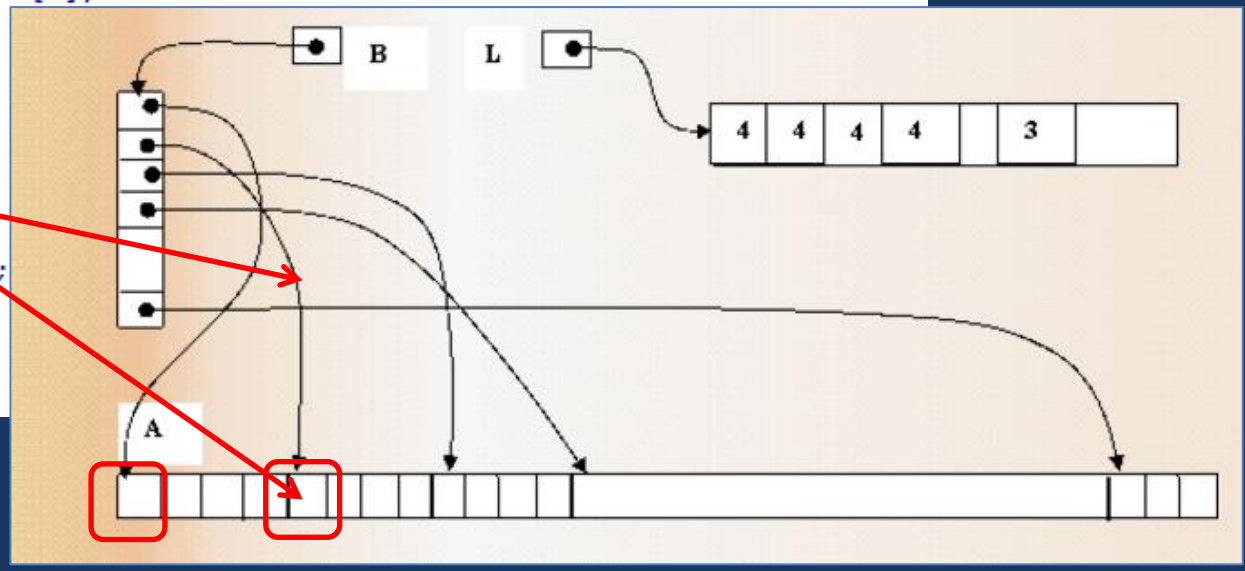
```
3 4
1 2 3 4
5 6 7 8
1 3 2 5
```

```
double sum(double **p , int n, int m ){
double s=0 ;
for (int i=0; i<n; i++)
    for (int j=0; j<m; j++) s+=p[i][j];
return s; }
void destroy(double **pp,int n){
    for (int i=0;i<n;i++) delete []pp[i];
    delete []pp;
    }
void main(){
int n1,m1;
double **pp=load("62-03.txt",n1,m1);
if (pp!=NULL) printf("sum(%d,%d)=%2.0lf\n",n1,m1,sum(pp,n1,m1));
destroy(pp,n1);
}
```

Примеры. Однократное слияние

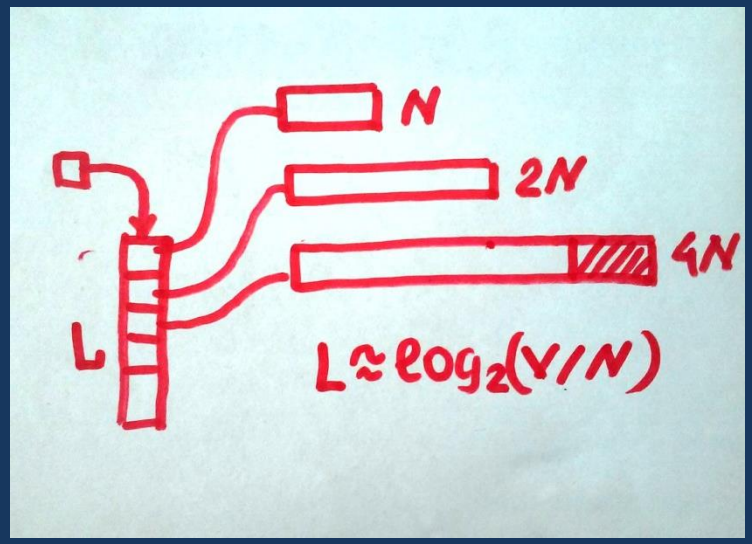
Двумерный массив строится как ДМУ на линейные массивы – интервалы исходного линейного массива + массив длин оставшихся частей

```
void sort(int a[], int n);                  // любая сортировка одномерного массива

void big_sort(int A[], int N){
    int max=A[0],i,j,k,n=sqrt((double)N);
    int **B=new int*[n];
    int *L=new int[n],*C=new int[N];        // массив размерностей частей
    for (i=0; i<n; i++) B[i]=&A[i*n];
    for (i=0; i<n-1; i++) L[i]=n;
    L[n-1]=N-n*(n-1);                       // Размерность последнего массива - остаток
    for (i=0; i<n; i++) sort(B[i],L[i]);// Сортировка частей
    for (i=0; i<N; i++){                    // Слияние
        for (k=-1, j=0; j<n; j++){          // k - индекс строки с минимальным начальным
            if (L[j]==0) continue;          // Пропуск слитых строк
            if (k==-1 || *B[j] < *B[k])
                k=j;
            }
        C[i] = *B[k];
        B[k]++;
        L[k]--;
        }
    for (i=0; i<N; i++) A[i]=C[i];
    delete []B;
    delete []C;
}
```



Примеры. Исключение realloc

```
int add(int value){
    if (cidx>sizep)
        return 0;        // ПЕРЕПОЛНЕНИЕ
    pp[cidx-1][lastn++]=value;
    if (lastn == csize){
        csize*=2;
        lastn=0;
        cidx++;
        if (cidx>sizep)
            return 1;    // ПЕРЕПОЛНЕНИЕ
                total++;
        pp[cidx-1]=new int[csize];
        }
            total++;
    return 1;
    }
int get(int idx){
    if (idx<0 || idx>=total)
        return -1;
    for(int i=0,vv=size0;i<sizep;i++,vv*=2){
        if (idx<vv)
            return pp[i][idx];
        idx -= vv;
        }
    return -1;
    }
};
```




```
#include <stdio.h>
int main(){
    DMURealloc xx;
    xx.create(10,2000000);
    for(int i=0;i<1000000;i++)
        xx.add(i);
    printf("%d\n",xx.get(777));
        getchar();
    return 0;
}
```

Трудоемкость get $O(\log_2(N))$

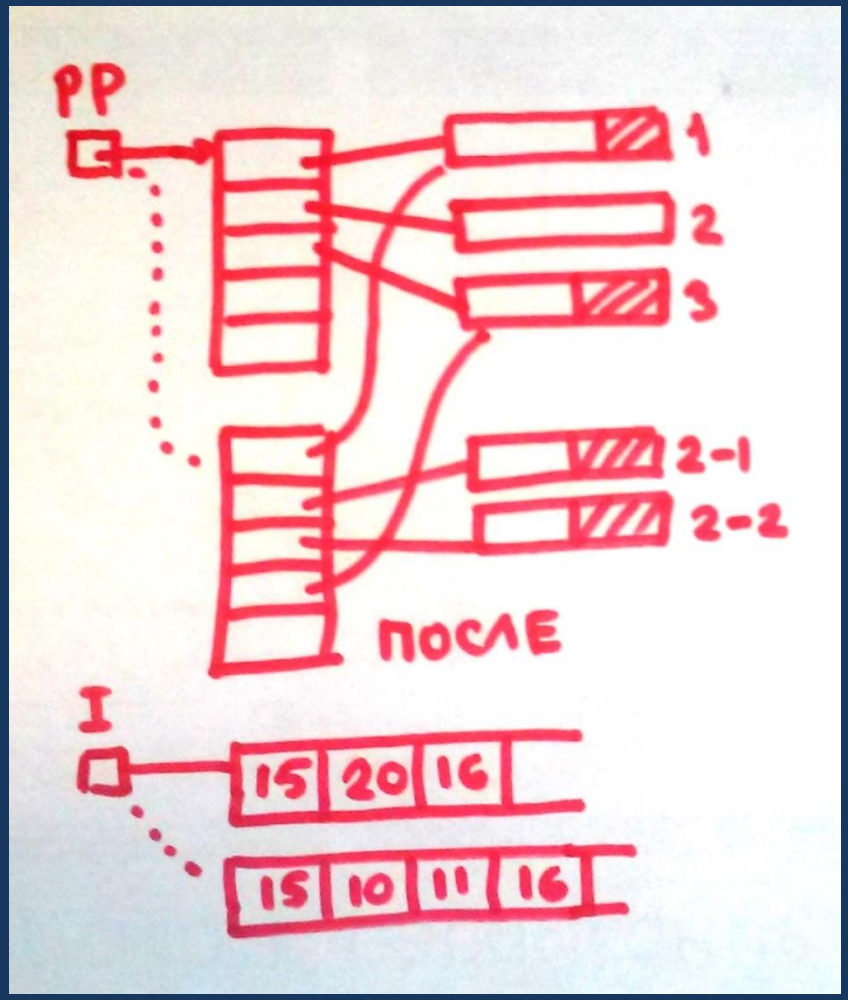
Примеры. ДМУ линейных массивов

Массив указателей на линейные массивы:

- Логическая структура – последовательность
- Физическая структура – массив указателей на массивы фиксированной размерности
- Локальность изменений при вставке/удалении
- Представление линейной структуры квадратной **n=√N**

Трудоемкость:

- Get – **O(√N)**
- Insert/Remove - **O(√N)**



Примеры. ДМУ линейных массивов

```
struct LDS_DMU{
    int size0;       // размерность линейного массива
    int sizep;       // размерность ДМУ
    int cidx;        // текущее кол-во массивов в ДМУ
    int total;       // Общее кол-во элементов
    int **pp;        // ДМУ на линейные массивы
    int *cnt;        // массив - количества элементов в линейных массивах
    void create(int size){
        size0 = sizep = sqrt((double)size);
        cidx=1;                  // Не пустая = 1 линейный массив (соглашение)
        pp = new int*[sizep];    // Создать ДМУ
        pp[0]=new int[size0];    // Создать 1 линейный массив
        cnt = new int[sizep];    // ДМ счетчиков (данных в  лин. массивах)
        cnt[0]=0;
        total=0;                 // Общее количество
        }
    void testDMUsize(){
        if (cidx==sizep){        // Переполнение ДМУ
            sizep*=2;            // Realloc для ДМУ и массива счетчиков
            pp = (int**)realloc(pp,sizep*sizeof(int**));
            cnt = (int*)realloc(cnt,sizep*sizeof(int*));
            }
        }
```

```
void show(){                 // Трассировка построчно
    for(int i=0;i<cidx;i++){
        printf("%3d:",i);
        for(int j=0;j<cnt[i];j++)
            printf("%5d",pp[i][j]);
        puts("");
        }
    }
```

Примеры. ДМУ линейных массивов

```
void add(int value){
    pp[cidx-1][cnt[cidx-1]++]=value;     // Посленим в последний массив
    if (cnt[cidx-1]==size0){             // Переполнение
        testDMUsize();                   // Нужно ли расширять ДМУ
        int kk1 = size0*3/4;             // 75% в старом
        int kk2 = size0 - kk1;           // 25% в новом
        cnt[cidx-1]=kk1;                 // Новые счетчики
        cnt[cidx]=kk2;
        pp[cidx] = new int [size0];      // Новый линейный массив
        for(int i=0;i<kk2;i++)           // Перенести хвост
            pp[cidx][i] = pp[cidx-1][i+kk1];
        cidx++;                          // Кол-во лин. массивов
        }
    total++;
    }
int getIndex(int &idx){                  // Преобразование лог. номера
    if (idx<0 || idx>=total)             // в физический [массив: индекс в массиве]
        return -1;
    for(int i=0;i<cidx;i++){             // Отрбразывать от idx количество
        if (idx<cnt[i])                  // элементов в текущем лин. массиве,
            return i;                    // пока не попадет в тек. диапазон
        idx -= cnt[i];                   // ВТОРОЙ ПАРАМЕТР - ПО ССЫЛКЕ
        }
    return -1;
```

```
int get(int idx){
    int ii = getIndex(idx);
    if (ii==-1)
        return -1;
    return pp[ii][idx];
    }
```

Примеры. ДМУ линейных массивов

```
int insert(int val, int idx){
    int ii = getIndex(idx);                 // Место вставки [ii,idx]
    if (ii==-1)
        return 0;
    for(int k=cnt[ii]-1;k>=idx;k--)         // Раздвижка в линейном массиве
        pp[ii][k+1] = pp[ii][k];
    pp[ii][idx] = val;                      // Вставка
    cnt[ii]++;                              // Увеличить кол-во в лин. массиве
    total++;
    if (cnt[ii]==size0){                    // Переполнение
        testDMUsize();                      // Возможный realloc ДМУ и массива счетчиков
        for(int k=cidx-1;k>ii;k--){         // Раздвижка ДМУ и счетчиков
            pp[k+1] = pp[k];
            cnt[k+1] = cnt[k];
            }
        pp[ii+1]=new int[size0];            // Вставка нового линейного массива
        cnt[ii]=size0/2;                    // Перенос половины из текущего в
        cnt[ii+1]=size0 - cnt[ii];          // следующий
        for(int k=0;k<cnt[ii+1];k++)
            pp[ii+1][k] = pp[ii][size0/2+k];
        cidx++;                             // Увеличить кол-во лин. массивов
        }
    return 1;
```

Примеры. ДМУ линейных массивов

```
int insert(int val, int idx){
    int ii = getIndex(idx);                 // Место вставки [ii,idx]
    if (ii==-1)
        return 0;
    for(int k=cnt[ii]-1;k>=idx;k--)         // Раздвижка в линейном массиве
        pp[ii][k+1] = pp[ii][k];
    pp[ii][idx] = val;                      // Вставка
    cnt[ii]++;                              // Увеличить кол-во в лин. массиве
    total++;
    if (cnt[ii]==size0){                    // Переполнение
        testDMUsize();                      // Возможный realloc ДМУ и массива счетчиков
        for(int k=cidx-1;k>ii;k--){         // Раздвижка ДМУ и счетчиков
            pp[k+1] = pp[k];
            cnt[k+1] = cnt[k];
            }
        pp[ii+1]=new int[size0];            // Вставка нового линейного массива
        cnt[ii]=size0/2;                    // Перенос половины из текущего в
        cnt[ii+1]=size0 - cnt[ii];          // следующий
        for(int k=0;k<cnt[ii+1];k++)
            pp[ii+1][k] = pp[ii][size0/2+k];
        cidx++;                             // Увеличить кол-во лин. массивов
        }
    return 1;
```

Примеры. ДМУ линейных массивов

```
77
1000
77
  0:     0     1     2     3     4     5     6
  1:     7     8     9    10    11    12    13
  2:    14    15    16    17    18    19    20
  3:    21    22    23    24    25    26    27
  4:    28    29    30    31    32    33    34
  5:    35    36    37    38    39    40    41
  6:    42    43    44    45    46    47    48
  7:    49    50    51    52    53    54    55
  8:    56    57    58    59    60    61    62
  9:    63    64    65    66    67    68    69
 10:    70    71    72    73    74    75    76
 11:  1000    77    78    79    80    81    82    83
 12:    84    85    86    87    88    89    90
 13:    91    92    93    94    95    96    97
 14:    98    99   100   101   102   103   104
 15:   105   106   107   108   109   110   111
 16:   112   113   114   115   116   117   118   119
----------------------------
```

```
int main(){
    LDS_DMU xx;
    xx.create(100);
    for(int i=0;i<120;i++)
        xx.add(i);
    printf("%d\n",xx.get(77));
        xx.insert(1000,77);
    printf("%d\n",xx.get(77));
    printf("%d\n",xx.get(78));
    xx.show();
    puts("----------------------------");
    for(int i=0;i<30;i++)
        xx.insert(1000+i,77);
    xx.show();
    puts("----------------------------");
    for(int i=0;i<30;i++)
        xx.remove(33);
    xx.show();
    getchar();
    return 0;
}
```

Примеры структур данных на МУ

```
----------------------------
  0:    0    1    2    3    4    5    6
  1:    7    8    9   10   11   12   13
  2:   14   15   16   17   18   19   20
  3:   21   22   23   24   25   26   27
  4:   28   29   30   31   32   33   34
  5:   35   36   37   38   39   40   41
  6:   42   43   44   45   46   47   48
  7:   49   50   51   52   53   54   55
  8:   56   57   58   59   60   61   62
  9:   63   64   65   66   67   68   69
 10:   70   71   72   73   74   75   76
 11: 1029 1028 1027 1026 1025 1024 1023 1022
 12: 1021 1020 1019 1018 1017
 13: 1016 1015 1014 1013 1012
 14: 1011 1010 1009 1008 1007
 15: 1006 1005 1004 1003 1002
 16: 1001 1000 1000   77   78
 17:   79   80   81   82   83
 18:   84   85   86   87   88   89   90
 19:   91   92   93   94   95   96   97
 20:   98   99  100  101  102  103  104
 21:  105  106  107  108  109  110  111
 22:  112  113  114  115  116  117  118  119
```

```
puts("----------------------------");
for(int i=0;i<30;i++)
    xx.insert(1000+i,77);
xx.show();
```

Примеры. ДМУ линейных массивов

```
puts("----------------------------");
for(int i=0;i<30;i++)
    xx.insert(1000+i,77);
xx.show();
puts("----------------------------");
for(int i=0;i<30;i++)
    xx.remove(33);
xx.show();
```

```
----------------------------
 0:    0    1    2    3    4    5    6
 1:    7    8    9   10   11   12   13
 2:   14   15   16   17   18   19   20
 3:   21   22   23   24   25   26   27
 4:   28   29   30   31   32
 5:   63   64   65   66   67   68   69
 6:   70   71   72   73   74   75   76
 7: 1029 1028 1027 1026 1025 1024 1023 1022
 8: 1021 1020 1019 1018 1017
 9: 1016 1015 1014 1013 1012
10: 1011 1010 1009 1008 1007
11: 1006 1005 1004 1003 1002
12: 1001 1000 1000   77   78
13:   79   80   81   82   83
14:   84   85   86   87   88   89   90
15:   91   92   93   94   95   96   97
16:   98   99  100  101  102  103  104
17:  105  106  107  108  109  110  111
18:  112  113  114  115  116  117  118  119
```

Внутренний мусор при удалении

```
----------------------------
 0:    0    1    2    3    4    5    6
 1:    7    8    9   10   11   12   13
 2:   14   15   16   17   18   19   20
 3:   21   22   23   24   25   26   27
 4:   28   29   30   31   32
 5:
 6:
 7:
 8:
 9:   63   64   65   66   67   68   69
10:   70   71   72   73   74   75   76
11: 1029 1028 1027 1026 1025 1024 1023 1022
12: 1021 1020 1019 1018 1017
13: 1016 1015 1014 1013 1012
14: 1011 1010 1009 1008 1007
15: 1006 1005 1004 1003 1002
16: 1001 1000 1000   77   78
17:   79   80   81   82   83
18:   84   85   86   87   88   89   90
19:   91   92   93   94   95   96   97
20:   98   99  100  101  102  103  104
21:  105  106  107  108  109  110  111
22:  112  113  114  115  116  117  118  119
```