Одномерные динамические массивы данных

(на основе указателей)

Память системы

Высший адрес

Динамическая память

Глобальные переменные

Область программы

Стековая память

Низший адрес

```
#include<cstdio>
using namespace std;
float g(){return 2;}
double global;
int main(){
    float *fptr = new float;
    bool local;
    printf("%p\n", &global);//static
    printf("%p\n", &local); //stack
    printf("%p\n", fptr);   //dynamic
    printf("%p\n", &g);     //program
    printf("%p\n", &main);  //program
}
```

Пример возможного размещения переменных в памяти

```
char ch = 'G';
int date = 1937;
float summa=0.02015;
```

Машинный адрес	0012FF48	0012FF49	0012FF4A	0012FF4B	0012FF54	0012FF55	0012FF56	0012F57	0012FF63
Значение в памяти	0.02015				1937				‘G’
Имя	summa				date				ch

Указатель

1. Указатель не является самостоятельным типом, он всегда связан с каким-то другим типом.

2. Указатели делятся на две категории:

- указатели на объекты;
- указатели на функции.

3. Тип объекта, адрес которого будет содержать указатель может соответствовать базовому, пустому, перечисляемому или структурному типу, типу объединения, пользовательскому типу.

Синтаксис указателя

тип_данных *имя_указателя;

Например:

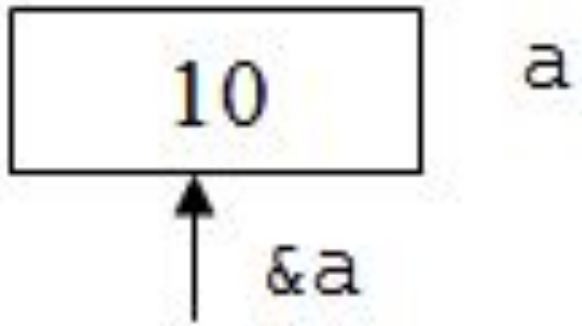
```
int *a;     // пробелы
bool * b;   // не
double* c; // влияют
float d, *e, f; // звездочка относится к имени
```

Операция взятия адреса

&имя_переменной

Например:

```
int a = 10;
cout << &a;
```



Инициализация указателя

- с помощью операции получения адреса:

```
char val = '$';
char *c = &val;
```

- с помощью проинициализированного указателя:

```
char *r = c;
```

- присваивание указателю адреса области памяти в явном виде:

```
char *cp = (char*)0xB8000000;
```

- присваивание указателю пустого значения:

```
int *N = nullptr; //since C++11
float *F = NULL; //C, until C++11
// равносильно float *F = 0;
```

*адрес, который помещается в указатель, должен быть **<u>одного с ним типа</u>**

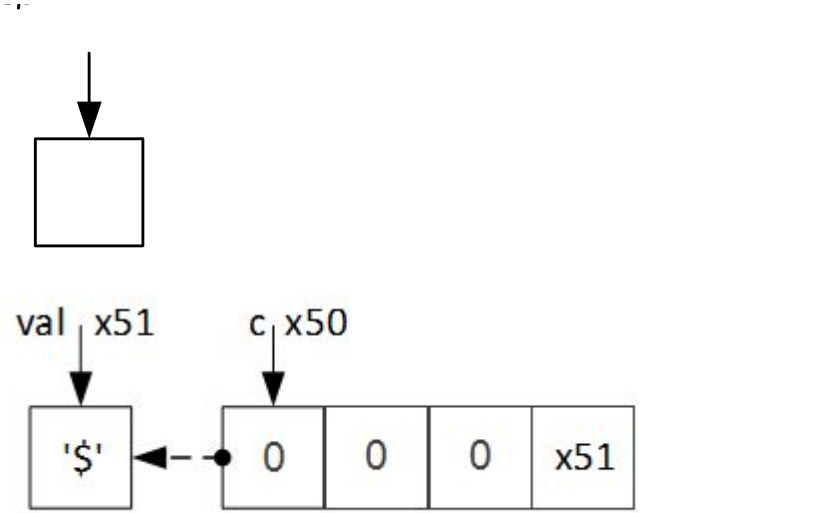
Разыменование указателя (доступ к значению по указателю)

*имя_указателя

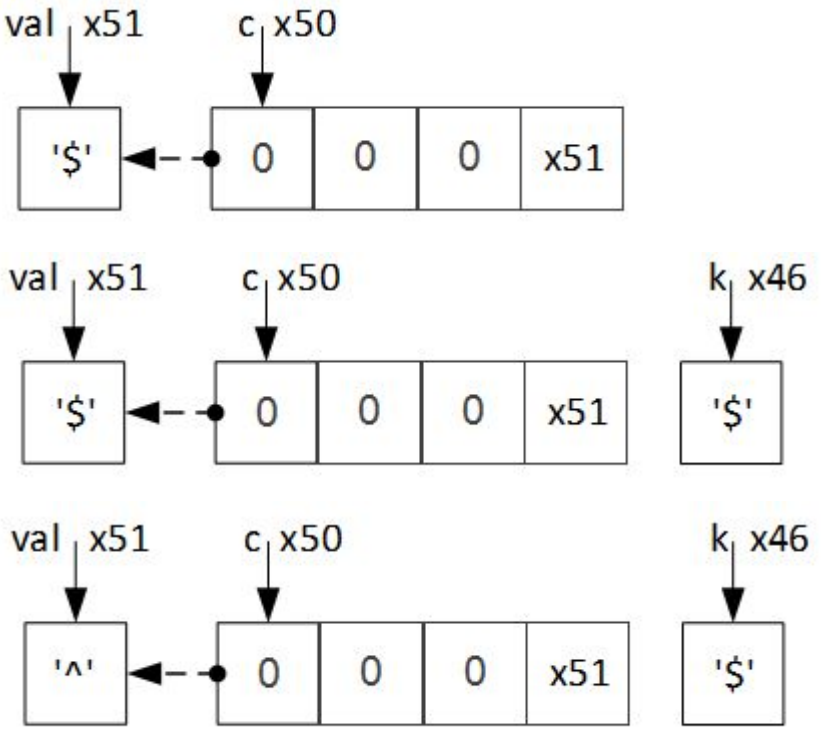
```
//имя1 - просто переменная
тип1 имя1 = значение;
/* имя2 - указатель, хранящий адрес первой переменной*/
тип1 *имя2 = &имя1;
/* имя3 – переменная, в которую записано значение,
  хранящееся по адресу имя2 */
тип1 имя3 = *имя2;
```

Пример разыменования указателя

```
char val = '$';
```



```
char *c = &val;
```




```
char k = *c;
```

```
*c = '^';
```

*на самом деле адреса выравниваются компиляторами (по 2 или 4 байта)

Операция new

тип *имя = new тип [ ]; - индексная форма (для блока данных)

тип *имя = new тип; - без индексная форма (для одиночного объекта)

Тип после оператора new должен быть согласован с базовым типом указателя.

Индексная операция new ищет в динамической памяти блок подходящего размера (исходя из размера типа хранимых данных и количества ячеек) и возвращает адрес первой ячейки из блока. Если поиск был неудачным, то new возвращает nullptr.

```
int *mas = new int[5];
int *obj = new int;
```

Пример создания динамического массива с данными типа float

```
int main(){
int n;
cin >> n;
float *mas = new float [n];
// в динамической памяти выделено n ячеек,
// каждая имеет размер, соответствующий размеру типа float
}
```

Инициализация значения по указателю на динамическую область памяти

```
float *p1 = new float;

float *p = new float(1.5);
//эквивалентен фрагменту

float *p = new float;
*p = 1.5;
```

*здесь выделяется 1 ячейка размера float

Операция delete

delete [ ] имя; - индексная форма (для блока данных)

delete имя; - без индексная форма (для одиночного объекта)

```
int n;
cin >> n;
float *mas = new float [n];
float *p = new float;
…
delete [] mas;
delete p;
```

Пример ввода и вывода одномерного динамического массива

```
unsigned short n;
cin >> n;
float *mas = new float[n];
for (int i = 0; i < n; i++)
   cin >> mas[i];
cout << "The array is consist of:\n";
for (int i = 0; i < n; i++)
   cout << mas[i] << " ";
}
```

*компилятор не проводит проверку границ массива, может произойти перезапись недопустимых областей памяти

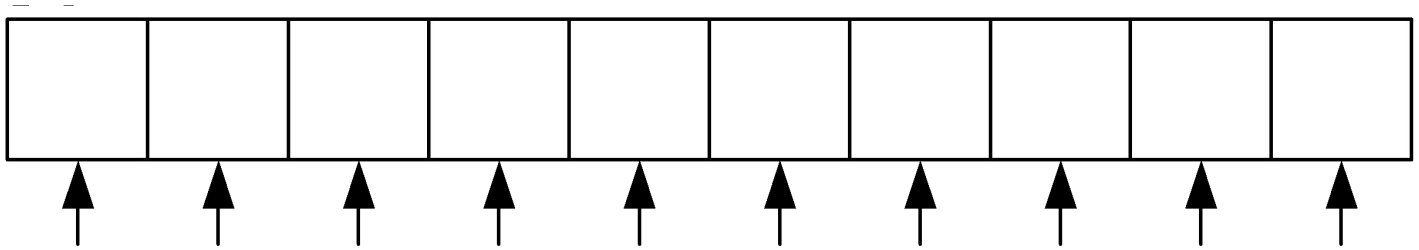
Оператор sizeof и одномерные массивы

```
const int n = 10;
short sa[n];
cout << sizeof(sa) << endl; //20 – размер массива в байтах
short *da;
cout << sizeof(da) << endl; //4 – размер указателя
da = new short [n];
cout << sizeof(da) << endl; //4 – размер указателя
```

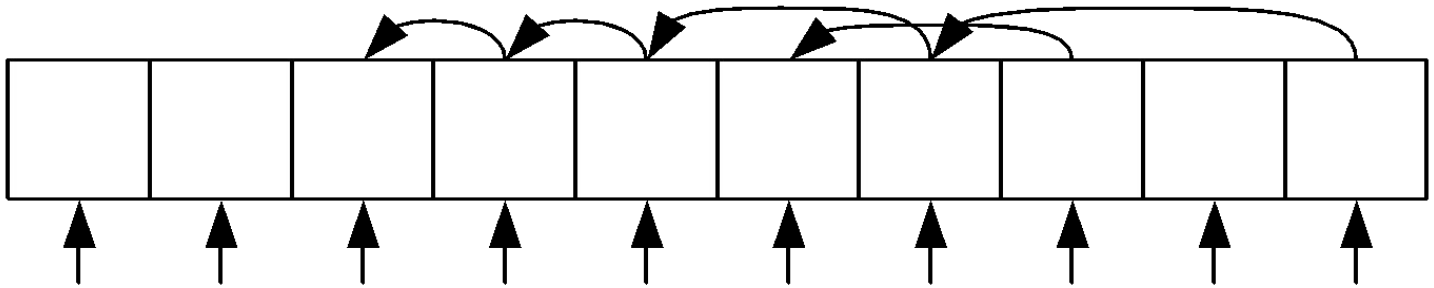
Перераспределение памяти

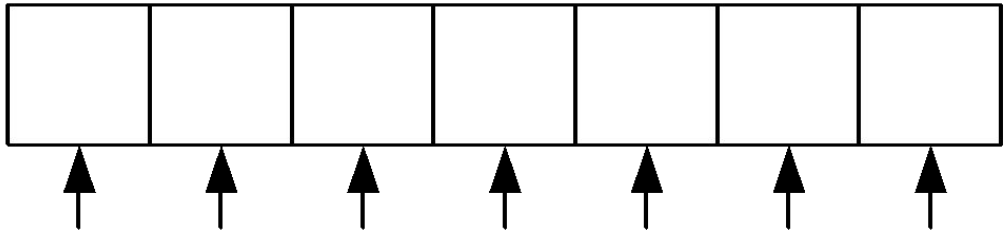
```
int n, m;
cin >> n >> m;
float *mas = new float [n];
//работа с динамическим массивом размера n
…
delete [] mas;
mas = new float [m];
//mas указывает на другую область памяти
//работа с динамическим массивом размера m
…
delete [] mas;
```

Удаление элементов массива



Удалить каждый 3-й элемент





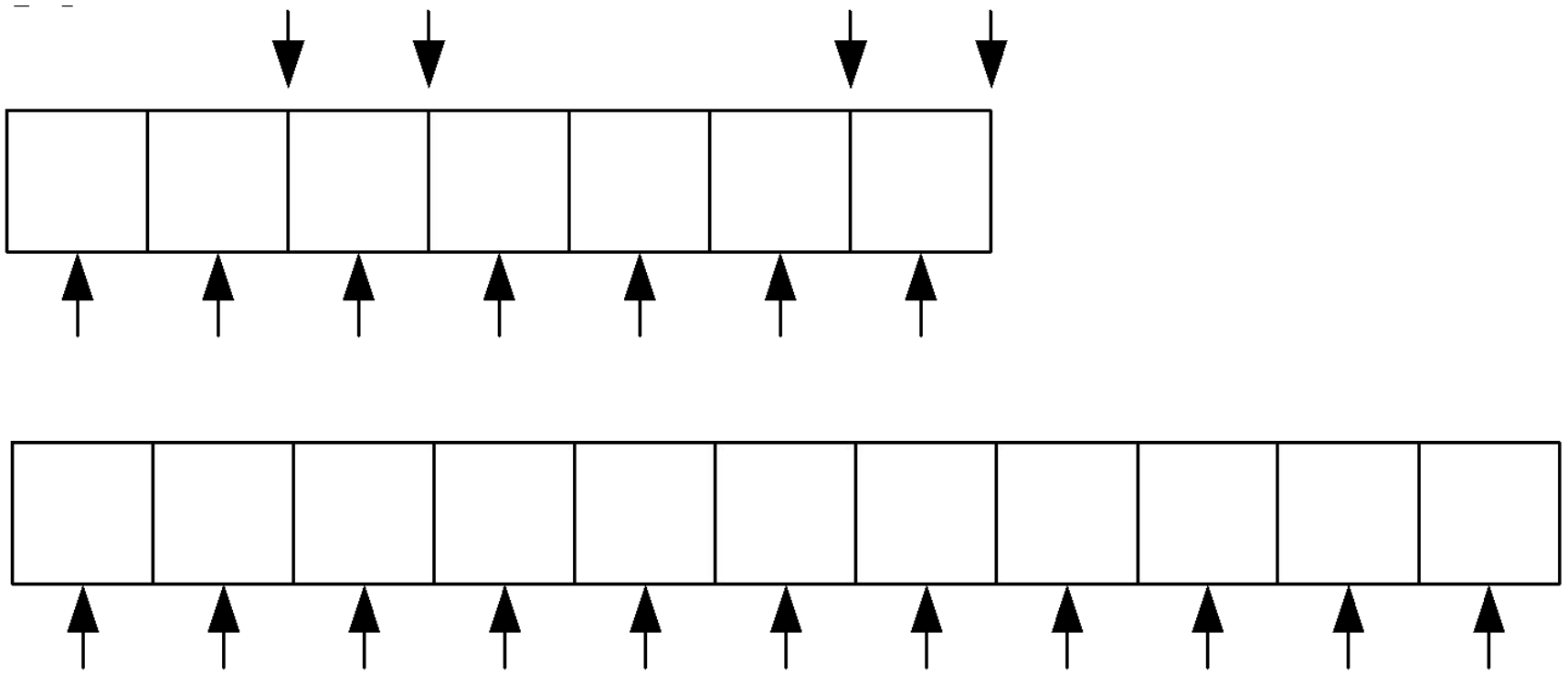
Удаление каждого 3-го элемента массива (имитация)

```
int n, *mas, *newmas;
cin >> n;
mas = new int[n];
newmas = new int[n * 2 / 3 + bool(n % 3)];
for (int i = 0; i < n; i++)
    cin >> mas[i];
for (int i = 0, j = 0; i < n; i++)
    if (i % 3 != 2)
        newmas[j++] = mas[i]; //mas[j++] = mas[i];*
delete[] mas;
mas = newmas;
n = n * 2 / 3 + bool(n % 3);
```

* в случай с закомментированным массив занимает такой же объем памяти, последние n/3 ячеек хранят ненужные значения

Добавление элементов в массив

После каждого четного значения элемента вставить 0.



После каждого четного значения элемента массива вставить значение 0 (имитация)

```
int t, m = 7, *b, *a;
a = new int[m];
t = m;
for (int i = 0; i < m; i++){
   cin >> a[i];
  if (!(a[i] % 2)) t++; //считаем размер нового массива
  }
b = new int[t];
for (int i = 0, j = 0; i < m; i++){
   b[j++] = a[i];
   if (a[i] % 2 == 0)
      b[j++] = 0;
   }
delete[] a;
a = b;
m = t;
```

Особенности нулевого указателя

```
double *a = nullptr;  //C++11
double *b = NULL // или 0 - нулевой указатель языка C

bool c;
. . .
if (c) {
    a = new double[1000];
    . . .
}
. . .
delete[] a;
```

*без инициализации a хотя бы nullptr будет утечка памяти

Указатель на константу

const тип *имя;

```
const float value = 3.5;
float nonConstValue = 4.5;
const float *pointer = &value;
/*можно скопировать значения из адресуемой ячейки*/
float valueCopy = *pointer;
//можно изменить указатель
pointer = &nonConstValue;
/* нельзя изменять значение адресуемой ячейки через
 указатель pointer
*pointer = valueCopy; */
```

Константный указатель

тип * const имя;

```
float value = 3.5, anotherValue = 4.5;
//нельзя не инициализировать указатель
float * const pointer = &value;
//можно менять значение по указателю
*pointer = 5.5;

/* нельзя изменить сам указатель
pointer = &anotherValue
*/
```

*статический массив является константным указателем

Константный указатель на константу

const тип * const имя;

```
float value = 3.5;
//нельзя не инициализировать указатель
const float * const pointer = &value;
```

Допустимые операции над указателями

- разыменование (*);
- взятие адреса (&);
- присваивание;
- арифметические операции
  - инкремент (++) увеличивает значение указателя на величину sizeof(тип);
  - декремент (--) уменьшает значение указателя на величину sizeof(тип);
  - сложение указателя только с целочисленной константой,
  - вычитание: допускается разность указателей и разность указателя и целочисленной константы,
- сравнение;
- приведение типов.

Примеры использования операций над указателями

```
//приведение указателя к типу др. указателя
unsigned long L = 12345678;
float *fp = (float*)&L;
int *ip = (int*)&L;
//сравнение указателей
int x = 10, y = 10;
int *xptr = &x, *yptr = &y;
if (xptr == yptr)
   cout << "Указатели равны\n";
//сравнение значений, на которые указывают указатели
if (*xptr == *yptr)
cout << "Значения равны\n";
```

Примеры использования операций над указателями

```
int n = 10;
float *array = new float[n];
float *ptr = array;//указывает на ячейку array[0]
ptr++; //указывает на ячейку array[1]
ptr += 2; //указывает на ячейку array[3]
ptr--; //указывает на ячейку array[2]
cout << ptr - array; //значение 2
```

Обращение к ячейке массива с использованием указателя

```
unsigned n;
cin >> n;
float *mas = new float [n];
/*тут должно быть заполнение ячеек массива*/
for (unsigned i = 0; i < n; i++){
   cout << *(mas + i) << " ";
    // равносильно cout <<  mas[i] << " ";
   }
```

Обращение к ячейке статического массива через указатели

```
int intArray[5] = { 31, 54, 77, 52, 93 };
int *ptrInt; // указатель на int
ptrInt = intArray;
for(int j = 0; j < 5; j++)
    //можно через копию указателя intArray
    cout << *ptrInt++ << endl;// *(ptrInt + j);
    //но нельзя через сам intArray (попытка изменить const)
    //cout << *intArray++ << endl;
```

* приоритет постфиксного++ выше разыменования

** префиксный ++ имеет тот же ранг приоритета, что и разыменование, но ассоциативность операции справа налево

Забавы с арифметикой указателей

```
int main(){
int a = 10;
int m[5] = {1, 2, 3, 4, 5};
int b = 20;
int *p = m;
for(int i = 0; i < 5; i++)
   cout << m[i] << " ";
cout << endl << "a=" << a << ", &a=" << &a
     << ", b=" << b << ", &b=" << &b
     << ", p=" << p <<  ", m=" << m << endl;
p--;
*p = 7;
for(int i = 0; i < 5; i++)
   cout << m[i] << " ";
cout << endl << "a=" << a << ", &a=" << &a
     << ", b=" << b << ", &b=" << &b
     << ", p=" << p <<  ", m=" << m << endl;
}
```

Указатель на void и преобразование типов

```
int i = 5;
void *p = &i;
int i2 = *((int*)p);
// *((int*)p)=7;
char c = 'r';
p = &c;
cout << *((char*)p);
```

* адрес, который помещается в указатель, должен быть одного с ним типа

Указатель на указатель

```
int i = 1, j = 10, *pi, **ppi;
cout << i << endl;
pi = &i;
(*pi)++; /*инкремент значения в i через указатель pi*/
ppi = &pi; /*указатель ppi хранит адрес указателя pi*/
(**ppi)++; /*инкремент значения в i через указатель ppi*/
*ppi = &j;/*запись в указатель pi адреса j, равносильно
 pi = &j; */
```

Приведение типов reinterpret_cast

```
int i;
cin >> i;
char* pc = reinterpret_cast<char*>(&i);
```

Выделение и освобождение динамической памяти в языке C

void * malloc(size_t size);

free (имя);

самостоятельное изучение: calloc, realloc

```
int  n;
cin >> n;
int *mas = (int*) malloc(n * sizeof(int));
…
free (mas);
```

Ссылка

```
int val1, val2;
int &ref1 = val1;
//ссылку нельзя переназначить
//ref1 = val2; - ошибка
//ссылка на динамически создаваемый объект
int &ref = *(new int);
ref = 5;
cout << ref; //5
```