Указатели

Динамическая память

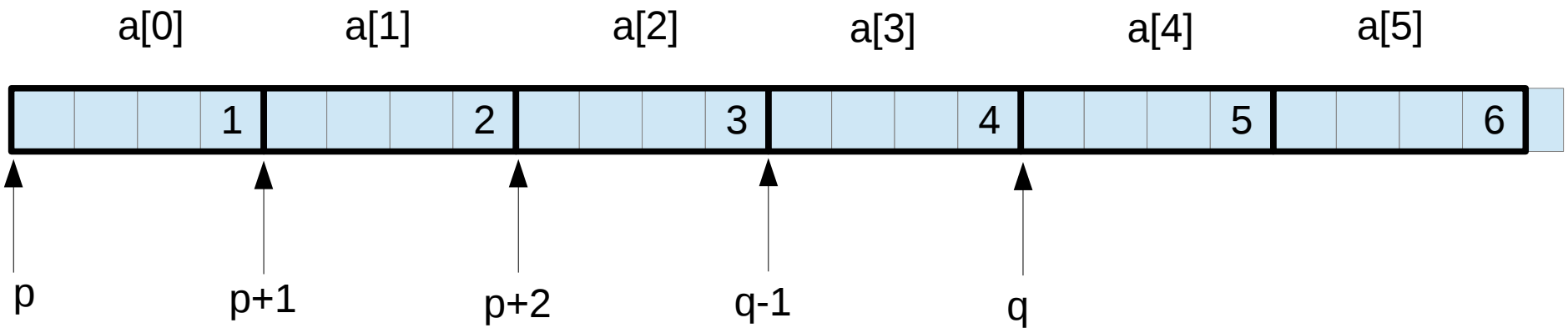
- **Статическая память** — это область памяти, выделяемая при запуске программы до вызова функции main из свободной оперативной памяти для размещения глобальных и статических объектов, а также объектов, определённых в пространствах имён.
- **Автоматическая память** — это специальный регион памяти, резервируемый при запуске программы до вызова функции main из свободной оперативной памяти и используемый в дальнейшем для размещения локальных объектов: объектов, определяемых в теле функций и получаемых функциями через параметры в момент вызова. Автоматическую память часто называют **стеком**.
- **Динамическая память** — это совокупность блоков памяти, выделяемых из доступной свободной оперативной памяти непосредственно во время выполнения программы под размещение конкретных объектов.

Указатели

Указатель – это переменная, в которой записан адрес ячейки памяти компьютера.

```
//целая переменная равная 5
int Variable = 5;

// указатель на нее
 // операция ВЗЯТИЯ АДРЕСА переменной
 int * pPointer = & Variable;

// переменная со значением по этому адресу
// операция РАЗЫМЕНОВАНИЯ
 int Value = * pPointer;

//динамическая память для int
int * pNewPointer = new int;

//динамический массив
double * arr = new double[10];

//освобождение памяти по адресу pNewPointer
delete pNewPointer;

//освобождение памяти, занимаемой массивом
delete [] arr;
```

Действия с указателями

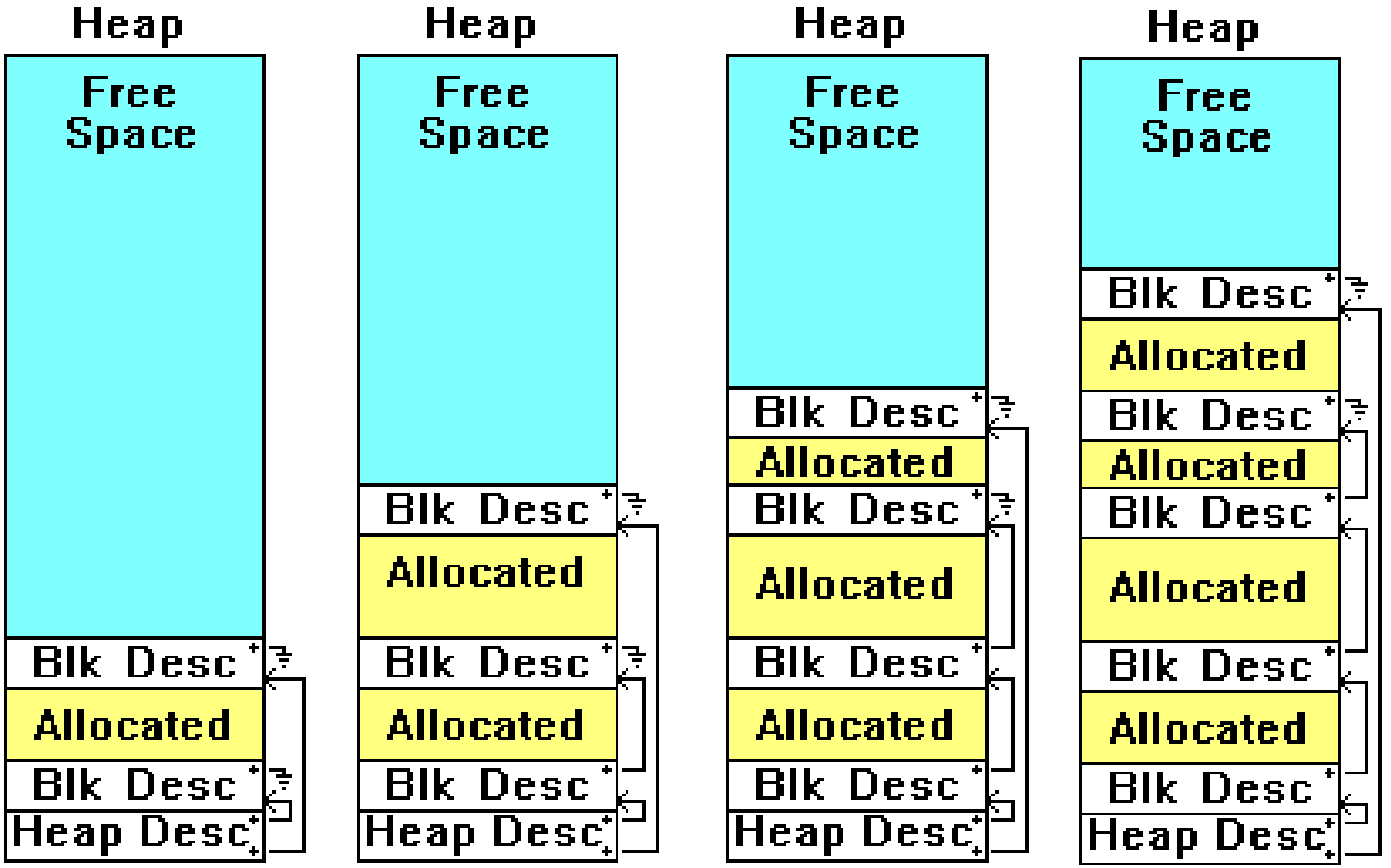



```
int a[] = { 1, 2, 3, 4, 5, 6 };

int* p = a; // имя массива тоже указатель

int* q = p + 4;

p+2 == q-2  // адреса равны
```

p+2 – хранит адрес третьего элемента массива a[]

p[2] – хранит значение 3

Динамическая память



Динамическая память

- **Динамическое распределение памяти** — способ выделения оперативной памяти компьютера для объектов в программе, при котором выделение памяти под объект осуществляется во время выполнения программы.
- **Менеджер памяти** — часть компьютерной программы (как прикладной, так и операционной системы), обрабатывающая запросы на выделение и освобождение оперативной памяти или запросы на включение заданной области памяти в адресное пространство процессора.
- **Ку́ча** (англ. heap) — название структуры данных, с помощью которой реализована динамически распределяемая память приложения, а также объём памяти, зарезервированный под эту структуру.

Указатели на структуры

```
struct Cat
{
    int age;
    int weight;
};
```

```
Cat* pCat = new Cat;
```

Для доступа к полям структуры используется два способа:

```
(*pCat).age = 5;

pCat->age = 5;

pCat->weight = 3;
```

Ссылки

Ссылка – это практически то же, что и псевдоним. В отличие от указателя, ссылки необходимо инициализировать при объявлении.

```
int iOne;
int &rSomeref = iOne;
```

После этого ссылка не может переназначаться.

Ссылки можно рассматривать как указатели, которые не нужно разыменовывать (с двумя оговорками, приведенными выше).

Подведем итоги

```
double pi = 3.14;
char arr[30];

int main()
{
    int i = 0;
    double sum = 0;
    return 0;
}

void function()
{
    int x = 4;
    double* p = new double[16];
    delete [] p;
}
```

Какие способы выделения памяти представлены на примере?

Подведем итоги

```
double pi = 3.14;
char arr[30];

int main()
{
    int i = 0;
    double sum = 0;
    return 0;
}

void function()
{
    int x = 4;
    double* p = new double[16];
    delete [] p;
}
```

Переменная и массив будут храниться в статической памяти.

Подведем итоги

```
double pi = 3.14;
char arr[30];

int main()
{
    int i = 0;
    double sum = 0;
    return 0;
}

void function()
{
    int x = 4;
    double* p = new double[16];
    delete [] p;
}
```

Локальные параметры будут храниться в стековой памяти. После выхода из подпрограммы память, занимаемая ими, будет считаться свободной.

Подведем итоги

```
double pi = 3.14;
char arr[30];

int main()
{
    int i = 0;
    double sum = 0;
    return 0;
}

void function()
{
    int x = 4;
    double* p = new double[16];
    delete [] p;
}
```

Выделяется память в «куче» под массив из 16-ти элементов типа double.