Память

Глобальная память– память, переменные которой существуют с начала программы. Объявлены снаружи, помечены statik.
statik int count = 0; – Статичная переменная, помогает избежать переинициализации/

Локальная память(стек) – память, переменные которой существуют только в пределах функции, в которой были объявлены. Попытка выделить память сверх стека приведёт к акарийной остановке программы.

Динамическая память–память, которая выделяется вручную. Обязательно возвращать выделенную память обратно в систему, когда она становится ненужной, иначе можно хаполнить всю память, что приведёт к ошибке.
new,delete – операторы выделения и удаления памяти.
new int – в динамической памяти появится кусочек int.
int *a;–указатель на ячейку памяти, в которой хранится адрес ячейки со значением переменной.
***a**–разыменователь
int *x= &x; – присвоение указателю адреса перемнной.&–взятие адреса переменной.
int *a=new int;
(часть программы, где задействован указатель)
delete a; – удаление динамической переменной.

Так будет выглядеть массив в памяти: ячейки памяти следующие друг за другом. **a[5]**–создать массив из 5-ти элементов. *(a+5) – разыменование пятой ячейки после a.

int	int a+1	int a+2	int a+3	int a+4
a	a+1	a+2	a+3	a+4

int *M = new int[n]; //n–кол-во элементов
for(int i=0;i<n;cin»a[i++]){}
(какой-то код)
delete[ ]M;

a[i][j]=(a[i])[j]=(*(a+i))[j]=*(*(a+i)+j)–представление двумерного массива через указатели

Заведение и удаление двумерного массива

```
int **Ar = new int[m];
Ar[i]=new int [n];
for (int i=0;i<n;i++){
    delete [ ]Ar[i];
}
delete [ ]Ar;
```

void swap (int a,int b){
int c=a;
a=b;
b=c;
}
swap(x,y)
	void swap(int *a,int *b){
	int c=*a;
	*a=*b;
	*b=c;
	}
	swap(&x,&y);

Значения x и y не поменяются	Значения x и y поменяются

```
void swap (int &a,int &b){
    int c=a;
    a=b;
    b=c;
}
```

swap(x,y)–тоже поменяяет значения, так как функция будет работать не со значениями переменных, а с их адресами.

int A[1000]; (чтение) int s = 0;	

for(int i=0;i<1000;i++){
s+=A[i];
}
	int *q = A+1000;
	for(int *p=A;p<q;p++){
	s+=*p}

1000 « < » 1000 «++» 1000 «+=» 1000 «*()» 1000 «+»	1000 « < » 1000«++» 1000 «+=» 1000 «*()»
В современном c++ осуществляется векторизация	интерирование

В современном c++ левый код выполнится бысрее.