Перегрузка операторов

Вернемся назад...

```
#include <iostream>

using namespace std;

struct vect
{
    int x;
    int y;
};

int main()

{
    vect V1, V2, V3;
    V1.x = 1;
    V1.y = 2;
    V2.x = 1;
    V2.y = 2;
    V3.x = V1.x + V2.x;
    V3.y = V1.y + V2.y;
    disp_vextor(V3);
    return 0;
}
```

Задача сложения двух векторов

Полный код см. в репозитории

Вернемся назад...

```
#include <iostream>

using namespace std;

struct vect
{
    int x;
    int y;
};

int main()

{
    vect V1, V2, V3;
    V1.x = 1;
    V1.y = 2;
    V2.x = 1;
    V2.y = 2;
    V3.x = V1.x + V2.x;
    V3.y = V1.y + V2.y;
    disp_vextor(V3);
    return 0;
}
```

Задача сложения двух векторов

Некрасиво!

Хотелось бы что-то вроде

V3 = V1 + V2

Перегрузка операторов

```
struct vect
{
    int x;
    int y;
    vect operator+(vect other);
};

vect vect::operator + (vect other)
{
    vect C;
    C.x = x + other.x;
    C.y = y + other.y;
    return C;
}


int main()
{
    V3 = V1 + V2;
}
```

Перегружаем **бинарный** оператор +

Полный код см. в репозитории

Перегрузка операторов

```
struct vect
{
    int x;
    int y;
    bool operator<(vect);
};
```

Перегружаем **бинарный** оператор <

```
bool vect::operator < (vect other) {
       return x < other.x && y < other.y ? 1 : 0;
}
```

```
int main()
{
    bool B = V1 < V3;
}
```

Полный код см. в репозитории

Перегрузка операторов

```
struct vect
{
    int x;
    int y;
    void operator++();
    void operator++(int);
};

void vect::operator ++ () {
    ++x;
    ++y;
}

void vect::operator ++ (int) {
    x++;
    y++;
}
```

Перегружаем **унарные** операторы ++

```
int main()
{
    ++V3;
    V3++;
}
```

Полный код см. в репозитории

Перегрузка операторов

```
struct vect
{
    int x;
    int y;
    void operator+();
};

void vect::operator + () {
        cout << "?????" << endl;
}


 int main()
 {
     +V3;
 }
```

Перегружаем **унарный** оператор +

Полный код см. в репозитории

Выделение динамической памяти

C

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

int main()
{
        int N;
        scanf_s("%d", &N);
        int* p = (int*)malloc(N * sizeof(int));
        for (size_t i = 0; i < N; ++i)
        {
            p[i] = i;
        }
        for (size_t i = 0; i < N; ++i)
        {
            printf("%d ",p[i]);
        }
        printf("\n");
        free(p);
        return 0;
}
```

C++

```cpp
#include <iostream>

int main()
{
        int N;
        std::cin >> N;
        int* p = new int[N];
        for (size_t i = 0; i < N; ++i)
        {
            p[i] = i;
        }
        for (size_t i = 0; i < N; ++i)
        {
            std::cout << p[i] << " ";
        }
        std::cout << std::endl;
        delete []p;
        return 0;
}
```

См. лекцию 8

Перегрузка операторов

```
void* operator new(std::size_t size)
{
    cout << "Malloc: " << size << endl;
    void* ptr;
    ptr = malloc(size);
    return ptr;
}

void operator delete(void* ptr)
{
    cout << "Free\n";
    free(ptr);
}

int main()
{
    int* a = new int;
    delete a;
}
```

Перегрузка операторов new/delete

Полный код см. в репозитории

Перегрузка операторов

```
void* operator new[](std::size_t size)
{
    cout << "Malloc[]: " << size << endl;
    void* ptr;
    ptr = malloc(size);
    return ptr;
}

void operator delete[](void* ptr)
{
    cout << "Free[]\n";
    free(ptr);
}

int main()
{
    int* a = new int;
    delete a;
}
```

Перегрузка операторов new[]/delete[]

Полный код см. в репозитории

Работа vector при перегрузке операторов

```cpp
int main()
{
    vector<int> vec;
    for (size_t i = 0; i < N; i++)
    {
        vec.push_back(i);
    }
    return 0;
}
```

```
Malloc: 16
Malloc: 4
Malloc: 8
Free
Malloc: 12
Free
Malloc: 16
Free
Malloc: 24
Free
Free
Free
```

Теперь vector использует перегруженные операторы

Полный код см. в репозитории