**СЕМИНАР 3. Подготовка к рейтингу 1**

**Теоретический материал**

Константа – это фиксированное значение, которое не может быть изменено программой. Это переменная, в которую можно заложить данные только 1 раз, потом она блокируется на запись, а любая попытка записать в неё что-нибудь потом воспринимается как ошибка программы.

**const char \*p;**

означает, что указатель **p** может указывать на некоторое символьное значение, но это значение этим указателям менять нельзя.

Можно сделать сам указатель константным, то есть значение указателя менять нельзя, а то, на что он показывает, менять можно. Делается это помещением модификатора *const* справа от звездочки:

**char \* const p;**

Можно сделать так, что сам указатель будет константным, то есть неизменным, и то, на что он указывает, также будет неизменным:

**const char \* const p;**

Кроме того модификатор *const* может указываться у функций-членов класса, означая тем самым, что подобная функция не имеет права изменять значение объекта.

Функция быстрой сортировки **qsort**

|  |  |
| --- | --- |
|  | **void** **qsort** ( **void** \* first, **size\_t** number, **size\_t** size,  **int** ( \* comparator ) ( **const** **void** \*, **const** **void** \* ) ); |

Функция **qsort** выполняет сортировку *num* элементов массива, на который ссылается указатель *first*. Для каждого элемента массива устанавливается размер в байтах, который передается через параметр *size*. Последний параметр функции **qsort** — указатель *comparator* на функцию сравнения, которая используется для определения порядка следования элементов в отсортированном массиве.

Параметры:

* **first**Указатель на первый элемент сортируемого массива.
* **number**Количество элементов в сортируемом массиве, на который ссылается указатель first.
* **size**Размер одного элемента массива в байтах.
* **comparator**Функция, которая сравнивает два элемента. Функция должна иметь следующий прототип:

|  |  |
| --- | --- |
| 1 | **int** funccmp( **const** **void** \* val1, **const** **void** \* val2 ); |

Функция должна принимать два параметра – указатели на элементы массива, типа *void*\*. Эти параметры должны быть приведены к определённым типам данных. Возвращаемое значение этой функции должно быть отрицательным, равным нулю или положительным.  Если *val1* меньше, равен или больше, чем *val2*, функция должна вернуть отрицательное значение, ноль или положительное значение, соответственно.

**sizeof** – это унарный оператор, возвращающий длину в байтах переменной или типа, помещенных в скобки.

**Разбор программы**

#include <stdio.h>

#include <stdlib.h>

class Point2

{

private:

const int x;

const int y;

public:

Point2 (int xx, int yy) : x(xx), y(yy) {};

int getx() { return x; };

int gety() { return y; };

friend int differ (const void\*, const void\*);

int differ (const Point2\*);

};

inline int Point2::differ(const Point2\* p)

{

return (p->x - x);

}

int differ(const void\* pp, const void\* qq)

{

Point2\*\* p = (Point2\*\*) pp;

Point2\*\* q = (Point2\*\*) qq;

return (p[0]->y - q[0]->y);

}

int main (int argc, char\* argv[])

{

Point2\*\* p;

int x, y;

int i=0;

int j;

int n=0;

if(argc < 2)

return (-1);

p = new Point2\*[argc - 1];

while (++i < argc)

{

x = y = 0;

sscanf(argv[i], "%d%\*c%d", &x, &y);

p[n++] = new Point2 (x, y);

}

qsort ((void \*) p, n, sizeof(Point2\*), differ);

for (i=0; i<n; i++)

{

printf("(%d;%d) >", p[i]->getx(), p[i]->gety());

for (j=0; differ(&p[j], &p[i]) < 0; j++)

if (p[i]->differ(p[j]) < 0)

printf (" (%d;%d)", p[j]->getx(), p[j]->gety());

putchar ('\n');

}

for (i=0; i<n; i++)

delete p[i];

delete []p;

return 0;

}