Язык программирования C++

Цель курса:

- Овладеть объектно-ориентированным языком программирования C++, освоить принципы работы в среде Visual C++ и приобрести базовые навыки разработки приложений под Windows.
- Курс предназначен для тех, кто умеет программировать на языке C и хочет развить свои профессиональные навыки, изучив C++.

Модуль 1. Типы данных, операции и функции в C++

- Ссылочный тип данных.
- Операции - расширения контекста, new, delete.
- Встроенные inline-функции.
- Перегрузка функций. Аргументы по умолчанию.

C++ без классов, или "улучшенный" C

```
// 1. --- Новые заголовочные файлы

/*
#include <iostream.h>          //заголовочный файл библиотеки ввода/вывода C++
*/                             //в достандартном стиле

#include <iostream>            //заголовочный файл библиотеки ввода/вывода C++
                               //в стандартном стиле
using namespace std;           //директива включения экспортируемых имен стандартной
                               //библиотеки в глобальное пространство имен
/*
using std::cout;               //объявление using включает только указанное имя
using std::cin;
using std::endl;
*/

#include <cstdlib>             //заголовочный файл библиотеки языка C
                               //в стандартном стиле

#include "ftpl_swap.h"         //"свои" заголовочные файлы по-прежнему
                               //имеют расширение .h

double a = 12.345;             //имя a принадлежит глобальному пространству имен
```

C++ без классов, или "улучшенный" C

```
int main()
{
        // 2. --- Альтернативная библиотека ввода/вывода

        std::cout << "Hello, world!\n";      //std::cout - выходной поток (объект
                                             //из пространства имен std)
                                             //:: - оператор доступа к контексту - уточняет,
                                             //что cout принадлежит пространству имен std
                                             //<< - оператор "вставки в поток”

        cout << "Hello, world!\n";           //здесь уточнение происходит благодаря
                                             //директиве using

        cout << "Hello, world!" << endl;     //манипулятор endl - конец строки (std::endl)
                                             //операторы << часто применяют "цепочкой"
```

```
Hello, world!
Hello, world!
Hello, world!
```

C++ без классов, или "улучшенный" C

```
// 3. --- Инструкции описания и исполнения могут чередоваться

        cout << "Enter int and float values: ";

        int a;                          //инструкции описания могут располагаться
        double b;                       //в любом месте кода, где они необходимы

        cin >> a >> b;                  //std::cin - входной поток (объект)
                                        //оператор >> - "извлечение из потока"
                                        //тоже часто применяют "цепочкой"

        cout << "a=" << a << ", b=" << b << endl;

        cout << "a == b is "
             << (a == b) << endl;       //надо помнить приоритеты операторов!!!
```

```
Enter int and float values: 5 5.25
a=5, b=5.25
a == b is 0
```

C++ без классов, или "улучшенный" C

```
        cout << "Half of b:" << b/2 << endl
             << "Half of a:" << a/2 << endl;
```

```
Half of b:2.625
Half of a:2
```

```
        cout << "Casting: "
             << (double)a/2 << '\t'        //(double) - оператор явного
                                           //приведения типа в стиле C
        // 4. --- Функциональная форма оператора явного приведения типа
             << double(a)/2 << endl;       //double() - функциональный стиль C++
                                           //оператора явного приведения типа
```

```
Casting: 2.5    2.5
```

C++ без классов, или "улучшенный" C

```
// 5. --- Новое в описании констант и массивов

const int n = 10;                  //константные переменные можно использовать
int arr[n];                        // для указания размера массива

for(int i=0; i<n; ++i)             //определять переменные можно даже
    arr[i] = rand()%100;           //в инструкциях for, if, while


// 6. --- Новый тип - логический, со значениями true и false

bool t = false;
t = a > b;
cout << a << (t == true ? " > " : " <= ") << b << endl;
```

```
5 <= 5.25
```

C++ без классов, или "улучшенный" C

```
// 7. --- Новое в неявных преобразованиях указателей

const int * cp = &n;     //указатель на константный объект адресует константу

          cp = &a;       //теперь – на неконстантный объект, но это безопасно

//        ++*cp;         //ERROR, т.к. cp - только для чтения (read only)

int * p = &a;            //указатель на неконстантный объект

++*p;                    //изменяет значение адресуемого объекта

cp = p;                  //это безопасно, т.к. ограничивает возможности
//      p = cp;          //не компилируется
        p = (int *) cp;  //требует явного приведения (т.к. это опасно)

void * vp = p;           //это безопасно, т.к. ограничивает возможности
//            p = vp;    //не компилируется
        p = (int *) vp;  //требуется явное приведение (т.к. это опасно)
                         //вспомним, что malloc() возвращает void *

vp = 0;                  //значение NULL не обязательно использовать, т.к.
                         //ноль неявно преобразуется к нулевому указателю
```

C++ без классов, или "улучшенный" C

```
// 8. --- Новый тип - ссылка, она вводит новое имя (псевдоним) для объекта

int & r = a;                    //д.б. инициализирована при создании (связана
                                //с объектом), после чего все операции над
                                //ссылкой относятся к связанному с ней объекту

cout << "a="  <<  a << ",  r=" <<  r << endl;

cout << "&a=" << &a << ", &r=" << &r << endl;

r = b;                          // т.е. эквивалентно a=b;

cout << "a=" << a << ", r=" << r << endl;
```

```
a=6,   r=6
&a=0012FF7C, &r=0012FF7C
a=5, r=5
```

```
const int & cr = a;             //константная ссылка на a для доступа read only
```

C++ без классов, или "улучшенный" C

```
// 8.1 --- Различие между указателем и ссылкой

int c = 12345;

p = &c;                 //указали на c вместо a
r =   c;                //c копируется в a (ссылку нельзя перенастроить)

cout << " c=" <<  c << ", *p=" << *p << endl;
cout << "&c=" << &c << ",  p=" <<  p << endl;

cout << " a=" <<  a << ",  r=" <<  r << endl;
cout << "&a=" << &a << ", &r=" << &r << endl;
```

```
c=12345, *p=12345
&c=0012FF28,  p=0012FF28
 a=12345,  r=12345
&a=0012FF7C, &r=0012FF7C
```

C++ без классов, или "улучшенный" C

```
// 8.2 --- Ссылки как аргументы функций

//предоставляют функции доступ к ее фактическим аргументам (как и указатели),
//но не требуют разадресации (код см. после main(){})


void swap1(int a, int b)                //передача по значению, фактический аргумент
{                                       //может быть выражением
        int t = a; a = b; b = t;
}


void swap2(int *x, int *y)              //передача по адресу, фактический аргумент
{                                       //должен быть адресным выражением
        int t = *x; *x = *y; *y = t;    //указатели разадресуются для доступа
}                                       //к "оригиналам"

void swap3(int & a, int & b)            //передача по ссылке, фактический аргумент
{                                       //НЕ может быть выражением
        int t = a; a = b; b = t;        //ссылки - "оригиналы" без разадресации
}
```

C++ без классов, или "улучшенный" C

```
void swap1(int, int);                  //передача по значению, фактический аргумент
                                       //может быть выражением
void swap2(int *, int *);              //передача по адресу, фактический аргумент
                                       //должен быть адресным выражением
void swap3(int &, int &);              //передача по ссылке, фактический аргумент
                                       //НЕ может быть выражением
a = b;
cout << "initial values:          " << "a=" << a << ", c=" << c << endl;
swap1(a, c);                           //аргументы - переменные (или выражения)
cout << "after swap1(int, int):     " << "a=" << a << ", c=" << c << endl;

swap2(&a, &c);                         //аргументы - адреса переменных
cout << "after swap2(int *, int *): " << "a=" << a << ", c=" << c << endl;

swap3(a, c);                           //аргументы - переменные (НЕ выражения)
cout << "after swap3(int &, int &): " << "a=" << a << ", c=" << c << endl;
```

```
initial values:              a=5, c=12345
after swap1(int, int):       a=5, c=12345
after swap2(int *, int *): a=12345, c=5
after swap3(int &, int &):   a=5, c=12345
```

C++ без классов, или "улучшенный" C

```
// 8.3 --- Ссылка как возвращаемое значение

int & max(int &, int &);
cout << "max=" << max(a,c) << endl;

//вызов функции может стоять слева от знака "=" и быть Lvalue в выражениях
max(a, c)=0;
cout <<"a=" << a <<", c=" << c << endl;

++max(a, c);
cout <<"a=" << a <<", c=" << c << endl;
```

```
max=12345
a=5, c=0
a=6, c=0
```

```
int & max(int & a, int & b)
{
    return a > b ? a : b;
}
```

C++ без классов, или "улучшенный" C

```
// 9. --- Новые операторы

// 9.1. --- доступ к глобальному контексту (унарный :: )

double a = 12.345;              //имя a принадлежит глобальному пространству имен

int main()
{
        ...
        int a;
        ...

        cout << "Local  a = " <<   a << endl;

        cout << "Global a = " << ::a << endl;
```

```
Local  a = 6
Global a = 12.345
```

C++ без классов, или "улучшенный" C

```
// 9.2. --- работа с динамической памятью ( heap )

p = new int(100);       //создание отдельного объекта в "куче"
                        //(100) - инициализатор объекта  (не обязателен)
 {
        // здесь д.б. блок обработки ошибки при выделении памяти
        // C++ имеет специальный механизм обработки подобных
        // "исключительных ситуаций"
}

cout << p << '\t' << *p << endl;
a = *p;
cout << "a = " << a << endl;

*p = 2*a;
cout << p << '\t' << *p << endl;

delete p;               //возврат памяти в кучу, p д.б. равен адресу,
                                //полученному от new
```

```
003207F0 100
a  = 100
003207F0 200
```

C++ без классов, или "улучшенный" C

```
p = new int[10];        //выделение в "куче" массива (размер может быть
                        //любым целочисленным выражением)
                        //массив элементов встроенных типов
                        // в "куче" не может иметь инициализатора

{
        // здесь д.б. блок обработки ошибки при выделении памяти
}


for(int j = 0; j<10; ++j) p[j] = j*j;

for( int j = 0; j<10; ++j) cout << p[j] << (j<9 ? ", " : "\n");

// ...

delete[] p;             //если new[], то и delete[]
```

```
0, 1, 4, 9, 16, 25, 36, 49, 64, 81
```

C++ без классов, или "улучшенный" C

```
//9.3. --- новый тип указателей и операторы для работы с ними

struct point            //имя (тег) структуры является полноценным именем
{                       //нового типа и НЕ ТРЕБУЕТ обязательного (как в C)
        int x, y;       //указания перед ним ключевого слова struct
};                      //(это же относится к union и enum)

point pt = {15,51};     //просто point вместо struct point (без typedef)

point *sp = &pt;        //указатель на структуру point

int point :: * mp       //указатель на член структуры/класса
    = &point::x;        //бинарный :: - доступ к контексту структуры/класса

cout << "pt.x = "
     << pt .* mp;       //  .* - разадресация указателя на член структуры

mp = &point::y;         //бинарный :: - доступ к контексту структуры/класса

cout << ", pt.y = "
     << sp ->* mp       // ->* - разадресация указателя на член структуры
          << endl;
```

```
pt.x = 15, pt.y = 51
```

C++ без классов, или "улучшенный" C

```
// 9.4. --- новые операторы явного приведения типа

// static_cast<>()              - для приведений между арифметическими типами,
//                                между указателями или ссылками

a = static_cast<int>(::a);
vp = sp;
sp = static_cast<point *>(vp);

// const_cast<>()               - для отмены константности
p = const_cast<int *>(cp);

// прочие:
//       dynamic_cast<>()       - для приведений полиморфных указателей и ссылок
//       reinterpret_cast<>()   - для системно-зависимых приведений

// 9.5. --- определение типа объекта во время работы программы (RTTI)

#include<typeinfo>
cout << typeid(pt).name() << endl;
cout << ( typeid(*sp) == typeid(point) ) << endl;
```

```
struct point
1
```

C++ без классов, или "улучшенный" C

- // 10. --- Функции в C++ НЕЛЬЗЯ вызывать без предварительного определения
  // или объявления (в отличие от C)

- // 11. --- Функции с аргументами по умолчанию

```
void print(const int *array, int size, const char *delimeter = ", ");
print(arr, n);
print(arr, n, " : ");
```

```
41, 67, 34, 0, 69, 24, 78, 58, 62, 64
41 : 67 : 34 : 0 : 69 : 24 : 78 : 58 : 62 : 64
```

```
void print(const int *ar, int sz, const char *delim)          //значение по умолчанию задается
{                                                             //только в объявлении (прототипе)
    while(sz--) cout << *ar++ << (!sz ? "\n" : delim);
}
```

C++ без классов, или "улучшенный" C

```
double sqr(double, int = 2);
cout << "sqr(2) = " << sqr(2) << endl;
cout << "sqr(2,10) = " << sqr(2,10) << endl;
```

```
sqr(2) = 4
sqr(2,10) = 1024
```

- // 14. --- Связывание с кодом C

```
extern "C" {
    double sin(double);
    double cos(double);
    double pow(double,double);
}

double sqr(double x, int n)
{
    return pow(x, static_cast<double>(n));
}
```

C++ без классов, или "улучшенный" C

```
// 12. --- Перегрузка (имен) функций

int & max(int &, int &);
int max(int, int, int);
int max(const int *, int);

cout<<"Max from "<<a<<", "<<c<<" is "<< max(a, c) <<endl;

cout<<"Max from "<<arr[0]<<", "<<arr[1]<<", "<<arr[2]
        <<" is "<< max(arr[0], arr[1], arr[2]) <<endl;

cout<<"Max from\n";
print(arr, n);
cout<<" is "<< max(arr, n) <<endl;
```

```
Max from 12, 0 is 12
Max from 41, 67, 34 is 67
Max from
41, 67, 34, 0, 69, 24, 78, 58, 62, 64
 is 78
```

C++ без классов, или "улучшенный" C

- // 13. --- Встраивание функций

```
inline
int & max(int & a, int & b)
{
    return a > b ? a : b;
}

int max(int a, int b, int c)
{
    return max(max(a,b),c);
}

int max(const int *ar, int sz)
{
    int *p = const_cast<int *>(ar);          //снять const из-за max()
    int m = *p++;
    while(--sz) m = max(m,*p++);
    return m;
}
```

C++ без классов, или "улучшенный" C

- // 15. --- Шаблон функции

```
template<typename T>                    // T - параметр шаблона
void tswap(T & a, T & b)                //вместо typename часто
{                                       //используют слово class
    T t = a; a = b; b = t;
}

    tswap(a, c);
    cout << "a = " << a << ", c = " << c << endl;
```

```
a = 0, c = 12
```

```
    double x = 2.5, y = 5.2;
    tswap(x, y);
    cout << "x = " << x << ", y = " << y << endl;
```

```
x = 5.2, y = 2.5
```

```
    char c1 = 'A', c2 = 'Z';
    tswap(c1, c2);
    cout << "c1 = " << c1 << ", c2 = " << c2 << endl;
```

```
c1 = Z, c2 = A
```

C++ без классов, или "улучшенный" C

```
template<typename T>                    // T - параметр шаблона
void tswap(T & a, T & b)                //вместо typename часто
{                                       //используют слово class
    T t = a; a = b; b = t;
}
```

- // 15. --- Перегрузка шаблона функции

```
Inline void tswap(char *a, char *b)     //перегрузка шаблона
{                                       //для типа char *
    tswap(*a, *b);
}

    char pc[10] = "ABCDE", pd[10] = "ZYX";
    tswap(pc, pd);
    cout << "pc -> " << pc << ", pd -> " << pd << endl;
}
```

```
pc -> ZBCDE, pd -> AYX
```