Объектно-ориентированное программирование

МГТУ им. Н.Э. Баумана
Факультет Информатика и системы управления
Кафедра Компьютерные системы и сети
Лектор: д.т.н., проф.
Иванова Галина Сергеевна

Структура дисциплины ООП

Лекции – Объектно-ориентированное программирование на C++ в среде разработки Qt Creator (Clang) или Visual Studio 2017 Community – 51 час.

Семинары: 34 часа – 16-17 занятий

Лабораторные работы: 34 часа – 10 работ

Домашние работы: 3 задания (2+3+2 = 7 задач)

Баллы:

1 модуль:

5 неделя – **РК1 (Лаб. 4)** Матрицы **(**с лекциями) – 6..10 баллов;

8 неделя – **РК2** Динамические структуры данных – 15..25 баллов;

2 модуль:

13 неделя – **РК3** Иерархии классов – 15..25 баллов;

15 неделя – **ДЗ 3.2** Защита ДЗ – 6..10 баллов

Экзамен: - 18..30 баллов.

Проектно-технологическая практика

Три задания:

1. Pascal. Создание информационной системы – 6 неделя – 24..40

2. C++. Создание программной системы с элементарным интерфейсом – 12 неделя – 18..30

3. C++. Создание программной системы с Qt интерфейсом – 16 неделя – 18..30

Необходимо подготовить 2 экз. бланка задания и подписать самим и у преподавателя, затем один экз. отдать Ивановой Г.С.: 1-й экз. пойдет в отчет по практике, 2-й – останется у меня.

Контроль знаний – дифференцированный зачет по баллам:

60..70 – удовлетворительно;

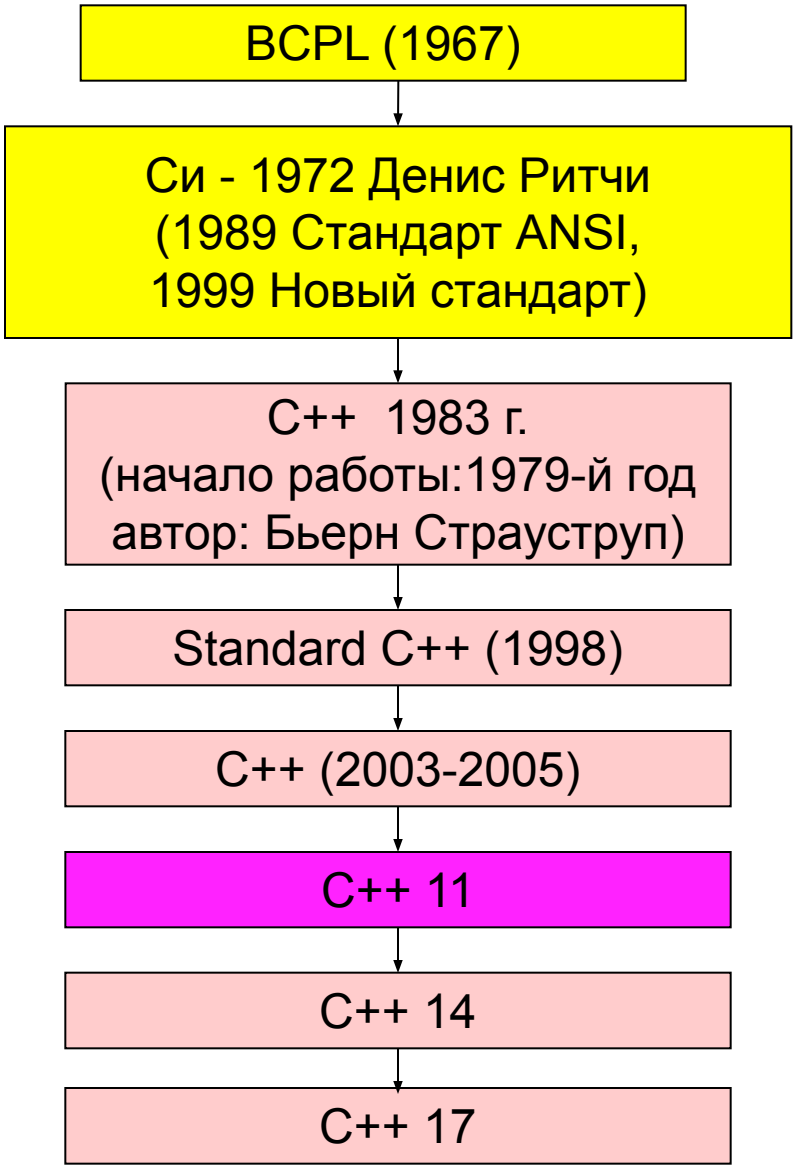
71..84 – хорошо;

85..100 – отлично

Литература

1. **Подбельский В.В. Стандартный Си++: Учеб. пособие. – М.: Финансы и статистика, 2008.**
2. Иванова Г.С., Ничушкина Т.Н. Объектно-ориентированное программирование. Учеб. для вузов. – М.: Изд-во МГТУ им. Н.Э. Баумана, 2014. 456 с.
3. **Иванова Г.С., Ничушкина Т.Н., Самарев Р.С. . С++. Часть 1. Средства процедурно-го программирования Microsoft Visual C++ 2008: Учебное пособие. – М.: МГТУ им. Н.Э. Баумана, 2010. – 126 с. – ЭУИ.**
4. **Иванова Г.С., Ничушкина Т.Н. С++. Часть 2. Объектно-ориентированное программирование на языке С++ в среде Visual Studio 2008: Учебное пособие. – М.: МГТУ им. Н.Э. Баумана, 2013. – 161 с. – ЭУИ.**
5. **Иванова Г.С. . С++. Часть 3. Создание графических интерфейсов пользователя с использованием библиотеки Qt 4.7: Учебное пособие. – М.: МГТУ им. Н.Э. Баумана, 2014. – 52 с. – В электронном виде.**
6. Шилдт Г. Полный справочник по C++, 4 изд. – М.: Изд. дом "Вильямс", 2009 (2015). – 800 с.
7. **Уроки С++.** URL: https://ravesli.com/uroki-cpp/ .
8. Шлее М. Qt 5.3. Профессиональное программирование на C++. СПб.: БХВ-Петербург, 2015. 928 с.
9. Дейтел Х., Дейтел П. Как программировать на C++.

История создания языка программирования C++



C++ - компилируемый статически типизированный язык программирования общего назначения (универсальный).

Первоначальное название - «C with Classes».

Основное достоинство – наличие большого количества специальных средств и механизмов, упрощающих написание сложных системных программ.

Основной недостаток – незащищенный синтаксис, который часто не позволяет точно идентифицировать ошибку на этапе компиляции программы.

Глава 1 Скалярные типы данных. Основные операторы C++

МГТУ им. Н.Э. Баумана
Факультет Информатика и системы управления
Кафедра Компьютерные системы и сети
Лектор: д.т.н., проф.
Иванова Галина Сергеевна

1.1 Алфавит языка. Структура программы

Алфавит языка C++ включает:

1) строчные и прописные латинские буквы: **`a, A, b, B`** и т.д.;
2) арабские цифры: **`0, 1, 2, 3, 4, 5, 6, 7, 8, 9`**;
3) шестнадцатеричные цифры: **`0..9, a..f`** или **`A..F`**;
4) специальные символы: **`+ - * / = ; { }`** и т. д.;
5) служебные слова: **`do, while, for, if, else`** и т. д.

В отличие от Паскаля C++, как и Си, различает строчные и прописные буквы.

Структура консольной программы

Консольная программа на C++ включает:

<Команды препроцессора>

[<Объявление типов, переменных и константа>]

[<Объявления (прототипы) функций>]

<Описание функции **main()**>

[<Описания других функций>]

Используемая нотация:
<...> - некоторая конструкция C++;
[<...>] – присутствие конструкции в операторе не обязательно

В C++ все программы/подпрограммы называются *функциями*.

Функция ***main()*** – основная программа, всегда имеющая имя main, принимающая управление от операционной системы и возвращающая его ей. Присутствие этой функции (функции WinMain() – для Windows или др. со словом main) является обязательным.

Команды ***препроцессора*** – команды, выполняемые перед компиляцией программы, могут использоваться для подключения необходимых библиотек.

Описание функции

<Тип результата или **void**> <Имя функции> ([<Список параметров>])

{[< Объявление локальных переменных и констант >]

<Операторы>

}

Если функция возвращает скалярное значение, то первое слово описания содержит его тип. Если функция не возвращает значения, т.е. фактически является процедурой, то указывают тип **`void`** – «пустой» тип.

Независимо от наличия или отсутствия параметров скобки после имени функции должны быть указаны обязательно. При отсутствии параметров в скобках может быть указано **`void`** или ничего.

{...} – операторные скобки, ограничивающие тело функции – аналог **`begin... end`** Паскаля.

Сравнение программ, написанных на Pascal и C++

Free Pascal (базовая объектная модель):

```
Program primer;
Uses SysUtils;

Var   A:integer=18;
      B:integer=24;
      C:integer;

Function Nod(A,B:integer):integer;
Begin
   while A<>B do
     if A>B then A:=A-B
     else B:=B-A;
   Nod:=A;
End;

Begin
   C:=Nod(A,B);
   Writeln('Nod=', C);
End.
```

Qt Creator (clang) C++ - Пример Ex1_01:

```
#include <iostream>
using namespace std;

int nod(int a,int b)
{
   while (a!=b)
     if  (a>b) a=a-b;
     else b=b-a;
   return a;
}
int main(int,char**)
{
   int a=18,
       b=24,
       c;

   c=nod(a,b);
   cout << "nod=" << c << '\n';
   return 0;
}
```

Команда препроцессора

Подключение адресного пространства

Описание функции

Основная функция main()

Объявление переменных

1.2 Фундаментальные типы данных

Имя типа	Подтипы	Размер, байт	Значения
char	**[signed] char** **unsigned char**	1	-128..127 0..255
wchar_t	**[signed] wchar_t** **unsigned wchar_t**	2	-32768..32767 0..65535
short	**[signed] short** **unsigned short**	2	-32768..32767 0..65535
int или **long**	**[signed] int** **unsigned int**	4	-2^{31}.. $2^{31}-1$ 0.. $2^{32}-1$
long long	**[signed] long long** **unsigned long long**	8	-2^{63}.. $2^{63}-1$ 0.. $2^{64}-1$
bool		1	false (0), true(1)

Для совместимости с Си по-прежнему считается: 0 – false; не 0 – true.

2. Вещественные типы

Тип	Размер, байт	Значащих цифр	Минимальное положительное число	Максимальное положительное число
float	4	6	1.175494351e-38	3.402823466e38
double (long double)	8	15	2.2250738585072014 e-308	1.797693134862318 e308

3. Неопределенный («пустой») тип void

Нельзя объявлять значения типа void, этот тип используется только при объявлении:

- нетипизированных указателей;
- функций, не возвращающих значений (процедур).

1.3 Объявление переменных и констант

Формат:

[<Изменчивость>] <Тип><Список идентификаторов>[=<Значение>];

где <Изменчивость> – описатель возможности изменения значений:

`const` – константа – неизменяемое значение,

`volatile` – независимо меняющаяся переменная,

без указания изменчивости – обычная переменная

<Тип> – описатель типа: **`int, char, float, double`** и т.д.;

<Список идентификаторов> – список имен переменных или констант;

<Значение> – начальное значение переменной или значение константы.

Примеры:

а) `int a, b;` // две целые переменные

б) `float c=1.05,d(3.5),e{1.1};` // инициализированные перем.

в) `const unsigned char letter='a';` // константа – код буквы «а»

г) `int const a=15;` // целая константа 15

д) `const int a(1);` // целая константа 1

Перечисляемый тип (Си)

Используется для объявления совокупности поименованных констант.

Формат:

enum [class <имя>] {<Ид>[=<Целое>] [,<Ид>[<>]…]}
[<Список переменных>];

Примеры:

```
enum class Options {None, One, All};
Options o = Options::All;

enum {SUN, MON, TUES, FRI=5, SAT} day;
```

Имя переменной

SUN =0, MON = 1, TUES = 2, FRI=5, SAT=6

Задание типа позволяет ограничить область видимости перечисленных значений (C++ 11).

Объявление типа

Появилось только в C++.

Формат:

typedef <Описание типа> <Имя объявляемого типа>;

Примеры:

Имя нового типа

1) `typedef unsigned int word;`

2) `typedef enum {false, true} boolean;`

Имя нового типа

1.4 Простейший ввод/вывод
1.4.1. Ввод-вывод с помощью функций Си
1 Форматный ввод /вывод

Ввод:

`int scanf`(<Форматная строка>, <Список адресов переменных>);

// возвращает количество значений или **EOF**(-1)

(VS)`int scanf_s`(<Форматная строка>, <Список адресов переменных с указанием размера буфера для символов и строк>);

Вывод:

`int printf`(<Форматная строка>, <Список выражений>);

где <Форматная строка> - строка, которая помимо символов содержит спецификации формата для выводимых значений вида:

%[-] [<Целое 1>].[<Целое 2>] [h/l/L]<Формат>

«-» - выравнивание по левой границе,

<Целое 1> - ширина поля вывода;

<Целое 2> - количество цифр дробной части вещественного числа;

h/l/L - модификаторы формата;

<Формат> - определяется специальной литерой.

Спецификации формата

d,i - целое десятичное число (int);

u - целое десятичное число без знака (unsigned int);

o - целое число в восьмеричной системе счисления;

x,X - целое число в шестнадцатеричной системе счисления, % 4x - без гашения незначащих нулей, X – буквы верхнего регистра;

f - вещественное число float в форме с фиксированной точкой;

e,E - вещественное число float в форме с плавающей точкой;

g,G - вещественное число float в одной из указанных выше форм;

c - символ;

p - указатель (адрес) в шестнадцатеричном виде;

s - символьная строка.

Кроме этого, форматная строка может содержать:

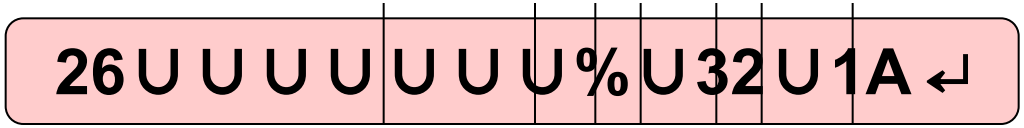
\n - переход на следующую строку;

\n hhh - вставка символа с кодом ANSI hhh (код задается в шестнадцатеричной системе счисления);

%% - печать знака %.

Примеры форматного ввода/вывода

а) `int i=26;`
`printf ("%-6dUUU%%U %oU %X\n", i, i, i);`



б) `scanf("%d %d",&a,&b);`
Вводимые значения: 1) **24 28** 2) **24**↵
28

в) `scanf("%d,%d",&a,&b);`
Вводимые значения: **24,28**

Ввод строки до пробела

г) `scanf("%s",name);`
Вводимые значения: **Иванов Иван**
Результат ввода: name="**Иванов**"

В параметрах функции указаны размеры буферов

д) `int i; char ch, name[20];`
`scanf_s("%d %s %c",&i,name,20,&ch,1);`

Модификаторы формата

Модификаторы употребляются с некоторыми форматами для указания типов переменных, отсутствовавших в первых версиях C++

Модификатор	Спецификатор формата	Тип переменной
h	i d u o x X	short
l	i d u o x X	long
l	e E f g G	double
L	e E f g G	long double

Примеры:

```
short s1; long L1; double d1;
printf("input short>"); scanf("%hd", &s1);
printf("input long >"); scanf("%ld", &L1);
printf("input double>"); scanf("%lf", &d1);
```

Ограничение набора вводимых символов при вводе строк

%[<Множество символов>] - можно вводить только указанные символы, при вводе другого символа ввод завершается

%[^<Множество символов>] – можно вводить все символы, кроме указанных

Примеры:

%[A-Z] - все заглавные английские буквы;

%[0-9A-Za-z] - все десятичные цифры и все буквы английского алфавита;

%[A-FT-Z] - все заглавные буквы от A до F и от T до Z;

%[-+*/] - четыре арифметических операции;

%[z-a] - символы 'z', '-' (минус) и 'a';

%[+0-9-A-Z] - символы '+', '- 'и диапазоны 0-9 и A-Z;

%[+0-9A-Z-] - то же самое;

%[^-0-9+A-Z] - все символы, кроме указанных.

```
scanf_s("%[A-Z]",st,20);  //ввод до другого символа
```

если ввести ABCD20, то st="ABCD"

2 Ввод/вывод строк

Ввод:

char* gets(<Строковая переменная>);

// возвращает копию строки или NULL

(VS) char* gets_s(<Строковая переменная>,<Размер буфера>);

Вывод:

int puts(<Строковая константа или переменная>);

Примеры:

а) `puts("Это строка");`

Результат: **Это строка**↵

б)
```
char st[21];
gets(st);
```

Вводимые значения: **Иванов Иван**↵

Результат: **st ="Иванов Иван"**

Ввод строки до маркера "конец строки"

в)
```
char st[21];
gets_s(st,20); // один байт для хранения '\0'
```

3 Ввод/вывод символов

Ввод символа:

`int getchar();` // возвращает символ или **EOF**

Вывод символа:

`int putchar(`<Символьная переменная или константа>`);`

Примеры:

а) `ch=getchar( );`

б) `putchar('t');`

1.4.2 Ввод-вывод с использованием библиотеки классов C++

Операции ввода-вывода с консолью могут осуществляться с использованием специальной библиотеки классов C++.

Для осуществления операций ввода-вывода с консолью необходимо подключить библиотеку **`iostream`**, содержащую описание этих классов, и разрешить программе использовать стандартное адресное пространства **`std`**, в котором работают все старые библиотеки C++:

```
//include <iostream>
using namespace std;
```

Вывод на экран

Операция вывода на экран компьютера предполагает вставку данных в стандартный поток вывода.

Стандартный поток вывода на экран

```
cout >> <Имя скалярной переменной, строки или константы
          значений или строк>;
```

Операция вставки в поток

С помощью операции вставки в поток можно выводить данные следующих типов: **int, short, long, double, char, bool, char * (строки Си) и т.п.**

Примеры вывода на экран

1) вывод строковых констант, чисел и логических значений:

```
int a=3;  float b=5.34; bool c=true;
cout<< "Results: a=" << a << " b=" << b <<
       " c=" << c << "." << '\n';
```

```
Results: a=3 b=5.34 c=1.↲
```

переход на следующую строку, можно также использовать endl

2) вывод в две строки:

```
cout<< "Results: a=" << a << " b=" << b <<'\n'<<
       " c=" << c << "." << '\n';
```

```
Results: a=3 b=5.34↲
c=1.↲
```

Управление выводом. Манипуляторы

Манипуляторы – специальные методы классов ввода-вывода, предназначенные для управления операциями ввода-вывода. Они непосредственно "вставляются" в поток.

Манипуляторы бывают с параметрами и без. Для использования манипуляторов с параметрами необходимо подключить библиотеку **iomanip**:

```
#include <iomanip>
```

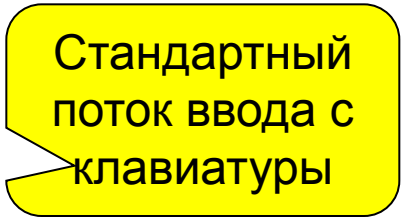
1) **setw(int n)-** устанавливает ширину поля печати n;
2) **setprecision(int n)** - устанавливает размер дробной части числа n (вместе с точкой);

```
int a=3; double b=-5.543;
cout << setw(8) << a << endl;
cout << setw(8) << setprecision(2) << b << endl;
```

```
       3↵
    -5.5↵
```

Ввод с клавиатуры

Операция ввода с клавиатуры программируется как операция извлечения из потока.




```
cin >> <имя скалярной переменной или строки>;
```

Можно вводить целые и вещественные числа, символы, строки, булевские значения: int, long, double, char, char *(строки) и т.д.

Значения (кроме символов) следует разделять пробелами и/или маркерами перехода на следующую строку. Символы при вводе не разделяются.

Примеры ввода с клавиатуры

1) ввод чисел:

```
int a;  float b; bool c;
cout << "Enter a, b, c: ";
cin >> a >> b >> c;
```

```
Enter a, b, c: 3 5.1 true↵
```

```
Enter a, b, c: ↵
3↵
5.1↵
true↵
```

2) ввод символов:

```
char ch1,ch2;
cout<< "Enter ch1, ch2: ";
cin >> ch1 >> ch2;
```

```
Enter ch1, ch2: ab↵
```

3) пропуск Enter перед вводом символов и строк:

```
cin >> n;
cin.ignore(2,'\n');
```

```
Enter n: 4↵
```

1.5 Операции

- **Арифметические**:

  + – сложение;

  - – вычитание;

  * – умножение;

  / – деление – результат – вещественное, если хотя бы одно из чисел – вещественное, результат – целое, если делимое и делитель – целые,

  % - остаток от деления целых чисел.

- **Логические**: **! (**не**), && (**и**), || (**или**).**

- **Логические поразрядные**:

  - (не), **&** (и), **|** (или), ^ (исключающее или).

- **Отношения**:

  <, **>**, **<=**, **>=**, **= =** (равно), **! =** (неравно).

Операции (2)

- **Сдвиги**:
  <идентификатор> >> <идентификатор> - сдвиг вправо,
  <идентификатор> << <идентификатор> - сдвиг влево.

- **Порядковые**:

  ++<идентификатор>, <идентификатор>++ (следующее);

  - -<идентификатор>, <идентификатор> - - (предыдущее).

- **Присваивания:**

  = += -= *= /= %= &= ^= |= <<= >>=

- **Условная:**

  <Выражение1>?<Выражение2>:<Выражение3>

Приоритет операций

1. **( ) [ ] -> :: .**
2. **!** (не) **+ - ++ -- &(**адрес**) *(**указатель**) sizeof new delete**
3. **.* ->***
4. *** / %**
5. **+ -** (бинарные)
6. **<< >>**
7. **< <= > >=**
8. **= = ! =**
9. **&**(поразрядное и)
10. **^**(исключающее или)
11. **|** (поразрядное или)
12. **&&**
13. **||**
14. **?:**
15. **= *= /= %= += -= &= ^= |= <<= >>=**
16. **,**

1.6 Выражение

Формат составного выражения:

<Выражение1>[,<Выражение2>,...[,<Выражение n>]...]

Примеры:

а) `int a=10, b=3; float ret; ret=a/b;` — ret=3

б) `c=1; b=c++;` — b=1, c=2

в) `c=1; sum=++c;` — c=2, sum=2

г) `c = a << 4;` — эквивалентно **c=a*16;**

д) `a+=b;` — эквивалентно **a=a+b;**

е) `a=b=5;` — эквивалентно **b=5; a=b;**

ж) `c=(a=5, b=a*a);` — эквивалентно **a=5; b=a*a; c=b;**

з) `a=(b=s/k)+n;` — эквивалентно **b=s/k; a=b+n;**

и) `c=(a>b)?a:b;` — если a>b, то c=a, иначе c=b

1.7 Блок операторов

Блок операторов используется в конструкциях ветвления, выбора и циклов, предусматривающих один оператор.

Формат:

{ <Оператор>;… <Оператор>;}

Пример:

```
{
 f = a + b;
 a += 10;
}
```

Точка с запятой в отличие от Паскаля является частью оператора, а потому не может опускаться перед фигурной скобкой.

1.8 Оператор условной передачи управления

if (<Выражение>) <Оператор;> [else <Оператор;>]

Примеры:

а)
```
if (!b)
    cout<<"c - не определено"<<endl; // если b=0, то – ошибка,
  else {c=a/b; cout<<"c="<<c<<endl);} // иначе - выводится с.
```

б)
```
if ((c=a+b)!=5) c+=b;
  else c=a;
```

в)
```
if ((cin>>ch, ch=='q')              // если в ch введено q,
    cout<<"Программа завершена. "<< endl;  //  то ...
 else cout<<"Продолжаем работу... "<< endl;  // иначе ...
```

г)
```
ch='a';
 if ((oldch=ch, ch='b')=='a')cout<<"Это символ 'a'\n";
 else cout<<"Это символ 'b'\n";
```

1.9 Оператор выбора

```
switch (<выражение>)
{        case <элемент>: <операторы;>
         case <элемент>: <операторы;>
                  ...
         [ default  : <операторы;>]
}
```

Пример:

```
switch (n_day)
{ case 1:
  case 2:
  case 3:
  case 4:
  case 5: cout << "Go work!" << endl; break;
  case 6: cout << "Clean the yard and";
  case 7: cout << "relax!" << endl;
}
```

1.10 Операторы циклов

1. Оператор цикла while

while (<Выражение>) <Оператор>

Пример Ex1_02. Вычислить при x > 1 сумму ряда $S=1+1/x-1/x^2+1/x^3-...$ с точностью ξ.

```
#include <stdio.h>
#include <math.h>
int main(int, char**)
{
    double s, r, x, eps;
    puts("Input x, eps:");
    scanf("%lf  %lf", &x, &eps);
    if (x <= 1)  puts("Error.");
    else
    {
        s = 1;   r = 1 / x;
        while (fabs(r) > eps) {s += r;   r = -r / x;}
        printf("Result= %lf.\n", s);
    }
```

2. Оператор цикла for

for (<Выражение1>;<Выражение2>;<Выражение3>)<Оператор>;

Эквивалентно:

```
 <Выражение1>
while (<Выражение2>)
   {<Оператор>;
    <Выражение3>;
   }
```

Пример Ex1_03. Вычислить сумму первых десяти натуральных чисел.

```
#include <iostream>
using namespace std;

int main()
{  int i,s;
   for (i=1,s=0;i<=10;i++) s+=i;
   cout << "Sum=" << s << endl;
   return 0;
}
```

3. Оператор цикла do ... while

do <Оператор > while (<Выражение>) ;

Пример. Игнорировать ввод значения, выходящего за пределы заданного интервала.

```
do {
   cout<<"Enter a in ["<<low<<","<<high<<"]\n";
   cin >> a;
   }
 while (a<low || a>high);
```

1.11 Неструктурные операторы передачи управления

1. Оператор безусловного перехода goto

goto <Метка перехода>;

Пример:

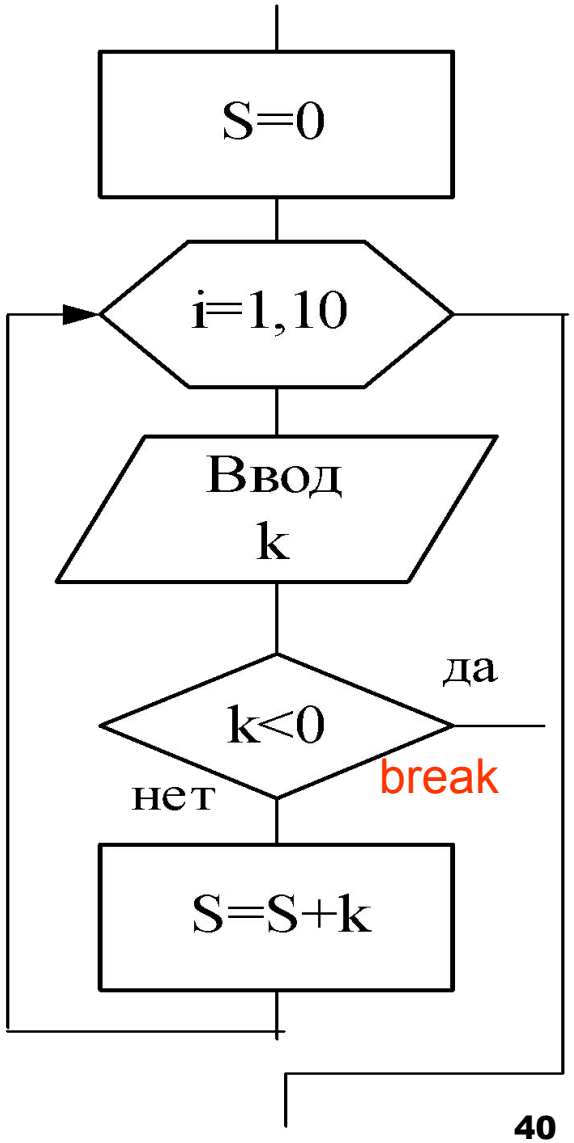
```
again: x=y+a;
                ...
       goto again;
```

2. Оператор досрочного завершения break

break;

Пример Ex1_04. Суммирование до 10 чисел вводимой последовательности. При вводе отрицательного числа работа программы завершается.

```
#include <iostream>
using namespace std;
int main()
{   int s=0,k;
    cout<<"Input up to 10 numbers."<<endl;
    for (int i=1; i<11; i++)
    {
      cin >> k;
      if (k < 0) break;
      s+=k;
    }
    cout << "Result =" << s << endl;
    return 0;
}
```

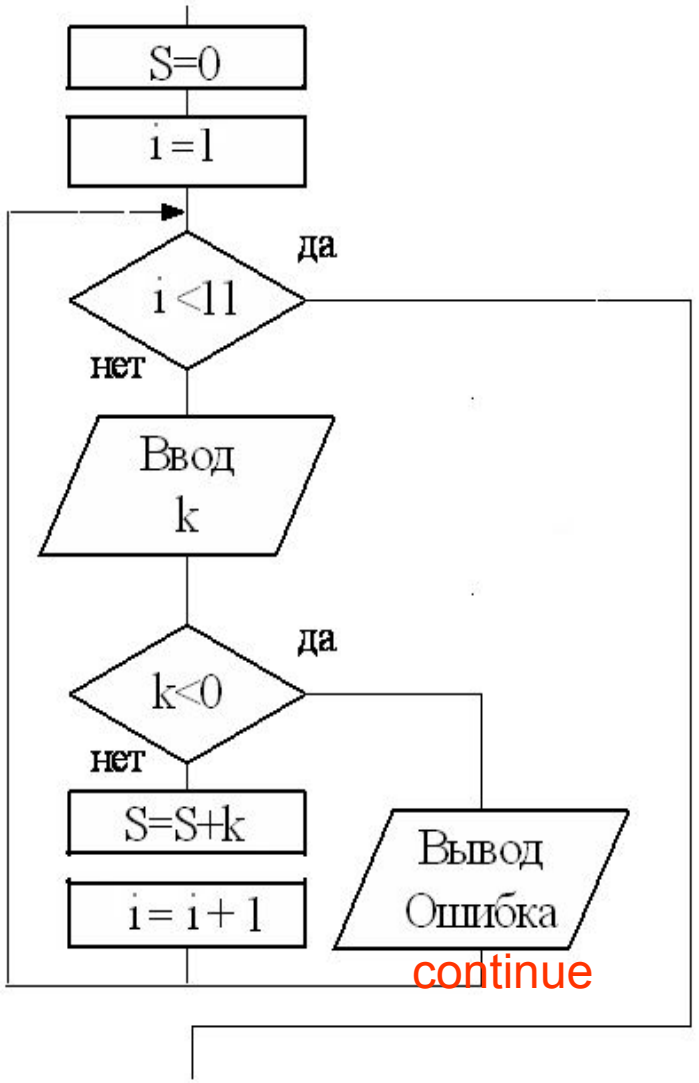


3. Оператор продолжения continue

continue;

Пример Ex1_05. Программа суммирует 10 целых положительных чисел.

```
#include <iostream>
using namespace std;
int main()
{   int s=0,i=1,k;
    cout <<"Enter 10 numbers." << endl;
    while(i<11)
    {   cin >> k;
        if (k<0){   cout<<"Error."<<endl;
                    continue;
                }
        s+=k; i++;
    }
    cout << "Result =" << s << endl;
    return 0;
```



Пример Ex1_06. Вывод таблицы кодов

```
32-  33-! 34-" 35-#;
36-$ 37-% 38-& 39-'.
```

```
#include <iostream>
using namespace std;
int main()
{   int i,i1,in,c;
    cout<<"Enter first and last values."<<endl;
    cin >> i1 >> in;
    cout<<"Enter number of colons: " << endl;
    cin >> c;
    for(i=i1;i<=in;i++)
      if (i<in)
         cout << i << "- "<< (char)i <<
                 (char)(((i-i1+1)%c!=0)?' ':'\n');
      else cout << i << " - " << (char) i << '.' << endl;
    return 0;
}
```