Тема 2
Основы языка программирования C++

Структура программы на языке C++

В состав программы входят:

- директивы препроцессора
- операторы (инструкции)
- комментарии

Пример программы на C++

```
// программа вводит с консоли число
// и выводит его на консоль
#include <iostream>
using namespace std;
int main()
{
   int a;
   cout << "input number\n";
   cin >> a;
   cout << "number = " << a << "\n";
   return 0;
}
```

Пример программы на C++

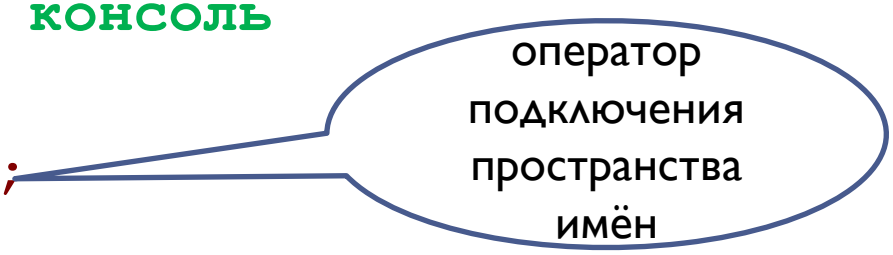
```
// программа вводит с консоли число
// и выводит его на консоль
#include <iostream>
using namespace std;
int main()
{
   int a;
   cout << "input number\n";
   cin >> a;
   cout << "number = " << a << "\n";
   return 0;
}
```

Пример программы на C++

```
// программа вводит с консоли число
// и выводит его на консоль
#include <iostream>
using namespace std;
int main()
{
   int a;
   cout << "input number\n";
   cin >> a;
   cout << "number = " << a << "\n";
   return 0;
}
```

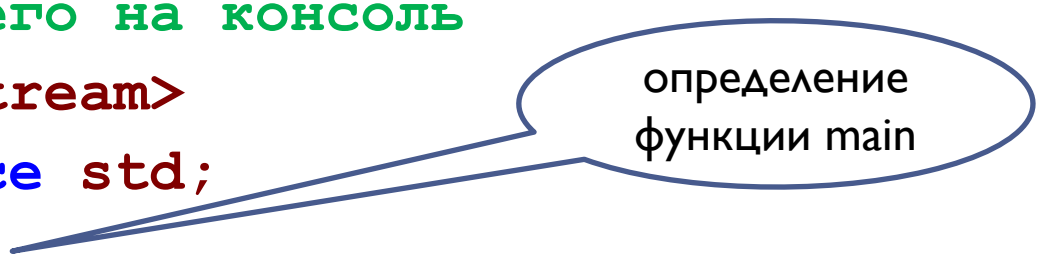
Пример программы на C++

```
// программа вводит с консоли число
// и выводит его на консоль
#include <iostream>
using namespace std;
int main()
{
   int a;
   cout << "input number\n";
   cin >> a;
   cout << "number = " << a << "\n";
   return 0;
}
```



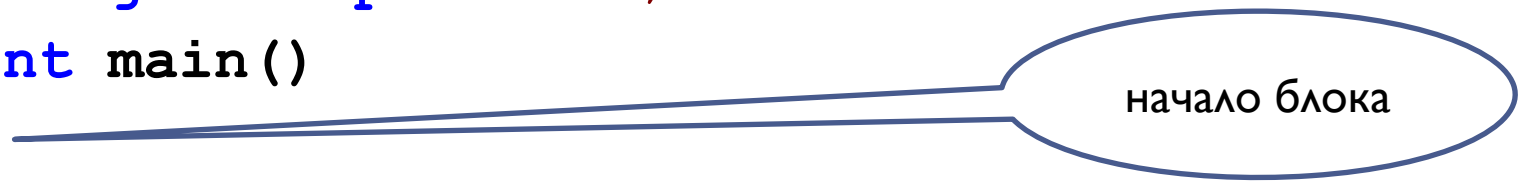
Пример программы на C++

```
// программа вводит с консоли число
// и выводит его на консоль
#include <iostream>
using namespace std;
int main()
{
   int a;
   cout << "input number\n";
   cin >> a;
   cout << "number = " << a << "\n";
   return 0;
}
```



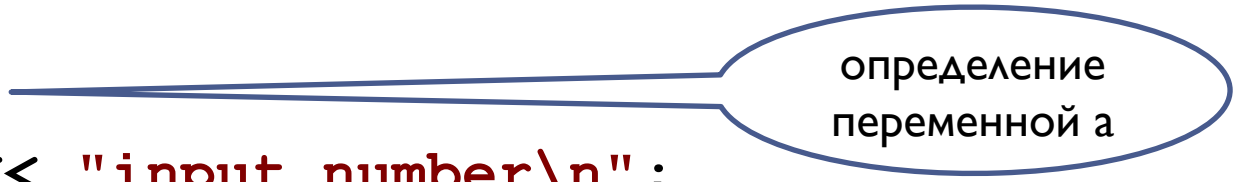
Пример программы на C++

```
// программа вводит с консоли число
// и выводит его на консоль
#include <iostream>
using namespace std;
int main()
{
   int a;
   cout << "input number\n";
   cin >> a;
   cout << "number = " << a << "\n";
   return 0;
}
```



Пример программы на C++

```
// программа вводит с консоли число
// и выводит его на консоль
#include <iostream>
using namespace std;
int main()
{
   int a;
   cout << "input number\n";
   cin >> a;
   cout << "number = " << a << "\n";
   return 0;
}
```

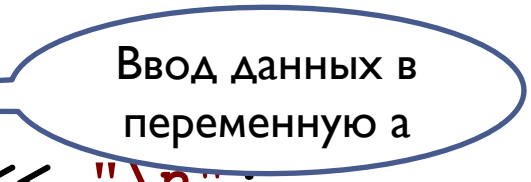


Пример программы на C++

```
// программа вводит с консоли число
// и выводит его на консоль
#include <iostream>
using namespace std;
int main()
{
   int a;
   cout << "input number\n";
   cin >> a;
   cout << "number = " << a << "\n";
   return 0;
}
```

Пример программы на C++

```
// программа вводит с консоли число
// и выводит его на консоль
#include <iostream>
using namespace std;
int main()
{
   int a;
   cout << "input number\n";
   cin >> a;
   cout << "number = " << a << "\n";
   return 0;
}
```

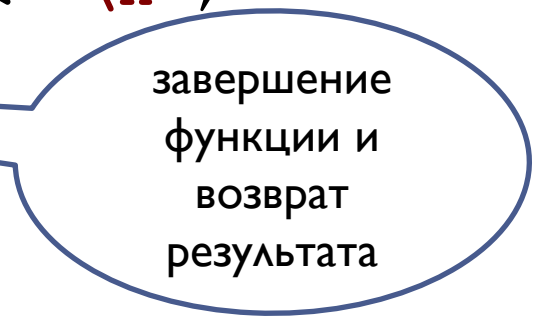


Пример программы на C++

```
// программа вводит с консоли число
// и выводит его на консоль
#include <iostream>
using namespace std;
int main()
{
   int a;
   cout << "input number\n";
   cin >> a;
   cout << "number = " << a << "\n";
   return 0;
}
```

Пример программы на C++

```
// программа вводит с консоли число
// и выводит его на консоль
#include <iostream>
using namespace std;
int main()
{
   int a;
   cout << "input number\n";
   cin >> a;
   cout << "number = " << a << "\n";
   return 0;
}
```



Пример программы на C++

```
// программа вводит с консоли число
// и выводит его на консоль
#include <iostream>
using namespace std;
int main()
{
   int a;
   cout << "input number\n";
   cin >> a;
   cout << "number = " << a << "\n";
   return 0;
}
```

конец блока

Целочисленные типы данных C++

- **int** (целый, его размер определяется компилятором, обычно – 2 или 4 байта);
- **char** (символьный, как правило, 1 байт);
- **wchar_t** (предназначен для хранения набора символов, для которых недостаточно 1 байта, например, для кодировки Unicode. Как правило, занимает 2 байта);
- **bool** (предназначен для хранения логических величин, 0 интерпретируется как false, а любое ненулевое значение – как true. при преобразовании из типа **bool** к другому значение true переводится в 1);

Кроме того, можно указывать *спецификаторы типа*, которые уточняют внутреннее представление и диапазон значений типа:

- **short**;
- **long**;
- **long long**;
- **signed**;
- **unsigned**.

Другие стандартные типы данных C++

- **float** (числа с плавающей точкой длиной 4 байта);
- **double** (числа с плавающей точкой длиной 8 байт);
- **long double** (числа с плавающей точкой длиной 10 байт);
- **void** (т.н. ”пустой” тип, используется для определения функций, которые не возвращают значений или не имеют аргументов, обозначения абстрактных указателей и для некоторых других целей).

Самоопределённые константы (литералы)

Тип константы	Пример
Целая десятичная	`8, 0, -5, 4U, 3000L`
Целая восьмеричная	`077, 0111`
Целая шестнадцатеричная	`0xFFA2, 0X00FF`
Вещественная	`3.5, 0.2e6, .1E10`
Символьная	`'s', 'П', '\n', 'ss', '\0xFF', '\077'`
Строковая	`"Здесь был я", "ЗАО \"МММ\"", "C:\\autoexec.bat"`

Запись длинных строковых констант

```
"Эта строковая константа размещена \_
на нескольких строках \_
программы"
```

Красным цветом обозначен невидимый пробел!

Переменные

Переменная – это именованная область памяти, в которой хранятся данные определенного типа. Оператор описания переменной в общем случае выглядит так:

[класспамяти] [**const**] тип {имя [инициализатор]}…;

Класс памяти может быть задан с помощью слов **auto**, **extern**, **static** либо **register**.

Модификатор **const** позволяет задать именованные константы времени выполнения.

Инициализатор позволяет присвоить начальное значение переменной (и обязателен при описании константы). Его можно записать в двух формах:

= *значение*

или

(*значение*)

Значение выражения в инициализаторе должно быть вычислимо в процессе компиляции!

Примеры определения переменных

- `short int a = 1;`
- `const char CR = '\n';`
- `char s, sf('a'), st = '1';`
- `static unsigned int P;`

Область действия идентификатора

Область действия переменной (а в общем случае – любого идентификатора) – это часть программы, в которой его можно использовать.

Переменная, описанная внутри блока, считается локальной. Область ее действия – от точки описания до конца блока, включая вложенные блоки.

Переменные, описанные вне блоков, являются глобальными, их область действия – до конца файла.

Переменные, описанные в операторе **for**, действуют до конца этого оператора.

Область видимости переменной

Область видимости чаще всего совпадает с областью действия, однако если во вложенном блоке описана переменная с уже существующим именем, то это делает невидимой внешнюю переменную. Тем не менее, к глобальной невидимой переменной можно обратиться с использованием операции :: (расширение области видимости).

Время жизни переменной

Время жизни переменной определяет, как долго сохраняется ее значение. Значение локальных переменных теряются при выходе из блока, в котором они описаны, если только эти переменные не были описаны с классом памяти **static**. Напротив, время жизни глобальных переменных и переменных с классом памяти **static** – постоянное.

Объявление и определение переменных (примеры)

```
int a;
int main() {
 int b;
 extern int x; //объявление, но не определение!
 a = 1;   // обращение к глобальной переменной
 float a; // а теперь она становится невидимой
 a = 2.5; // обращение к локальной переменной
 ::a = 4; // обращение к глобальной переменной
// за счет расширения области видимости
 return 0;
}
int x = 4; // а вот теперь x определена!
```

Операции

- Операции содержат знак операции (иногда слово) и операнды.
- Результат вычисления операции может быть использован далее при вычислении выражений.
- Некоторые операции изменяют значение одного из операндов *(в дальнейшем такие операнды обозначены красным)*

Арифметические операции

Запись операции (a, b, c – операнды)	Описание операции
a+ b	сумма
a – b	разность
a * b	произведение
a / b	частное (если оба операнда целые, выполняется целочисленное деление)
a % b	остаток от деления
a++	постфиксный инкремент (результат – старое значение операнда)
++a	префиксный инкремент (результат – новое значение операнда)
a – –	постфиксный декремент
– – a	префиксный декремент
– a	унарный минус – изменение знака

Присваивание и связанные с ним операции

Запись операции (a, b, c – операнды)	Описание операции
a = b	присваивание: a получает значение b
a += b	то же, что a = (a + b)
a –= b	то же, что a = (a – b)
a *= b	то же, что a = (a * b)
a / =b	то же, что a = (a / b)
a %= b	то же, что a = (a % b)

Операции сравнения

Запись операции (a, b, c – операнды)	Описание операции
a == b	Возвращает истину, если a равно b
a != b	Возвращает истину, если a не равно b
a < b	Возвращает истину, если a меньше b
a <= b	Возвращает истину, если a меньше или равно b
a > b	Возвращает истину, если a больше b
a >= b	Возвращает истину, если a больше или равно b

Таблицы истинности для логических операций

- Отрицание

операнд	false	true
результат	true	false

- «И» (and)

операнды	false	true
false	false	false
true	false	true

- «Или» (or)

операнды	false	true
false	false	true
true	true	true

- Исключающее «или» (xor)

операнды	false	true
false	false	true
true	true	false

Логические и побитовые операции

Запись операции (a, b, c – операнды)	Описание операции
! a	Отрицание. Возвращает истину, если a – ложь, и наоборот
a || b	Логическое «или». Возвращает истину, если a или b истинно
a && b	Логическое «и». Возвращает истину, если a и b истинно
a | b	Побитовое «или». Операция «или» выполняется для всех битов операндов
a & b	Побитовое «и». Операция «и» выполняется для всех битов операндов
a ^ b	Побитовое «исключающее или». Операция «и» выполняется для всех битов операндов
~a	Инверсия битов операнда
a << b	Сдвиг значения a на b битов влево

Другие операции

Запись операции (a, b, c – операнды)	Описание операции
sizeof (a) sizeof (тип)	Возвращает размер операнда в памяти (в байтах)
(тип) a тип (a)	Преобразовывает значение операнда к указанному типу
a << b	Вывод значения b в поток a
a >> b	Чтение из потока a и помещение результат в b
a ? b : c	Условная операция. Если a истинно, возвращается значение b, в противном случае – c
a, b	Вычисляется выражение a, а затем b. Значение первого выражения теряется.

L-value выражения

Если операция изменяет значение одного из своих операндов, то не всякое выражение может служить изменяемым операндом таких операций. На этом месте должна стоять конструкция, однозначно адресующая некоторый участок памяти, значение которой можно изменить (например, имя переменной). Подобного рода выражения получили название L-выражений (L-values).

Результаты операций присваивания и префиксного инкремента (декремента) являются L-values, других рассмотренных операций – не являются.

Неполное вычисление логических выражений

Логические операции выполняются слева направо.

Если значения первого операнда достаточно, чтобы определить результат операции, второй операнд не вычисляется.

Пример:

```
a=0, b=4;
```

В выражении `(a>0) && (b==4)` будет вычислена только левая часть.

Выражение `(b==4) && (a>0)` будет вычислено полностью.

Преобразование типов

В выражение могут входить операнды различных типов. Если операнды имеют одинаковый тип, то результат операции будет иметь тот же тип. Если операнды разного типа, перед вычислениями автоматически выполняются неявные преобразования типов. Обычно короткие типы приводятся к более длинным (upcast), что обеспечивает сохранение значимости и точности:

```
( char, short ) -> int -> long -> long long
  -> float -> double -> long double
```

Сужающие преобразования типов

Однако в выражениях с операцией присваивания встречается и обратный вариант, когда более "объемный" тип присваивается менее "объемному", например, float в int (downcast). Подобные преобразования связаны с потерей информации (дробная часть отбрасывается), поэтому называются сужающими приведениями (преобразованиями). Компилятор предупреждает о таких операциях.

Пример:

```
int a; float b;
…
a=b; // компилятор выдаст предупреждение!
a=floor(b) // теперь предупреждения нет!
```

Операторы C++

- Оператор-выражение

```
выражение;
```

- Оператор проверки условия

```
if (выражение) оператор1;
[else оператор2;]
```

- Оператор switch

```
switch (целое_выражение){
case константное_выражение_1: операторы_1;
[case константное_выражение_2: операторы 2;] ...
[default: операторы ;]
}
```

Пример работы оператора switch

```
#include <iostream>
using namespace std;
int main() {
  int a, b, res; char op;
  cout << "\nВведите 1й операнд : "; cin >> a;
  cout << "\nВведите знак: "; cin >> op;
  cout << "\nВведите 2й операнд : "; cin >> b;
  bool f = true;
  switch (op) {
    case '+': res = a + b; break;
    case '-': res = a - b; break;
    case '*': res = a * b; break;
    case '/': res = a / b; break;
    default : cout << "\nНеизвестная операция";
       f = false;
  }
  if (f)
    cout << "\nРезультат : " << res;
}
```

Операторы цикла C++

- Цикл с предусловием

```
while (выражение) оператор;
```

- Цикл с постусловием

```
do оператор while (выражение);
```

- Оператор for

```
for (выражение1; выражение2; выражение3) оператор;
```

- выражение1 вычисляется один раз перед входом в цикл;
- выражение2 вычисляется перед каждой итерацией; если оно ложно, выходим из цикла;
- выражение3 вычисляется после выполнения каждой итерации

Пример использования цикла for

```
int M[100], t, i, j;
// ввод массива
for (i = 0, j = 99; i<j; i++, j--) {
 t = M[i];
 M[i] = M[j];
 M[j] = t;
}
```

Операторы передачи управления

В C++ есть пять операторов, изменяющих естественный порядок выполнения вычислений:

- оператор выхода из цикла и переключателя **`break`**;
- оператор перехода к следующей итерации цикла **`continue`**;
- оператор возврата из функции **`return`**;
- оператор безусловного перехода **`goto`**;
- оператор генерации исключения **`throw`**.