Инструкции C++

March 3, 2021

Инструкции C++

Инструкции задают порядок и условия выполнения программы.

Обозначения терминов и сокращений:

statement — инструкция (st)
operator — оператор (op)
expression — выражение (expr)
identifier — идентификатор (id).

Инструкция вычисления выражения

expr ;

Действие инструкции заключается в вычислении выражения. Результат последнего оператора игнорируется. Для того, что бы инструкция имела смысл, необходимо, что бы как минимум последний из операторов имел побочное действие.

```
rt = 2 * j++;
j++;
z <<= 4 ^ x;
++i;
;              // Пустая инструкция
f( 5 );
a + b * c; // Bad: нет побочных действий
```

Блочная инструкция (блок)

{ st ... }

Используется в тех случаях, когда по синтаксису требуется одна инструкция, а надо поместить несколько, например, в цикле или условной инструкции.

```
{
  int a = 5;
  DataBase db( "127.0.0.1", mydb );
  a = 7 + a;
} // здесь a и db будут уничтожены
```

Блок ограничивает область видимости и время жизни локальны объектов. **Локальные объекты** блока доступны непосредственно только в самом блоке и уничтожаются при выходе из него.

Инструкция объявления

type name; type name = expr;

type name {list}; type name (list);

Создаёт объект заданного типа, при необходимости инициализируя его заданным значением. C++11: auto, decltype()

```
double b;
char c5 = 'Z', *p;
Message tx( "me", "you",  5000 );
int q { 123 };
auto x = find( start, end, what );
decltype(c5) c7 = ++c5;
```

Единственная инструкция, которая допустима вне функции. В этом случае создаётся глобальный объект (специализируется до работы функции main, и уничтожается после её завершения).
extern type name; — описание без определения.

Инструкция цикла for

for(expr1+; expr2; expr3) st

Используется для организации повторяющихся вычислений. **st** — тело цикла, если более одной инструкции — использовать **{ }**.
expr1 — выражение инициализации, выполняется при входе в цикл. Можно создавать локальные переменные.
expr2 — условие продолжения, вычисляется перед каждой итерацией. Если false — выход из цикла.
expr3 вычисляется после тела цикла.

```
for( auto i=0x8000; i; i>>=1 ) {
  cout << "i=␣" << i << endl;
}
```

C++11: **for(type var : container) st**

C++20: **for(init; type var : container) st**

Инструкция цикла while

while(expr+) st

Цикл с предусловием:
st — тело цикла.
expr — условие продолжения цикла, проверяется **до** входа в тело цикла.

```
char c;
while( (c = get_next_char() ) != 'q' )
  cout << c;

while( state_is_good() ) {
  output_state();
  get_new_state();
}
```

Инструкция цикла do . . . while

do st while(expr);

Цикл с постусловием:
st — тело цикла.
expr — условие продолжения цикла, проверяется **после** тела цикла.

```
char c;
do {
   process_data();
   print_data();
} while( data_is_good() );
```

Инструкция выхода из цикла

break;

Приводит к безусловному завершению текущего цикла. Локальные переменные уничтожаются. Можно выйти только из одного цикла.

```
for( int i=0; i<N; ++i ) {
  if( a[i] < 0 ) {
    break;
  }
  cout << "a[" << i << "]= " << i << '\n';
}
```

Инструкция перехода к следующей итерации цикла

continue;

Приводит к завершению текущей итерации цикла и переходу к следующей. При этом выполняются выражения expr3 (для for) и проверка условия (для всех циклов).

```
for( char c='A'; c<'z'; ++c ) {
  if( ! isalpha(c) ) {
     continue;
  }
  cout << c << ' ';
}
```

Инструкции безусловного перехода и метки

goto id; id: st

Приводит к переходу на указанную метку.
Можно выйти сразу из нескольких вложенных циклов, но нельзя войти в цикл, блок, или перейти к другой функции.

```
int i, j; // outside for!
for( i=0; i<10; ++i ) {
  for( j=0; j<10; ++j ) {
    if( i*i+j*j==74 ) {
      goto found;
    }
  }
}
i = j = -1;
found: cout << i << ' ' << j << '\n';
```

Инструкция возврата из функции

return expr;

Приводит к выходу из функции и возврату значения. Тип выражения должен или совпадать с типом возвращаемого значения функции, или неявно приводится к нему.

```
int f( const char *s, char c )
{
  for( int i = 0; *s; ++s, ++i ) {
    if( *s == c ) {
      return i;
    }
  }
  return -1;
}
```

Более подробно инструкция рассмотрена в разделе, посвящённом функциям.
C++20: co_return

Условная инструкция

if(expr+) st1 | **if(expr+) st1 else st2**

Если выражение expr истинно, выполняется инструкция **st1**, в противном случае — **st2** (если есть). Если надо больше одной инструкции — используется блок.

```
if( a>0 && b > 0 )
  c = sqrt( a * b );
if( z ) {
  Q = 0; dq = 0.1;
}
if( bool z = df > 5 ) {
  f( 5, z ); dq = 0;
} else {
  f( 0, z ); dq = -0.2;
}
```

Инструкции выбора

switch(expr+) . . . **case c_i_expr . . .**

Позволяют выбрать один из множества перечислимых вариантов.

```
switch( char c=get_char() ) {
  case 'q': case 'Q': action = 0; break;
  case 'd': debug = 1;  [[fallthrough]]; // C++17
     /* nobreak */ // к следующей
  case 'v': action = 2; break;
  case 'x': return 0;
  default:  action = -1; break; // по-умолчанию
}
```

Здесь инструкция “break” позволяет выйти из блока за “switch”. Отсутствие “break” означает переход к следующей инструкции.

Инструкции перехвата и обработки исключений

try{ .... } catch(type name) { . . . }

Инструкции будут рассмотрены в соответствующем разделе

Комментарии

C++, как и современный C, поддерживает 2 вида комментариев.

/* comment */

// comment to end of line

Комментарии не могут быть вложенными. Каждый из видов комментариев комментирует другой.
Существуют специальные комментарии, позволяющие автоматизировать создание и поддержку документации к программам (doxygen, qdoc, javadoc).

```
/** @author XXX */
/*! \brief Brief description. */
/** @param a Parameter: left margin. */
//* some horrible function
```