Константы. Присваивание. Арифметические операции

1. Константы

Константа — это ограниченная последовательность символов алфавита языка, представляющая собой изображение фиксированного (неизменяемого) объекта

Способы представления констант в C++

1) с помощью директивы препроцессора ***#define***:

#define MAX 100

2) с использованием ключевого слова ***const***:

const тип ИмяПеременной = НачальноеЗначение;

const int n=10;

Виды констант (по типу представляемых данных)

1. Числовые
2. Символьные
3. строковые

Числовые константы

- числовые: целочисленные, вещественные
- числовые константы могут быть описаны с использованием суффиксов, которые определяют их типы. Эти суффиксы не являются обязательными, так как компилятор понимает из контекста, константу какого типа данных вы хотите использовать, однако Если программиста не устраивает тип, который компилятор приписывает константе, то тип можно явно указать в записи константы с помощью суффиксов
- суффиксы для целочисленных типов используются редко

Тип данных	Суффикс	Значение
int	u или U	unsigned int
int	l или L	long
int	ul, uL, Ul, UL, lu, lU, Lu или LU	unsigned long
int	ll или LL	long long
int	ull, uLL, Ull, ULL, llu, llU, LLu или LLU	unsigned long long
double	f или F	float
double	l или L	long double

Пример для целочисленного типа:

```
1 unsigned int nValue = 5u; // тип int unsigned
2 long nValue2 = 5L; // тип long
```

Пример для вещественного типа:

```
1 float fValue = 5.0f; // тип float
2 double d = 6.02e23; // тип double (по умолчанию)
```

Простые типы данных

Тип	Количество байтов	Диапазон значений
short	2	–32 768 ÷ 32 767
unsigned short	2	0 ÷ 65 535
int	4	–2 147 483 648 ÷ 2 147 483 647
unsigned int	4	0 ÷ 4 294 967 295
long	4	–2 147 483 648 ÷ 2 147 483 647
unsigned long	4	0 ÷ 4 294 967 295
long long	8	–9 223 372 036 854 775 808 ÷ 9 223 372 036 854 775 807
unsigned long long	8	0 ÷ 18 446 744 073 709 551 615
signed char	1	–128 ÷ 127
unsigned char	1	0 ÷ 255
bool	1	false или true
float	4	3.4E +/– 38 (7 знаков после запятой)
double	8	1.7E +/– 308 (15 знаков после запятой)

Числовые: целочисленные константы

Десятичные	Последовательность цифр (0 — 9), которая начинаются с цифры, отличной от нуля. Пример: 1, -29, 385. Исключение — число 0.
Восьмеричные	Последовательность цифр (0 — 7), которая всегда начинается с 0. Пример: 00, 071, -052, -03.
Шестнадцатеричные	Последовательность шестнадцатеричных цифр (0 — 9 и A — F), которой предшествует присутствует 0x или 0X. Пример: 0x0, 0x1, -0x2AF, 0x17.

Числовые: вещественные константы

Константа с фиксированной точкой

Константа с плавающей точкой (вещественная константа) всегда представляется числом с плавающей точкой двойной точности, т. е. как имеющая тип double, и состоит из следующих частей:

1. целой части — последовательности цифр;
2. точки — разделителя целой и дробной части;
3. дробной части — последовательности цифр;
4. символа экспоненты e или E;
5. значения экспоненты в виде целой константы (может быть со знаком)

1.2345E-20

Примеры вещественных констант

Любая часть (но не обе сразу) из нижеследующих пар может быть опущена: • целая или дробная часть; • точка или символ e (E) и экспонента в виде целой константы	**345.** **3.14159** **2.1E5** **.123E3** **4037e-5**

Символьные константы

- Символьная константа — это один символ, например: ‘z’.
- Символьные константы оформляются в программе как один или несколько символов, заключенных в апострофы (' ‘)
- Они могут состоять из одного символа, имеют тип char и занимают в памяти один байт
- Символьные константы, состоящие из двух символов, имеют тип int и занимают два байта
- В качестве символьных констант также могут использоваться управляющие коды, не имеющие графического представления. При этом код управляющего символа начинается с символа ‘\’ (обратный слеш)

Примеры записи символьных констант

символ	**'a'**	**'B'**	**'='**	**'+'**	**'F'**	**'['**	**'d'**
код	'\141'	'\102'	'\75'	'\53'	'\106'	'\133'	'\144'
код	'\x61'	'\x42'	'\x3D'	'\x2B'	'\x46'	'\x5B'	'\x64'

Управляющий символ «гудок» - '\a', '\07'
Управляющий символ «перевод на новую строку» - '\n', '\x0A'

Строковые константы

- Строковая константа — это последовательность символов (латинские буквы, русские буквы, цифры), заключенная в кавычки, например: «Это строковая константа»
- Внутри строк также могут использоваться управляющие последовательности: «\n Я делаю лабораторную работу №4 \«Константы. Операция присваивания\» »

Перечислимые константы

С помощью ключевого слова **enum** можно объявить особый целочисленный тип с набором именованных целых констант, называемых перечислимыми константами:

enum тег {СписокИменованныхКонстант};

Примеры:

enum day {sun, mon, tue, wen, thu, fri, sat}
enum flag {false, true}
enum { one = 1, two = 2, three = 3 }
enum number {a=54,b,c=60,d=c+5}

2. Присваивание

1. **Лексема** — это логически выделенная единица языка, воспринимаемая как единое целое компилятором и программистом.
2. **Операция** – лексема, определяющая специальный способ записи некоторых действий
3. **Операнд** – это лексема, к которой применена операция
4. **Оператор** – конструкция языка, посредством которой задается очередной шаг вычислительного процесса
5. **Разделители** (знаки пунктуации) предназначены для разделения ключевых слов и идентификаторов (имен)
6. **Выражение** – это последовательность операндов, разделителей и знаков операций, результатом которой является определенное значение

Разделители

Знак	Назначение
[]	ограничители-индексы одно- и многомерных массивов
()	группирование выражений и указание на вызовы функций и их параметры
{}	ограничители области составного оператора и блока
,	операция разделения элементов списков
;	терминатор оператора (пустой оператор)
:	разделитель метки и оператора
…	обозначение переменного числа параметров в функциях и их прототипах
*	знак умножения и доступа через указатель
=	знак операции присваивания и инициализации
#	признак директивы предпроцессора

Операции

- результатом выполнения операции является число
- операции могут быть бинарными или унарными
- виды операций: операции присваивания; арифметические; сдвиговые операции; операции отношения; логические

Операция присваивания

Операция присваивания обозначается символом “=“
и выполняется в 2 этапа:

1. Вычисляется выражение в правой части;
2. Результат присваивается операнду, стоящему в левой части:

объект = выражение

3. В случае если объекты в левой и правой части операции присваивания имеют разные типы используется операция явного приведения типа:

объект = (тип)выражение

Примеры операции присваивания

- int a = 4; *// переменной **a** присваивается значение 4*
  int b;
  b = a + 2; *// переменной **b** присваивается значение 6, вычисленное в правой части*

- float a = 241.5;
  *// перед вычислением остатка от деления **a** приводится к целому типу*
  int b = (int)a % 2; // b = 1

3. Арифметические операции

Основные **бинарные операции**, расположенные в порядке уменьшения приоритета:
***** — умножение
/ — деление
+ — сложение
— — вычитание
% — остаток от целочисленного деления

Основные **унарные операции**:
++ — инкрементирование (увеличение на 1)
-- — декрементирование (уменьшение на 1)
— — изменение знака

!!! Про некоторые унарные

Результат вычисления выражения, содержащего операции инкрементирования или декрементирования, зависит от того, где расположен знак операции (до объекта или после него). Если операция расположена до объекта, то сначала происходит изменение значения переменной на 1, а потом это значение используется для выполнения следующих операций. Если операция ++ или — расположена после переменной, то сначала выполняется операция, а потом значение переменной изменяется на 1.

Бинарные арифметические операции с операцией присваивания:

объект *= выражение; // объект = объект * выражение

объект /= выражение; // объект = объект / выражение

объект += выражение; // объект = объект + выражение

объект -= выражение; // объект = объект —

Таблица приоритетов

НАЙДИ!!!

```
1  int a = 10;
2  int b = 7;
3  int c = a + b;
4  int d = 4 + b;
```

```
1  int a = 10;
2  int b = 7;
3  int c = a - b;
4  int d = 41 - b;
```

```
1  int a = 10;
2  int b = 7;
3  int c = a * b;
4  int d = b * 5;
```

```
1  int a = 20;
2  int b = 5;
3  int c = a / b;
4  double d = 22.5 / 4.5;
```

```
1  double k = 10 / 4;     // 2
2  std::cout << k;
```

```
1  double k = 10.0 / 4;     // 2.5
2  std::cout << k;
```

```
1  int a = 33;
2  int b = 5;
3  int c = a % b;  // 3
4  int d = 22 % 4; // 2 (22 - 4*5 = 2)
```

```
1  int a = 8;
2  int b = ++a;
3  std::cout << a << "\n"; // 9
4  std::cout << b << "\n"; // 9
```

```
1  int a = 8;
2  int b = a++;
3  std::cout << a << "\n"; // 9
4  std::cout << b << "\n"; // 8
```

```
1  int a = 8;
2  int b = --a;
3  std::cout << a << "\n"; // 7
4  std::cout << b << "\n"; // 7
```

```
1  int a = 8;
2  int b = a--;
3  std::cout << a << "\n"; // 7
4  std::cout << b << "\n"; // 8
```

```
1  int a = 8;
2  int b = 7;
3  int c = a + 5 * ++b;      // 48
4  std::cout << c;
```

```
1  int a = 8;
2  int b = 7;
3  int c = (a + 5) * ++b;      // 104
4  std::cout << c;
```

int a=2;
int b=3;
int c;
c = a*++b;
	int a=2;
	int b=3;
	int d;
	d = a*b++;

```
#include <stdio.h>

void main()

{
    int x = 5;
  int y = x + 1;
  int z = x++;

  printf("x= %d\ny= %d\nz= %d", x, y, z);

  return 0;
}
```

```
#include <iostream>

int main()
{
  using namespace std;
  cout << 2*3 << '\t' << 2*+3 << '\t' << +2*+3 << '\t'
       << 2*-3 << '\t' << +2*-3 << '\t'
       << -2*3 << '\t' << -2*+3 << '\t' << -2*-3 << endl;
  cout << 3/2 << '\t' << 3/+2 << '\t' << +3/+2 << '\t'
       << 3/-2 << '\t' << +3/-2 << '\t'
       << -3/2 << '\t' << -3/+2 << '\t' << -3/-2 << endl;
  cout << 2%3 << '\t' << 2%+3 << '\t' << +2%+3 << '\t'
       << 2%-3 << '\t' << +2%-3 << '\t'
       << -2%3 << '\t' << -2%+3 << '\t' << -2%-3 << endl;
  cout << 3+2 << '\t' << 3+(+2) << '\t' << +3+(+2) << '\t'
       << 3+-2 << '\t' << +3+-2 << '\t'
       << -3+2 << '\t' << -3+(+2) << '\t' << -3+-2 << endl;
  cout << 3-2 << '\t' << 3-+2 << '\t' << +3-+2 << '\t'
       << 3-(-2) << '\t' << +3-(-2) << '\t'
       << -3-2 << '\t' << -3-+2 << '\t' << -3-(-2) << endl;
  return 0;
}
```

```cpp
#include <iostream>

int d = 0;

int main()
{
  using namespace std;
  int a, b, c, d(1);
  double e(0.5);
  cout << (d + e) << endl;
  cout << ::d << '\t' << d << endl;
  cout << d << '\t';
  c = d++;
  cout << c << '\t' << d << '\t';
  b = d--;
  cout << b << '\t' << d << endl;
  cout << sizeof ::d << '\t' << sizeof d << endl;
  cout << b << '\t';
  a = ++b;
  cout << a << '\t' << b << '\t';
  c = --b;
  cout << c << '\t' << b << endl;
  return 0;
}
```