Операции

$+, -, *, /, \%$

Особенности

1) только над **одинаковыми**

Когда int + float → конверсии (если один вещественный, друзой тоже в вещественный; еще сравнение по рангу и конверт в высший; при одном ранке unsigned выше; ранк меньше инта промоушите до int)

2) Операции с unsigned дает модулярную арифметику; иначе если по signed, то undifind behavier (ведет себя как угодно)

3) результат деления имеет тот же тип что и высший ранг делимое/делитель

4) Округление к нулю

5) деление на ноль → undefind behavier

6) деление min signed type /-1 → 128 > max signed type (undof behavier)

== ; != ; < ; <= ; >=

Особенности

1) приведение к одному типу

Пр.   uint u = 3;   i < u → False

     int i = -1,   i станет unsigned т.е. $2^{32} - 1$

2) результат сравнение → int (0 или 1), но в C++ bool

x && y ; x || y, !x

Особенности:

1) результат: 0 или 1

2) для && и || вып. неполное вычисление (иногда не идет до конца при ответе)

& ; | ; ^ ; ~ ; << ; >>

Пр.:

1) 3 && 6 → 1

   3 & 6 → 2

№3 Сдвиги:

1) Сдвиг влево

2) Логический сдвиг вправо

3) арифметический сдвиг направо

Пр.:

Uint u=3

u << $\#$ 32 ничего не изменит

№4 еще одна операция

Sizeof (int) выбирает размер в байтах

Size of (3+2.0) → 8

$a = b$

$a = b = c = 0,$

$if (a = b)$ ← вполне законно

$f (3, a = 4)$

$a \mathrel{+}= b$ ~ $a = a + b$

$a++$; $++a$ ← всегда +1

$a = 3$
$b = a++ = 3$
$c = ++a = 4$

```
if ( ... )              if ( ... )
                        {
    действие;               действие;
        иначе:
    действие2;              действие2;
                        }
```
← не дружат

(№3) в C_{++} законно

```
if ( int x = f() )
{
    y = x
} и в else тоже дуж доступна
```

```
while ( усл. )
{
    Команда;
}
```

```
do
{
    Команда;
}
while ( усл ).
```

Break — выход из цикла (внутреннего)

Continue —

goto метка (нужно ставить чтобы попасть)

```
while (усл)
  while ( )
    while ( )
    {
      if ( )
        goto метка (lend);
    }

метка (lend)
```

```
for ( A; B; C)        ~        A;
{
  D;        continue не          while (B)
}  '      пропускает C           {                  continue пропускает
                                   D;                C
                                   C;
                                 }
```

(N3)

```
for ( A; B; C);   ← уже конец цикла
```

int x,y;

scanf ("%i %i", &x, &y); ==2;

ждет , и читает y , иначе только x (пробел скипаются)

(N3) scanf возвращает кол-во прочтенных т.е. можно проверит

printf ("x+y = %i \n", x+y);

Работа с файлами:

```c
FILE *f = fopen("имя", "способ");
```
(type)

```c
способ = {"r", "w"}
```
читать / писать

⑬ для чтения бинарных модификатор b

Пр:

```c
FILE * fin = fopen("имя", "r");
FILE * fout = fopen("имя", "r");
fscanf(fin, "%i %i", &x, &y);   // читать из
fprintf(fout, "x+y = %i \n", x+y);   // писать в
fclose(fin);
fclose(fout);   // закрыть файл
if (fin != NULL)   ← проверка на открытие
fclose(NULL) – вы падаете
```

Полная форма main:

```c
int main(int argc, char **argv)
    my.exe 1 abc   (argc = 3)
{
    if (argc != 3) → error
        fopen(argv[1], "r")
        fopen(argv[2], "w")
}
```