Wyrażenia logiczne w języku C++

Algebra Boole'a to dział matematyki i logiki, który operuje wartościami prawda/fałsz (true/false) oraz operacjami logicznymi. W C++, algebra Boole'a jest używana do manipulacji wartościami logicznymi.

W języku C++, typ `bool` służy do reprezentowania wartości logicznych, które mogą być albo `true` (prawda), albo `false` (fałsz). Przykład ilustrujący jego działanie:

```
#include <iostream>

int main() {
    bool jestSlonecznie = true;
    bool padaDeszcz = false;

    if (jestSlonecznie) {
        std::cout << "Na zewnątrz jest słonecznie!" << std::endl;
    } else {
        std::cout << "Na zewnątrz nie jest słonecznie." << std::endl;
    }

    if (padaDeszcz) {
        std::cout << "Na zewnątrz pada deszcz." << std::endl;
    } else {
        std::cout << "Na zewnątrz nie pada deszcz." << std::endl;
    }

    return 0;
}
```

W tym przykładzie:

- `jestSlonecznie` jest przypisane wartość `true`.
- `padaDeszcz` jest przypisane wartość `false`.

Program sprawdza te wartości logiczne i drukuje odpowiednie komunikaty na konsoli. Możesz używać zmiennych logicznych w instrukcjach warunkowych, takich jak `if`, `while` i `for`, aby kontrolować przepływ swojego programu na podstawie warunków prawdziwych lub fałszywych.

Wartości logiczne (bool) w C++ są reprezentowane przez wartości numeryczne całkowite:

- `true` (prawda) jest reprezentowane przez 1.
- `false` (fałsz) jest reprezentowane przez 0.

Przykład, który pokazuje, jak te wartości mogą być używane:

```
#include <iostream>

int main() {
    bool prawda = true;
    bool falsz = false;
    std::cout << "Prawda: " << prawda << std::endl;  // Wydrukuje 1
    std::cout << "Fałsz: " << falsz << std::endl;    // Wydrukuje 0
    return 0;
}
```

W tym przykładzie zmienne `prawda` i `falsz` są wypisywane jako liczby 1 i 0.

W C++ każda liczba różna od zera jest traktowana jako `true`. Oznacza to, że zarówno liczby dodatnie, jak i ujemne, będą interpretowane jako `true`. Tylko liczba 0 jest traktowana jako `false`.

Przykład:

```
#include <iostream>

int main() {
    int liczba1 = 42;
    int liczba2 = -5;
    int liczba3 = 0;

    if (liczba1) {
        std::cout << "Liczba 1 jest prawdziwa." << std::endl; // Wydrukuje się
    } else {
        std::cout << "Liczba 1 jest fałszywa." << std::endl;
    }

    if (liczba2) {
        std::cout << "Liczba 2 jest prawdziwa." << std::endl; // Wydrukuje się
    } else {
        std::cout << "Liczba 2 jest fałszywa." << std::endl;
    }

    if (liczba3) {
        std::cout << "Liczba 3 jest prawdziwa." << std::endl;
    } else {
        std::cout << "Liczba 3 jest fałszywa." << std::endl; // Wydrukuje się
    }

    return 0;
}
```

W tym przykładzie:

- `liczba1` jest równa 42, więc zostanie potraktowana jako `true`.
- `liczba2` jest równa -5, więc również zostanie potraktowana jako `true`.
- `liczba3` jest równa 0, więc zostanie potraktowana jako `false`.

Operatory logiczne w C++ służą do manipulacji wartościami logicznymi (prawda/fałsz). Zestawienie podstawowych operatorów logicznych:

1. **NOT (`!`)**: Negacja - zmienia wartość logiczną na przeciwną.

   ```
   bool a = true;
   bool b = !a; // b jest false
   ```

2. **AND (`&&`)**: Koniunkcja - zwraca true, jeśli oba operandy są prawdziwe.

   ```
   bool a = true;
   bool b = false;
   bool c = a && b; // c jest false
   ```

3. **OR (`||`)**: Alternatywa - zwraca true, jeśli przynajmniej jeden z operandów jest prawdziwy.

   ```
   bool a = true;
   bool b = false;
   bool c = a || b; // c jest true
   ```

4. **XOR (^)**: Alternatywa wykluczająca - zwraca true, jeśli dokładnie jeden z operandów jest prawdziwy. Jest to operator bitowy, ale można go użyć także do wartości bool.

```
bool a = true;
bool b = false;
bool c = a ^ b;
```

Przykład użycia operatorów logicznych w kodzie:

```
#include <iostream>

int main() {
    bool a = true;
    bool b = false;

    std::cout << "a && b: " << (a && b) << std::endl;
    std::cout << "a || b: " << (a || b) << std::endl;
    std::cout << "!a: " << (!a) << std::endl;

    return 0;
}
```

Ten program wyświetli:

```
a && b: 0
a || b: 1
!a: 0
```

Tablice prawdy

Tablice pokazują, jak zachowują się operatory logiczne dla różnych kombinacji wartości logicznych.

Operator AND (&&)

A	B	A && B
true	true	true
true	false	false
false	true	false
false	false	false

Operator OR (||)

A	B	A || B
true	true	true
true	false	true
false	true	true
false	false	talse

Operator NOT (!)

A	!A
true	false
false	true

Operator XOR (^)

A	B	A B
true	true	false
true	false	true
false	true	true
false	false	false

Operatory C++ zwracające wartość logiczną

1. **Operator równości (==)**: Sprawdza, czy dwa operandy są równe.

   ```
   int a = 5, b = 5;
   bool result = (a == b); // result jest true
   ```

2. **Operator nierówności (!=)**: Sprawdza, czy dwa operandy są różne.

   ```
   int a = 5, b = 4;
   bool result = (a != b); // result jest true
   ```

3. **Operator większe niż (>)**: Sprawdza, czy lewy operand jest większy niż prawy operand.

   ```
   int a = 5, b = 4;
   bool result = (a > b); // result jest true
   ```

4. **Operator mniejsze niż (<)**: Sprawdza, czy lewy operand jest mniejszy niż prawy operand.

   ```
   int a = 4, b = 5;
   bool result = (a < b); // result jest true
   ```

5. **Operator większe lub równe (>=)**: Sprawdza, czy lewy operand jest większy lub równy prawemu operandowi.

   ```
   int a = 5, b = 5;
   bool result = (a >= b); // result jest true
   ```

6. **Operator mniejsze lub równe (<=)**: Sprawdza, czy lewy operand jest mniejszy lub równy prawemu operandowi.

   ```
   int a = 4, b = 5;
   bool result = (a <= b); // result jest true
   ```

7. **Operator logiczny NOT (!)**: Neguje wartość logiczną operandów.

   ```
   bool a = true;
   bool result = !a; // result jest false
   ```

8. **Operator logiczny AND (`&&`)**: Sprawdza, czy oba operandy są prawdziwe.

```
bool a = true, b = false;
bool result = (a && b); // result jest false
```

9. **Operator logiczny OR (`||`)**: Sprawdza, czy przynajmniej jeden z operandów jest prawdziwy.

```
bool a = true, b = false;
bool result = (a || b); // result jest true
```

10. **Operator logiczny XOR (`^`)**: Sprawdza, czy dokładnie jeden z operandów jest prawdziwy (częściej używany w operacjach bitowych).

```
bool a = true, b = false;
bool result = (a ^ b); // result jest true
```

Przykłady wyrażeń logicznych

- **AND (`&&`) oraz równość (`==`)**:

```
int a = 10;
int b = 20;
bool result = (a == 10) && (b > 15); // result jest true
```

- **OR (`||`) oraz nierówność (`!=`)**:

```
int a = 5;
int b = 10;
bool result = (a != 5) || (b == 10); // result jest true
```

- **NOT (`!`) oraz większe niż (`>`)**:

```
int a = 7;
int b = 5;
bool result = !(a < b); // result jest true
```

- **OR (`||`) oraz mniejsze niż (`<`)**:

```
int a = 3;
int b = 7;
bool result = (a < 5) || (b > 10); // result jest true
```

- **AND (`&&`) oraz mniejsze lub równe (`<=`)**:

```
int a = 8;
int b = 12;
bool result = (a <= 10) && (b >= 10); // result jest true
```

- **NOT (`!`) oraz OR (`||`)**:

```
bool a = true;
bool b = false;
bool result = !(a || b); // result jest false
```

- **AND (&&) oraz większe lub równe (>=)**:

```
int a = 5;
int b = 5;
bool result = (a >= 5) && (b <= 10); // result jest true
```

- **XOR (^) oraz nierówność (!=)**:

```
bool a = true;
bool b = false;
bool result = (a ^ b) != false; // result jest true
```

- **AND (&&) oraz mniejsze niż (<)**:

```
int a = 4;
int b = 9;
bool result = (a < 10) && (b > 5); // result jest true
```

- **OR (||) oraz równość (==)**:

```
int a = 6;
int b = 4;
bool result = (a == 6) || (b == 8); // result jest true
```

- **AND (&&), równość (==), OR (||)**:

```
int a = 10;
int b = 20;
int c = 30;
bool result = (a == 10) && (b > 15) || (c < 25); // result jest true
```

- **NOT (!), OR (||), mniejsze niż (<)**:

```
int a = 5;
int b = 10;
bool result = !(a < 3) || (b > 8) && (a == 5); // result jest true
```

- **AND (&&), większe niż (>), nierówność (!=)**:

```
int a = 7;
int b = 14;
int c = 21;
bool result = (a > 5) && (b != 10) && (c < 25); // result jest true
```

- **AND (&&), OR (||), większe lub równe (>=)**:

```
int a = 8;
int b = 12;
int c = 5;
bool result = (a >= 8) && (b < 15) || (c != 5); // result jest true
```

- **OR (||), NOT (!), większe lub równe (>=)**:

```
int a = 3;
int b = 7;
bool result = (a < 5) || !(b >= 10) && (a == 3); // result jest true
```

- **XOR (^), OR (||), większe lub równe (>=)**:

```
bool a = true;
bool b = false;
bool result = (a ^ b) || (a && b) && (b != false); // result jest
true
```