Основы C++

1 курс ИС, БИ

Информатики 1 семестр,

семинар 3

Проверочная работа 2.

Вопросы:

1. **Что такое программа?**
2. **Что такое структурное программирование?**
3. **Что такое директива препроцессора?**
4. **Какие символы входят в состав алфавита С++?**
5. **Что такое идентификатор?**
6. **Перечислите основные правила создания идентификаторов.**
7. **Что такое венгерская нотация?**
8. **Применение знака нижнего подчеркивания в идентификаторах.**
9. **Что такое ключевые слова?**
10. **Что такое знаки операций?**
11. **Что вы знаете о целых константах?**
12. **Что вы знаете о вещественных константах?**
13. **Что вы знаете о символьных константах?**
14. **Что вы знаете о строковых константах?**
15. **Что такое управляющие последовательности? Приведите пример.**
16. **Что определяет тип данных? Перечислите основные типы данных.**
17. **Типы данных: целый тип - охарактеризуйте, приведите примеры.**
18. **Типы данных: символьный тип - охарактеризуйте, приведите примеры.**
19. **Типы данных: логический тип - охарактеризуйте, приведите примеры.**
20. **Типы данных: с плавающей точкой - охарактеризуйте, приведите примеры.**
21. **Типы данных: тип void.Особенности его использования?**

На листе с ответом обязательно указать:

Номер работы: 2

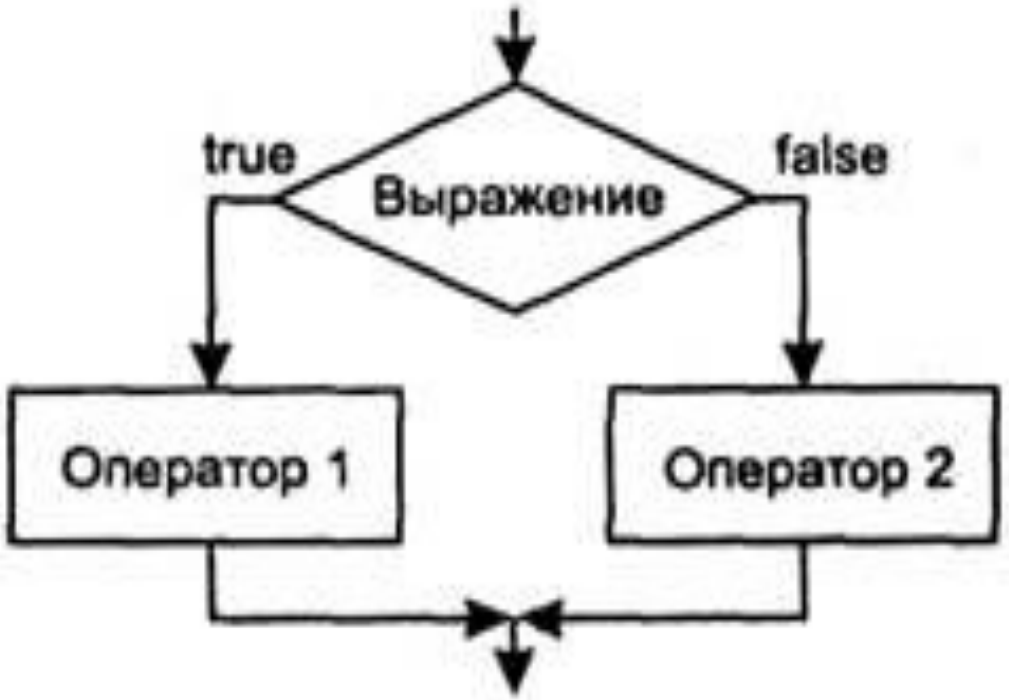
Дата выполнения:

Группа:

Фамилия:

Тема занятия: Базовые конструкции языка C++: ветвление.

Семинар 3



Полная конструкция

Неполная

Ветвление задает выполнение либо одного, либо другого оператора в зависимости от выполнения какого-либо условия.

Логические выражения С++

Операция в С++	Условие	Смысл записанных условий в С++
==	a == b	a равно b
!=	a != b	a не равно b
>	a > b	a больше b
<	a < b	a меньше b
>=	a >= b	a больше или равно b
<=	a <= b	a меньше или равно b

Пробелов между знаками ==; !=; >=; <=; быть не должно.

Также необходимо помнить правильную последовательность записи символов операций равенства и отношений.

Пример:

=! не правильно; != правильная запись;

=> не правильно; >= правильная запись;

=< не правильно; <= правильная запись.

Не путать операцию = (операция присваивания) и

операцию == (операция равенства)

так как это приводит к логическим ошибкам, т. е. программа даже скомпилируется без всяких ошибок, но работать будет не правильно.

Реализация конструкции ветвления

1. **Условная операция (?:)**

   операнд_1 ? операнд_2 : операнд_3

```
max = (b > a)? b : a;  //если (b > a), то max = b, иначе  max = a;


a > b ? cout << a : cout << b; // если a > b, то выполняется cout << a, иначе
      выполняется cout << b, То есть, программа печатает большее из
      чисел.

cout << "y = " << (x < 0 ? x : (x >= 0) && (x < 30) ? 0 : x * x  ) << endl;
//находящий значение y, если y=x, при x<0; y=0, при 0<=x<30; y=x2, при
      x>=30;

i = (i < n) ? i + 1: 1; //  i увеличивается на 1, если его значение не
      превышает n, а иначе принимала значение 1
```

Реализация конструкции ветвления

2. Условный оператор IF используется для разветвления процесса вычислений на два направления, позволяет определить действие, когда условие истинно и альтернативное действие, когда условие ложно.

if (выражение) оператор_1; [else оператор_2;]

Синтаксис записи оператора выбора:

```
if (/*проверяемое условие*/)
{
  /*тело оператора выбора 1*/;
} else
   {
     /*тело оператора выбора 2*/;
  }
```

Читается так: «Если проверяемое условие истинно, то выполняется **тело оператора выбора 1**, иначе (то есть проверяемое условие ложно) выполняется **тело оператора выбора 2**«. Обратите внимание на то, как записан оператор if else. Слово else специально сдвинуто вправо для того чтобы программный код был понятен и его было удобно читать.

Примеры:

1. if (a<0) b = 1; // отсутствует ветвь else, конструкция называется «пропуск оператора»

2. if (a<b && (a>d || a==0)) b++; else {b *= a; a = 0;} // проверка нескольких условий, их объединяют знаками логических операций.

```
3. if (a<b) {
if (a<c) m = a;
else m = c;
}
else {
if (b<c) m = b;
else m = c;
}
```

// Фигурные скобки в данном случае не обязательны, так как компилятор относит часть else к ближайшему if.

4. if (a++) b++; // в качестве выражений в операторе if чаще всего используются операции отношения, это не обязательно.

5. if (b>a) max = b; else max = a; // проще и нагляднее записывать в виде условной операции (в данном случае: **max = (b > a) ? b : a;**).

Пример: «Даны два числа, необходимо их сравнить».

```
...
   int a, b;
   cout << "Vvedite pervoe chislo: ";
   cin >> a;
   cout << "Vvedite vtoroe chislo: ";
   cin >> b;
   if ( a >= b) // если а больше либо равно b, то
   {
    cout << a << " >= " << b << endl;
   } else  // иначе
       {
        cout << a << " <= " << b << endl;
       }
```

Пример: найти значение y

$$y = \begin{cases} x, & \text{при } x<0; \\ 0, & \text{при } 0<=x<30; \\ x^2, & \text{при } x>=30; \end{cases}$$

```
...
   int x, y;
   cout << "Vvedite x: ";
   cin >> x;
   if (x < 0)
   {
    y = x; // выполняется, если x меньше нуля
   } else
         {
            if ( (x >= 0) && (x < 30) )
            {
             y = 0; // выполняется, если x больше либо равно нуля и меньше 30
             } else
                 {
                    if (x >= 30)
                  {
                    y = x * x; // выполняется, если x больше либо равен 30
                  }
                 }
    }
    cout << "y=" << y << endl;
```

Распространенные ошибки

- использование в выражениях вместо проверки на равенство (==) простого присваивания (=),

if(a=1)b=0 //присваивание переменной b будет выполнено независимо от значения переменной a

if (1==a)b=0; //рекомендуется в выражениях проверки переменной на равенство константе константу записывать слева от операции сравнения:

- неверная запись проверки на принадлежность диапазону. Например, чтобы проверить условие **0<x<1**, нельзя записать его в условном операторе непосредственно, так как будет выполнено сначала сравнение 0<x. а его результат (true или false, преобразованное в int) будет сравниваться с 1. Правильный способ записи: **if(0<x && x<1)**…

Основные операции языка C++

В соответствии с количеством операндов, которые используются в операциях, они делятся на унарные (один операнд), бинарные (два операнда) и тернарную (три операнда).

операция	Краткое описание
Унарные операции	
+ +	увеличение на 1
- -	уменьшение на1
sizeof	размер
~	поразрядное отрицание
!	логическое отрицание
-	арифметическое отрицание (унарный минус)
+	унарный плюс
&	взятие адреса
*	разадресация
new	выделение памяти
delete	освобождение памяти
(type)	преобразование типа

операция	Краткое описание
Бинарные и тернарная операции	
*	умножение
/	деление
%	остаток от деления
+	сложение
-	вычитание
<<	сдвиг влево
>>	сдвиг вправо
<	меньше
<=	меньше или равно
>	больше
>=	больше или равно
= =	равно //можно спутать с =
!=	не равно
&	поразрядная конъюнкция (И)
^	поразрядное исключающее ИЛИ
|	поразрядная дизъюнкция (ИЛИ)
&&	логическое И
||	логическое ИЛИ
? :	условная операция (тернарная)
=	присваивание
*=	умножение с присваиванием
/=	деление с присваиванием
%=	остаток отделения с присваиванием
+=	сложение с присваиванием
-=	вычитание с присваиванием
<<=	сдвиг влево с присваиванием
>>=	сдвиг вправо с присваиванием
&=	поразрядное И с присваиванием
|=	поразрядное ИЛИ с присваиванием
^=	поразрядное исключающее ИЛИ с присваиванием
,	последовательное вычисление

Операции сравнения и логические операции

Символ операции	Значение	Использование
!	Логическое НЕ	!expr
<	меньше	expr<expr
<=	Меньше либо равно	expr<=expr
>	больше	expr>expr
>=	больше либо равно	expr>=expr
==	равно	expr==expr
!=	не равно	expr!=expr
&&	логическое И	expr&&expr
||	логическое ИЛИ	expr||expr

Все операции в результате дают значение типа bool

И (&&) — логическая *конъюнкция* (умножение)

```
...
int main()
{
int one, two;
cin>>one>>two;
if ((one==1) && (two==2))
cout<<"true";
else cout<<"false";
system("pause");
return 0;
}
```

Возвращает истину только в том случае, когда истинны два простых условия, находящиеся по бокам от нее. Например:

Эта программа выводит *true* только в том случае, если оба простых условия верны. Первое условие оказалось ложным! — второе не проверяется.

Или (||) - логическая *дизъюнкция* (сложение).

```
…
int main()
{
int one, two;
cin>>one>>two;
if ((one==1) || (two==2))
cout<<"true";
else cout<<"false";
system("pause");
return 0;
}
```

Внешнее условие будет истинно в том случае, когда хотя бы одно из внутренних условий верно.

Для возвращения *true* достаточно выполнения одного из условий.

не (!) — логическая *инверсия* (отрицание).

```
…
int main()
{
int one;
cin>>one;
if (one!=1)
cout<<one<<" - не единица ";
else cout<<one<<" - единица";
system("pause");
return 0;
}
```

Возвращает истину тогда, когда выполняется условие с частицей *не*.

Условие в программе, выполняется в том случае, если переменная one не равна 1.

Выполним задания

Что будет выведено на экран при x=4, n=5, w=6, z=0.1 ???

…

```
((x+n)<0 || sin(z)<0) ? cout<<"Error"
:((x>w) ? cout<<2*z : cout<<3*z);
```

Что будет на экране в результате выполнения следующей программы??

```
…
void main()
{
int x;
cout << "Enter x: ";
cin >> x;
cout<<"y= "<<(x<0?x:(x>=0)&&(x < 30)?0:x*x)<<endl;
system("pause");
}
```

Домашнее задание

- 1. Павловская Т.А. С/С++. Программирование на языке высокого уровня – Спб.: Питер, 2004. – 461с.: ил. ISBN 5-94723-568-4 – разделы:
- **основные операции**
- **Базовые конструкции структурного программирования: Условный оператор if**

.

2. Напишите алгоритм и программу по нахождению корней квадратного уравнения.

Домашнее задание

1. Выучить

-Павловская Т.А. С/С++. Программирование на языке высокого уровня – Спб.: Питер, 2004. – 461с.: ил. ISBN 5-94723-568-4 – раздел

- **Базовые конструкции структурного программирования: Условный оператор if**

Вопросы:

1. Операции увеличения на 1 (+ +).
2. Операции уменьшения на 1 (- -).
3. Операция определения размера sizeof.
4. Операции отрицания (-, ! и ~).
5. Операция деление (/).
6. Операция остаток от отделения (%).
7. Операции сдвига (<< и >>) .
8. Операции отношения (<, <=, >, >=, ==, !=).
9. Поразрядные операции (&, |, ^).
10. Логические операции (&& и ||).
11. Операции присваивания (=, +=, -=, *= и т. д.).
12. Условная операция (?:).
13. Виды преобразования типов данных. Примеры.

2. Напишите алгоритм и программу для вычисления площади прямоугольника по его стороне и его периметру.