ОПЕРАТОРЫ ЯЗЫКА СИ++

СИ++

Типы данных в Си++

В Си++ определено 6 простых типов данных:
int (целый)
char (символьный)
wchar_t (расширенный символьный)

целочисленные

(число=мантисса x $10^к$)
bool (логический)
float(вещественный)
double (вещественный с двойной точностью)

с плавающей точкой
(число=мантисса x $10^к$)

Существует 4 спецификатора типа, уточняющих внутреннее представление и диапазон стандартных типов
short (короткий)
long (длинный)
signed (знаковый)
unsigned (беззнаковый)

Знаки операций в Си++

СИ++

&	*получение адреса операнда*
*	Обращение по адресу (разыменование)
-	унарный минус, меняет знак арифметического операнда
~	поразрядное инвертирование внутреннего двоичного кода целочисленного операнда (побитовое отрицание)
!	логическое отрицание (НЕ). В качестве логических значений используется 0 - ложь и не 0 - истина, отрицанием 0 будет 1, отрицанием любого ненулевого числа будет 0.

++	Увеличение на единицу:
	префиксная операция - увеличивает операнд до его использования,
	постфиксная операция увеличивает операнд после его использования.
	int m=1,n=2;
	int a=(m++)+n; // a=4,m=2,n=2
	int b=m+(++n);//a=3,m=1,n=3

- -	уменьшение на единицу: префиксная операция - уменьшает операнд до его использования, постфиксная операция уменьшает операнд после его использования.

sizeof	вычисление размера (в байтах) для объекта того типа, который имеет операнд
	имеет две формы
	sizeof выражение
	sizeof (тип)
	Примеры:
	sizeof(float)//4
	sizeof(1.0)//8, т. к. вещественные константы по умолчанию имеют тип double

СИ++

Операции в СИ++

Операции присваивания

*=, +=, -=, *= и т.д.*

Формат операции простого присваивания:

операнд1=операнд2

Условная операция.

Выражение1 ? Выражение2 : Выражение3;

Первым вычисляется значение выражения1. Если оно истинно, то вычисляется значение выражения2, которое становится результатом. Если при вычислении выражения1 получится 0, то в качестве результата берется значение выражения3.

Например:

x<0 ? -x : x ; //вычисляется абсолютное значение x.

СИ++

Ввод и вывод данных

В языке Си++ нет встроенных средств ввода и вывода – он осуществляется с помощью функций, типов и объектов, которые находятся в стандартных библиотеках. Существует два основных способа: функции унаследованные из Си и объекты Си++.

Для ввода/вывода данных в стиле Си используются функции, которые описываются в библиотечном файле stdio.h.

1) printf (форматная строка, список аргументов);

printf (“Значение числа Пи равно %f\n”, pi);

Форматная строка может содержать

1) ***символы печатаемые текстуально;***
2) ***спецификации преобразования;***
3) ***управляющие символы.***

Каждому аргументу соответствует своя спецификация преобразования:

***%d, %i -** десятичное целое число;*

%f -** число с плавающей точкой**;

***%e,%E –** число с плавающей точкой в экспоненциальной форме;*

***%u –** десятичное число в беззнаковой форме;*

***%c -** символ;*

***%s -** строка.*

В форматную строку также могут входить управляющие символы:

***\n** - управляющий символ новая строка;*

***\t** – табуляция;*

***\a** – звуковой сигнал и др.*

Модификаторы формата.

%[-]m[.p]C, где

- - задает выравнивание по левому краю,

m – минимальная ширина поля,

p – количество цифр после запятой для чисел с плавающей точкой и минимальное количество выводимых цифр для целых чисел (если цифр в числе меньше, чем значение p, то выводятся начальные нули),

C- спецификация формата вывода.

Пример

printf("\nСпецификации формата:\n%10.5d - целое,\n%10.5f - с плавающей точкой

\n%10.5e – в экспоненциальной форме\n%10s - строка",10,10.0,10.0,"10");

Будет выведено:

Спецификации формата:

00010 – целое

10.00000 – с плавающей точкой

1.00000e+001 - в экспоненциальной форме

10 – строка.

2) scanf (форматная строка, список аргументов);

В качестве аргументов используются адреса переменных. Например:

scanf(“ %d%f ”, &x,&y);

При использовании библиотеки классов Си++, Используется библиотечный файл iostream.h, в котором определены стандартные потоки ввода данных от клавиатуры cin и вывода данных на экран дисплея cout, а также соответствующие операции

1) << - операция записи данных в поток;

2) >> - операция чтения данных из потока.

Например:

```
#include <iostream.h>;
. . . . . .
cout << “\nВведите количество элементов: ”;
    cin >> n;
```

Оператор «выражение»

Примеры:

```
i++;

a+=2;

x=a+b;
```

Составные операторы

Это последовательность операторов, заключенная в фигурные скобки. Блок отличается от составного оператора наличием определений в теле блока.

```
{
n++;
summa+=n;
}
```

это составной оператор

```
{
int n=0;
n++;
summa+=n;
}
```

это блок

СИ++

Операторы выбора

Операторы выбора - это ***условный оператор*** и ***переключатель***.

if (выраж-условие **) оператор;** //сокращенная форма

if (выраж-условие **) оператор1;else оператор2;** //полная форма

```
if (d>=0)
{
x1=(-b-sqrt(d))/(2*a);
x2=(-b+sqrt(d))/(2*a);
cout<< "\nx1="<<x1<<"x2="<<x2;
}
else cout<<"\nРешения нет";
```

Переключатель определяет множественный выбор.

switch (выражение**)**
{
case ***константа1*** **:** ***оператор1*** **;**
case ***константа2*** **:** ***оператор2*** **;**
. . . . . .
[default: ***операторы*****;]**
}

Пример:

```
#include <iostream.h>
void main()
{
    int i;
    cout<<"\nEnter the number";
    cin>>i;
    switch(i)
    {
    case 1:cout<<"\nthe number is one";
    case 2:cout<<"\n2*2="<<i*i;
    case 3: cout<<"\n3*3="<<i*i;break;
    case 4: cout<<"\n"<<i<<" is very beautiful!";
    default:cout<<"\nThe end of work";
    }
}
```

СИ++

Операторы циклов

1. Цикл с предусловием:

while (выражение-условие**) оператор;**

Пример

```
while (a!=0)
{
cin>>a;
s+=a;
}
```

СИ++

2. *Цикл с постусловием:*

do

оператор

while (выражение-условие**);**

Тело цикла выполняется до тех пор, пока выражение-условие истинно.

Пример:

```
do
{
cin>>a;
s+=a;
}
while(a!=0);
```

3. *Цикл с параметром:*

for (*выражение_1;выражение-условие;выражение_3***)**

оператор;

СИ++

Примеры использования цикла с параметром.

1) Уменьшение параметра:

```
for ( n=10; n>0; n--)
{ оператор};
```

2) Изменение шага корректировки:

```
for ( n=2; n>60; n+=13)
{ оператор };
```

3) Возможность проверять условие отличное от условия, которое налагается на число итераций:

```
for ( num=1;num*num*num<216; num++)
{ оператор };
```

4) Коррекция может осуществляться не только с помощью сложения или вычитания:

```
for ( d=100.0; d<150.0;d*=1.1)
{ <тело цикла>};
for (x=1;y<=75;y=5*(x++)+10)
{ оператор };
```

5) Можно использовать несколько инициализирующих или корректирующих выражений:

```
for (x=1, y=0; x<10;x++;y+=x);
```

Операторы перехода

СИ++

Операторы перехода выполняют безусловную передачу управления.

1) break - оператор прерывания цикла.

{

< операторы>

if (<выражение_условие>**) break;**

<операторы>

}

Пример:

// ищет сумму чисел вводимых с клавиатуры до тех пор, пока не будет введено 100 чисел или 0

```
for(s=0, i=1; i<100;i++)
{
cin>>x;
if( x==0) break; // если ввели 0, то суммирование заканчивается
s+=x;
}
```

2) continue - переход к следующей итерации цикла. Он используется, когда тело цикла содержит ветвления.

Пример:

```
//ищет количество и сумму положительных чисел
for( k=0,s=0,x=1;x!=0;)
{
cin>>x;
if (x<=0) continue;
k++;s+=x;
}
```

2) Оператор goto

Оператор goto имеет формат**:**

goto метка**;**

3) Оператор **return** – оператор возврата из функции. Он всегда завершает выполнение функции и передает управление в точку ее вызова. Вид оператора:

return [выражение];

Массивы

int a[100]; //массив из 100 элементов целого типа

Операция **sizeof(a)** даст результат 400, т. е.100 элементов по 4 байта. Элементы массива всегда **нумеруются с 0**.

a[0] – индекс задается как константа,

a[55] – индекс задается как константа,

a[I] – индекс задается как переменная,

a[2*I] – индекс задается как выражение.

Элементы массива можно задавать при его определении:

int a[10]={1,2,3,4,5,6,7,8,9,10} ;

Операция *sizeof(a)* даст результат 40, т. е.10 элементов по 4 байта.

int a[10]={1,2,3,4,5};

Операция sizeof(a) даст результат 40, т. е.10 элементов по 4 байта. Если количество начальных значений меньше, чем объявленная длина массива, то начальные значения элементы массива получат только первые элементы.

int a[]={1,2,3,4,5};

Операция sizeof(a) даст результат 20, т. е.5 элементов по 4 байта. Длин массива вычисляется компилятором по количеству значений, перечисленных при инициализации.

СИ++

Использование датчика случайных чисел для формирования массива.

В Си++ есть функция

int rand() – возвращает псевдослучайное число из диапазона 0..RAND_MAX=32767, описание функции находится в файле **<stdlib.h>.**

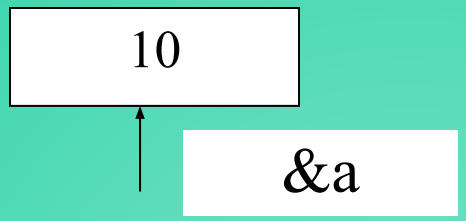
Пример формирования и печати массива с помощью ДСЧ:

```
#include<iostream.h>
#include<stdlib.h>
void main()
{
int a[100];
int n;
cout<<"\nEnter the size of array:";cin>>n;
for(int I=0;I<n;I++)
{a[I]=rand()%100-50;
cout<<a[I]<<" ";
}
}
```

Задача 1Найти максимальный элемент массива.

```
#include<iostream.h>
#include<stdlib.h>
void main()
{
int a[100];
int n;
cout<<"\nEnter the size of array:";cin>>n;
for(int I=0;I<n;I++)
{a[I]=rand()%100-50;
cout<<a[I]<<" ";
}
int max=a[0];
for(I=1;I<n;I++)
if (a[I]>max)max=a[I];
cout<<"\nMax="<<max";
}
```

Указатели



Программист может определить собственные переменные для хранения адресов областей памяти. Такие переменные называются указателями.

Указатель не является самостоятельным типом, он всегда связан с каким-то другим типом.

Указатели делятся на две категории: *указатели на объекты* и *указатели на функции*.

Указатели на объекты хранят адрес области памяти, содержащей данные определенного типа .

В простейшем случае объявление указателя имеет вид:
тип *имя;
Тип может быть любым, кроме ссылки.

Примеры:

int *i;

double *f, *ff;

char *c;

Можно определить указатель на указатель: ***int**a;***

Указатель может быть константой или переменной, а также указывать на константу или переменную.

Примеры:

1. **int i;** //целая переменная
const int ci=1; //целая константа
int *pi; //указатель на целую переменную
const int *pci; //указатель на целую константу

Указатель можно сразу проинициализировать:

int *pi=&i; //указатель на целую переменную
const int *pci=&ci; //указатель на целую константу

2.**int*const cpi=&i;** //указатель-константа на целую переменную
const int* const cpc=&ci; //указатель-константа на целую константу

Если модификатор const относится к указателю (т. е. находится между именем указателя и *), то он запрещает изменение указателя, а если он находится слева от типа (т. е. слева от *), то он запрещает изменение значения, на которое указывает указатель.

Присваивание адреса существующего объекта:

1) с помощью операции получения адреса

int a=5;

int *p=&a; или int p(&a);

2) с помощью проинициализированного указателя

int *r=p;

СИ++

Динамические переменные

В языке Си определены библиотечные функции для работы с динамической памятью, они находятся в библиотеке **<stdlib.h>:**

Для создания динамических переменных используют операцию new, определенную в СИ++:
указатель = new имя_типа[инициализатор];
где инициализатор – выражение в круглых скобках.

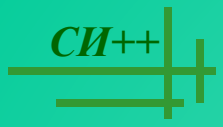
Например:

```
int*x=new int(5);
```

Для удаления динамических переменных используется операция delete, определенная в СИ++:
delete указатель;

Например:

```
delete x;
```


СИ++