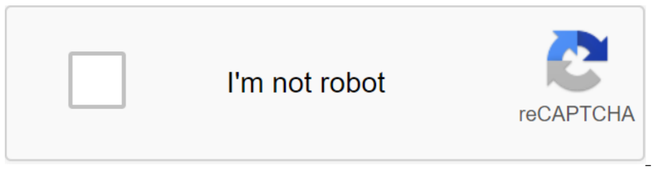
C



Continue

Explore everything about Constants in C E along with his Types.In this Easy C ' Training Courses, we discussed the variables and variables in C' in our previous tutorial. We learned that the value assigned to the variable can be changed throughout the program. Sometimes, depending on our requirements, we need some values that cannot be changed or changed in the program. However, we cannot guarantee that if these values are assigned to variables, no one will change the values in these variables. This is because the characteristics of the variable itself do not allow the values to be permanent. ReviewTho-like situations, we need one entity to which we can assign a value that will remain the same. Even if there is an attempt to change this value, the compiler will generate an error. This entity is called permanent/literal. They are also called Symbolic Constants as we have a certain name for these constants. In contrast, the constant values assigned to the variables are called literal constants. Constants can be any type of data. Constants in the NHS are treated in the same way as variables, except that their values do not change. Types of Data ConstantV C, Constants can be any type of data. All of them are named constants, i.e. each of these constants has its own name. The below types of constants in C':#1) Integer ConstantsThese are constants consisting of whole numbers without a decimal point. We may also have some suffixes associated with it depending on whether the number is signed or unsigned or long etc. In this case, we specify the prefix to the permanent: 0 for octal, 0x for hexadecimal, etc. Prefix for decimal constant is not specified. Below are some examples of a valid permanent integrator in C:0512 /octal0xFF /hexadecimal36 /decimal50L /long24U /unsignedPlease note, that we can't replicate a set-top box or suffix like a 50UU, as it will make permanent invalid.#2) The floating dots of ConstantFloating-point literals are literal with a decimal point. These constants can be represented in a decimal form or exponential form. When we use a decimal note, it must contain a decimal point, an exhibitor, or both. An exponential view should include an integration part, a faction, or both. We must submit a signed exhibitor e or E.Some examples of valid Literals Floating Points are:3.1423142E -5L1.143'3) The LiteralsThese character type symbol and are usually enclosed in a single quote ('). Characters that begin with the letter 'L' are widely characterized and stored in wchar_t character) type. Sort of. буквы символов хранятся в типе данных символов. Широкие буквы символов используются в основном в программировании GUI, таких как MFC или другое продвинутое программирование, включая STL. Некоторые примеры символов Literals являются:'xyz'L'M'The выше примеры символа Literals являются простой характер. Есть также символ буквальности известный как побег последовательности, которые дают особое значение для нескольких символов. Они используются для представления таких действий, как новые символы, вкладки и т.д. В приведенной ниже таблице перечислены последовательности побега, используемые в C3. Эти последовательности побега в основном используются при форматировании в C3 и могут быть использованы в качестве комбинации одной или нескольких последовательностей побега. После того, как программа C3 показывает использование некоторых из этих последовательностей побега.#include <iostream>#include с <string>помощью пространства имен std; int main () - Cout Смотрите здесь, чтобы исследовать <C++ program= to= demonstrate= escape= sequences;=></C++> <Hello\tthere\t\'STH\'; }= output:c++= program= to= demonstrate= escape= sequenceshello= there= ‘sth’as= the= above= code= shows,= we= can= use= these= escape= sequences= as= a= combination= as= well= to= format= the= output.#4)= string= literalunlike= character= literals,= string= literals= are= enclosed= in= double-quotes= (“= “).= string= literals= can= also= contain= simple= characters,= escape= sequences= or= other= universal= characters.following= are= some= of= the= valid= string= literals.“ hello,= world”= “hello,= world”= “hello”= “,”= “world”= get= to= know= the= ways= to= declare= and= use= references= in= c++.reference= is= a= major= concept= in= c++= programming= language.= although= it’s= not= as= strong= as= pointers,= nonetheless= it= allows= us= to= use= it= to= write= efficient= programs.= the= major= use= of= the= reference= variable= is= in= passing= parameters= to= functions.the= popular= ‘pass= by= reference’= parameter= passing= technique= makes= use= of= references.= in= this= tutorial,= we= will= see= what= a= reference= is,= and= how= to= declare= &= use= it.= we= will= also= discuss= the= differences= between= pointers= and= references= as= well= as= passing= and= returning= a= reference= to/from= functions.=> полный список учебников C. Что такое ссылка? Ссылка — это псевдоним или другое название переменной. Учитывая переменную с идентификатором, мы можем предоставить другой идентификатор этой переменной, чтобы мы могли обратиться к этой переменной либо с ее оригинальным именем, либо с другим именем. Это другое имя является то, что называется Reference.Consider y нас есть переменная i со значением 17. Если j является ссылкой, то представление памяти переменной i и reference j показано ниже. Как показано на рисунке выше, переменная и это псевдоним, т.е. точка and the same value. The VariablesA link announcement can be announced using the 'q' statement. The countdown declaration is displayed at below.int 10 euros; Refvar a; So in the above code, we announced a variable with a value of 10. Then we announce another variable refvar and assign it a variable. Note: qlt; /Hello'tthere't'STH'; объявляя refvar, мы использовали оператора как раз перед переменным именем. Это означает, что refvar является отсылкой к уже существующей переменной. Мы можем сослаться на переменную 'a' либо с помощью переменного имени a, либо используя имя ссылки 'refvar'. Ниже приведен простой пример ссылок: #include <iostream>#include <string>с помощью std пространства имен; int main () - int age - 22; int' numYears - возраст; двойная зарплата - 10000,00 евро; двойная заработная плата - зарплата; cout #include использование namespace std; недействительный свол <Age:></Age:> <> <> <endl;></endl;> <NumYears:></NumYears:> <> <> <endl;></endl;> <Salary:></Salary:> <> <> <endl;></endl;> <Wages:></Wages:> <> <> <endl; return= 0;= }= output:age:= 22= numyears:= 22= salary:= 10000= wages:= 10000in= the= above= code,= we= have= an= integer= variable= age.= next,= we= declare= a= reference= integer= variable= numyears= to= the= age= variable.= we= have= another= variable= salary= of= type= double.= next,= we= declare= a= double= reference= variable= wages= to= the= variable= salary.next,= we= print= the= variables,= first= age= then= its= reference= is= followed= by= salary= and= its= reference.= if= we= check= the= output= of= the= program,= we= understand= that= variable= and= reference= to= it= points= to= the= same= value= and= hence= age= and= numyears,= as= well= as= salary= and= wages,= have= the= same= values.reference= vs= pointerswhen= compared= to= pointers,= references= are= safer= and= easier= to= use.we= will= discuss= a= few= differences= between= pointers= and= references :unlike= pointers,= references= cannot= have= a= null= value.= it= is= mandatory= for= references= to= have= a= value= assigned= to= it.references= are= initialized= the= moment= they= are= created.= unlike= this,= pointers= can= be= initialized= at= any= point= of= time= and= not= necessarily= during= declaration.pointers= can= be= reassigned= to= the= values= at= ease.= but= with= references,= we= cannot= do= this.= once= a= value= of= the= variable= is= assigned,= i.e.= once= an= alias= of= a= variable= is= created,= we= cannot= assign= another= variable= to= this= reference.we= do= not= have= void= references.= by= definition,= a= reference= is= an= alias= for= the= variable= and= it= is= initialized= during= the= creation= itself.= thus,= there= is= no= chance= that= we= will= have= void= reference= and= later= on= assign= a= variable= with= a= concrete= data= type= to= it.= in= contrast ,= we= can= have= void= pointers.due= to= these= limitation= discussed= above,= the= references= in= c++= cannot= be= used= with= data= structures= like= a= linked= list.= please= note= that= in= java,= we= do= not= have= all= these= restrictions= or= limitation= on= references.passing= to= functionsin= our= previous= tutorials= on= functions,= we= have= already= discussed= the= ‘pass= by= reference’= parameter= technique= and= we= have= seen= the= swapping= of= two= numbers= example= using= this= technique.= we= skip= the= explanation= of= this= technique= in= this= section= and= only= present= a= swap= function= once= again= as= an= example.but= this= time= instead= of= swapping= numbers= we= are= going= to= swap= two=></endl;> <iostream> <string>(char - sstr1, char - sstr2) char</string> </iostream> </string> </iostream> </iostream> str2 - температура; - int main () - char sstr1 - ссылки; char sstr2 - указатели; cout с использованием <str1 ==></str1> <> <> <str2 ==></str2> <> <> <endl; swap(str1,= str2);=></endl;> <After></After> <endl;></endl;> <str1 ==></str1> <> <> <> <str2 ==></str2> <> <> <endl; return= 0;= }= output:str1=references str2=pointers after= swap….= str1=pointers str2=referencesSo in= this= program,= we= pass= strings= (char*)= to= the= swap= function.= the= formal= parameters= are= two= references= to= a= variable= of= type= char*.= we= note= that= when= two= values= are= swapped,= their= modification= is= reflected= in= the= calling= function= as= we= are= using= references/aliases= for= parameters.returning= referencesjust= like= returning= pointers= from= functions,= we= can= also= return= references= from= functions.= returning= references= from= a= function= also= return= an= implicit= pointer= to= the= return= value.= because= of= this= reason,= a= function= returning= a= reference= can= be= used= on= the= left-hand= side= of= the= assignment.let= us= an= example= of= returning= a= reference= from= a= function.= #include=></endl;> <iostream> namespace std; int myarray - No1, 0, 2, 3, 5; int' setValues (int < myarray= before= change=> < endl;= for= (= int= i=0; i=> < 5;= i++=)= {= cout=> < myarray[=> < i=> <]=; cout << myarray[i] << endl; } setValues(1) = 1; setValues(3) = 8; cout << value= after= change=> < endl;= for= (int= i=0; i=> < 5;= i++=)= {=> < myarray[=> < i=> <] = ; cout << myarray[i] << endl; } return 0; } Output:myarray before change myarray[0] = 1 myarray[1] = 0 myarray[2] = 2 myarray[3] = 3 myarray[4] = 5 Value after change myarray [0] = 1 myarray[1] = 1 myarray[2] = 2 myarray[3] = 8 myarray[4] = 5Screenshot for the same is shown below:As seen in the above code, we define a function setValues that return a reference and a parameter which is an integer. Inside the function, we just return the array reference to the position I in C++. In the main function, we print the values of the array. Then using the setValues function, we change the values of two elements in the array. Again we print the value of the array. One thing that we must note with references is that we can have a function return a reference only when the data is either static or global. It is illegal to return a reference to a local variable in C++. ConclusionReaders should note that the main use of references is for passing parameters to functions. In the upcoming tutorials, we will cover lambda functions/expressions in C++ before we jump to object-oriented programming in C++.=> Check Out The Best C++ Training Tutorials Here. Here.]=></] = ; cout myarray[i] << endl; } return 0; } Output:myarray before change myarray[0] = 1 myarray[1] = 0 myarray[2] = 2 myarray[3] = 3 myarray[4] = 5 Value after change myarray [0] = 1 myarray[1] = 1 myarray[2] = 2 myarray[3] = 8 myarray[4] = 5Screenshot for the same is shown below:As seen in the above code, we define a function setValues that return a reference and a parameter which is an integer. Inside the function, we just return the array reference to the position I in C++. In the main function, we print the values of the array. Then using the setValues function, we change the values of two elements in the array. Again we print the value of the array. One thing that we must note with references is that we can have a function return a reference only when the data is either static or global. It is illegal to return a reference to a local variable in C++. ConclusionReaders should note that the main use of references is for passing parameters to functions. In the upcoming tutorials, we will cover lambda functions/expressions in C++ before we jump to object-oriented programming in C++.=> Check Out The Best C++ Training Tutorials Here. Here. > i) //возвращает</iostream> ссылку на массив . .