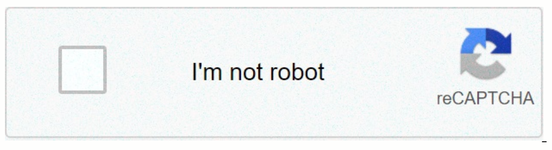

Continue

Arrays c pdf

V tem razdelku bomo ustvarili majhen program C, ki ustvari 10 naključnih številk in jih razvrsti. Za to bomo uporabili novo spremenljivko, imenovano matrika. Matrika vam dovoli, da razglašate in delate z zbirko vrednosti iste vrste. Morda boste na primer želeli ustvariti zbirko petih celo številov. Eden od načinov bi bil, da razglasiš pet vseh število neposredno: To je v redu, ampak kaj, če bi potrebovali tisoč vseh? Lažji način je razglasiti matriko petih številov: Do petih ločenih delnih številov znotraj te matrike dostopa indeks. Vse matrike se začnejo pri indeksu nič in gredo na n-1 v C. Tako, int a[5]; vsebuje pet elementov. Na primer: int a[5]; a[0] = 12; a[1] = 9; a[2] = 14; a[3] = 5; a[4] = 1; Ena od lepih stvari o indeksiranje matrike je, da lahko uporabite zanko za manipulacijo indeksa. Naslednja koda na primer inicializira vse vrednosti v matrici do 0: int a[5]; int i; za (i=0; i <5; i++)= a[i]=0; the= following= code= initializes= the= values= in= the= array= sequentially= and= then= prints= them= out:= #include=></5;> <stdio.h> int main() { int a[5]; int i; za (i=0; i <5; i++)= a[i]=i; for= (i=0;></5;> <5; i++)= printf(a[%d]=%d, i,= a[i]);= }= arrays= are= used= all= the= time= in= c.= to= understand= a= common= usage,= start= an= editor= and= enter= the= following= code:= #include=></5;> <stdio.h> #define MAX 10 int a[MAX]; int rand_seed=10; /* iz K&R - vrne naključni number med 0 i 32767.*/ int rand() { rand_seed = rand_seed * 1103515245 +12345; return (12345; return (nepodč. int)(rand_seed / 65536) % < MAX; i++) { a[i]=rand(); printf(%d,a[i]); } /* more stuff will go here in a minute */ return 0; } This code contains several new concepts. The #define line declares a constant named MAX and sets it to 10. Constant names are traditionally written in all caps to make them obvious in the code. The line int a[MAX]; shows you how to declare an array of integers in C. Note that because of the position of the array's declaration, it is global to the entire program. The line int rand_seed=10 also declares a global variable, this time named rand_seed, that is initialized to 10 each time the program begins. This value is the starting seed for the random number code that follows. In a real random number generator, the seed should initialize as a random value, such as the system time. Here, the rand function will produce the same values each time you run the program. The line int rand() is a function declaration. The rand function accepts no parameters and returns an integer value. We will learn more about functions later. The four lines that follow implement the rand function. We will ignore them for now. The main function is normal. Four local integers are declared, and the array is filled with 10 random values using a for loop. Note that the array a contains 10 individual integers. You point to a specific integer in the array using square brackets. So a[0] refers to the first integer in array, a[1] max;= i++)= {= a[i]=rand(); printf(%d,a[i]);= }= more= stuff= will= go= here= in= a= minute= */= return= 0;= }= this= code= contains= several= new= concepts.= the= #define= line= declares= a= constant= named= max= and= sets= it= to= 10.= constant= names= are= traditionally= written= in= all= caps= to= make= them= obvious= in= the= code.= the= line= int= a[max];= shows= you= how= to= declare= an= array= of= integers= in= c.= note= that= because= of= the= position= of= the= array's= declaration,= it= is= global= to= the= entire= program.= the= line= int= rand_seed=10 also= declares= a= global= variable,= this= time= named= rand_seed,= that= is= initialized= to= 10= each= time= the= program= begins.= this= value= is= the= starting= seed= for= the= random= number= code= that= follows.= in= a= real= random= number= generator ,= the= seed= should= initialize= as= a= random= value,= such= as= the= system= time.= here,= the= rand= function= will= produce= the= same= values= each= time= you= run= the= program.= the= line= int= rand()= is= a= function= declaration.= the= rand= function= accepts= no= parameters= and= returns= an= integer= value.= we= will= learn= more= about= functions= later.= the= four= lines= that= follow= implement= the= rand= function.= we= will= ignore= them= for= now.= the= main= function= is= normal.= four= local= integers= are= declared,= and= the= array= is= filled= with= 10= random= values= using= a= for= loop.= note= that= the= array= a= contains= 10= individual= integers.= you= point= to= a= specific= integer= in= the= array= using= square= brackets.= so= a[0]= refers= to= the= first= integer= in= the= array ,= a[1]=></ MAX; i++) { a[i]=rand(); printf(%d,a[i]); } /* more stuff will go here in a minute */ return 0; } This code contains several new concepts. The #define line declares a constant named MAX and sets it to 10. Constant names are traditionally written in all caps to make them obvious in the code. The line int a[MAX]; shows you how to declare an array of integers in C. Note that because of the position of the array's declaration, it is global to the entire program. The line int rand_seed=10 also declares a global variable, this time named rand_seed, that is initialized to 10 each time the program begins. This value is the starting seed for the random number code that follows. In a real random number generator, the seed should initialize as a random value, such as the system time. Here, the rand function will produce the same values each time you run the program. The line int rand() is a function declaration. The rand function accepts no parameters and returns an integer value. We will learn more about functions later. The four lines that follow implement the rand function. We will ignore them for now. The main function is normal. Four local integers are declared, and the array is filled with 10 random values using a for loop. Note that the array a contains 10 individual integers. You point to a integer in the array using square brackets. So a[0] refers to the first integer in the array, a[1] > 32768; } int main() { int i,t,x,y; /* fill array */ for (i=0; i</stdio.h> </stdio.h> </stdio.h> to the other and so on. A line that starts with /* and ends with */ is called a comment. The compiler completely ignores the line. You can place notes with yourself or other programmers in your comments. Now add the following code on the site of several things ... comment: /* bubble sort array */ for (x=0; x < MAX-1; x++) for (y=0; y < MAX-x-1; y++) if (a[y] > a[y+1]) { t=a[y]; a[y]=a[y+1]; a[y+1]=t; } /* print sorted array */ printf(----------); for (i=0; i < MAX; i++) printf(%d, a[i]); This code sorts random values and is typed in sorted order. Each time you run it, you will get the same values. To change the values that are sorted, change the value rand_seed each time you start the program. The only easy way to truly understand what this code is to execute it manually. This means that the MAX 4 is to make it a little more manageable, take a sheet of paper and pretend you're a computer. Draw an array on paper and insert four random, unsorted values into a field. Execute each row of the code sorting section and pull out exactly what happens. You will find that there are always higher values in the matrix pushed towards the bottom of the matrix and smaller bubble values up towards the top. This site is not available in your PeopleImages.com/Digital Vision/Getty Images In many educational institutions, C is considered average. In some graduate schools, C is the lowest possible transitional score. C is equivalent to a numerical estimate in the low 70s. On a scale of 4.0, which is widely used in faculties and universities, C is usually around 2.0. A C is significantly lower than the top grade, A, which is given for exceptional works. But it represents about 92 per cent or more, and is at the top of the scale of 4.0. B is registered at 3,0 and indicates approximately 82 per cent or more. However, D is only about 62 per cent or 1.0. The plus sign attached to the rating adds a few points, while the minus subtracts the same amount. In addition, H associated with the assessment indicates that it is for an honorary course. Some schools have other sorting systems. Sometimes these are used only in selected courses. For example, a student earns an S, for satisfactory or U, for unsatisfactory, no numbers or other letters. Some classes are available on a Pass/Fail basis. These indications show that the requirements have been met or have not been met. Arrays and pointers are intimately linked in C. A full understanding of the relationship between the two probably requires several days of study and experimentation, but it is well worth the effort. Let's start with a simple example of arrays in C:#define MAX 10 int main() { int a[MAX]; int b[MAX]; int i; for(i=0; i<MAX; i++) a[i]=i; b=a; return 0; } Enter code and try to migrate it. You'll find that C won't move him. To copy a to b, you must enter something similar instead: for (i=0; i<MAX; i++) b[i]=a[i]; Or more concise: for (i=0; i<MAX; b[i]=a[i], i++); Even better, use the memcpy utility in string.h. The fields in C are unusual in that variables a and b are not, technically, the matrix itself. Instead, they are constant indicators of arrays. a and b permanently point to the first elements of their individual matriquees -- they have titles a[0] and b[0] respectively. Because they are constant indicators, you cannot change their addresses. Statement a=b; therefore does not work. Because a and b are pointers, you can do several interesting things with pointers and arrays. For example, the following code works: #define MAX 10 void main() { int a[MAX]; int i; p=p;p=a; za(i=0; i<MAX; i++) a[i]=i; printf(%d,*p); } Statement p=a; it works because it's a pointer. Technically, points to title 0. element of the actual matrix. This item is an all-in-one number, so the cursor is on a single number. Therefore, declaring p as an indicator on all number and setting equal parts. Another way to say exactly the same would be to replace p=a; with p=&a[0];. Because it contains a title[0], they mean a and &a[0] in the same way. Now that p points to the 0. element, you can do some pretty weird things with it. A variable is a constant indicator and cannot be changed, but p is not subject to such restrictions. C actually encourages you to move it using the arithmetic pointer. For example, if you say p++;, the compilator knows that p points to all numbers, so this statement p increases the corresponding number of bytes to move it to the next element of the array. If p is an index to an array of 100-byte-long structures, p++; to move p by 100 bytes. C takes care of the detail of the size of the item. You can copy field a to b by using cursors as well. The following code can be replaced (for i=0; i<MAX; a[i]=b[i], i++); : p=a; q=b; for (i=0; i<MAX; i++) { *q = *p; q++; p++; } You can shorten this code as follows: p=a; q=b; for (i=0; i<MAX; i++) *q++ = *p++; And you can further reduce it to: (p=a,q=b,i=0; i<MAX; *q++ = *p++, i++); What if you go over the end of field a or b with pointers p or q? C does not care - this quickly increases p and q, copying the page over other variables with abandoned. You need to be careful when indexing in boxes in C because C assumes that you know what you are doing. A field, such as a or b, can be hinged with functions in two different ways. Imagine a down a function that accepts an integer type as a parameter and tests the contents of an array in a stdout. There are two ways to code dump: void dump(int a[],int nia) { int i; for (i=0; i<nia; i++) printf(%d, a[i]); } or: shion gap(int*p,int nia) { int i; for (i=0; i<nia; i++) printf(%d, *p++); } Nia the variable is required in such a way that the matrix size is known. Note that only the pointer to the matrix is transferred to the function, not the contents of the matrix. Note also that functions C can accept variable-size arrays as parameters. Parameters.