

1a). #include <stdio.h>

/* Comparator pentru selection sort*/

int cmp(int *a, int *b)

{

 return *a > *b;

}

void selectionSort(int nodes[], int first, int last, int (*cmp)(int * a, int* b))

{

 int position = 0, c = 0, d = 0;

 /* Cautam minimum n - 1 ori*/

 for (c = first; c < (last - 1); c++) {

 position = c;

 /* Luam mereu elementele din fata si le schimbam daca indeplinesc conditia */

 for (d = c + 1; d < last; d++) {

 if (cmp(&nodes[position], &nodes[d]))

 position = d;

 }

 /* Daca nu este elementul insusi, facem swap cu elementul mai mic*/

 if (position != c) {

 int t = nodes[c];

 nodes[c] = nodes[position];

 nodes[position] = t;

 }

 }

}

int main()

{

 int n = 10;

```
/* Facem vectorul n cu aceste valori*/
```

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

```
/* Apelam functia */
```

```
selectionSort(nodes, 0, n, &cmp);
```

```
/* Printam vectorul*/
```

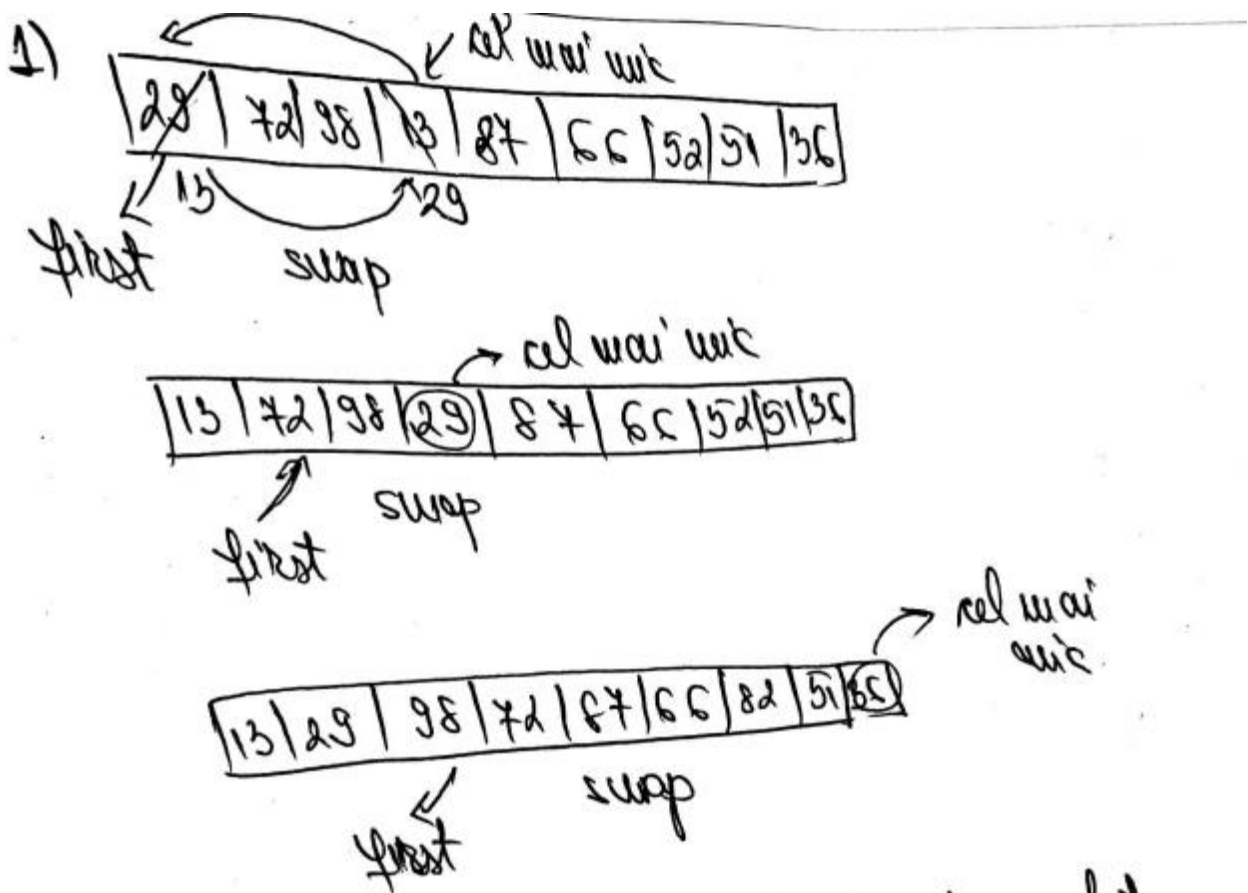
```
for (int i = 0; i < n; i++) {
```

```
    printf("%d ", nodes[i]);
```

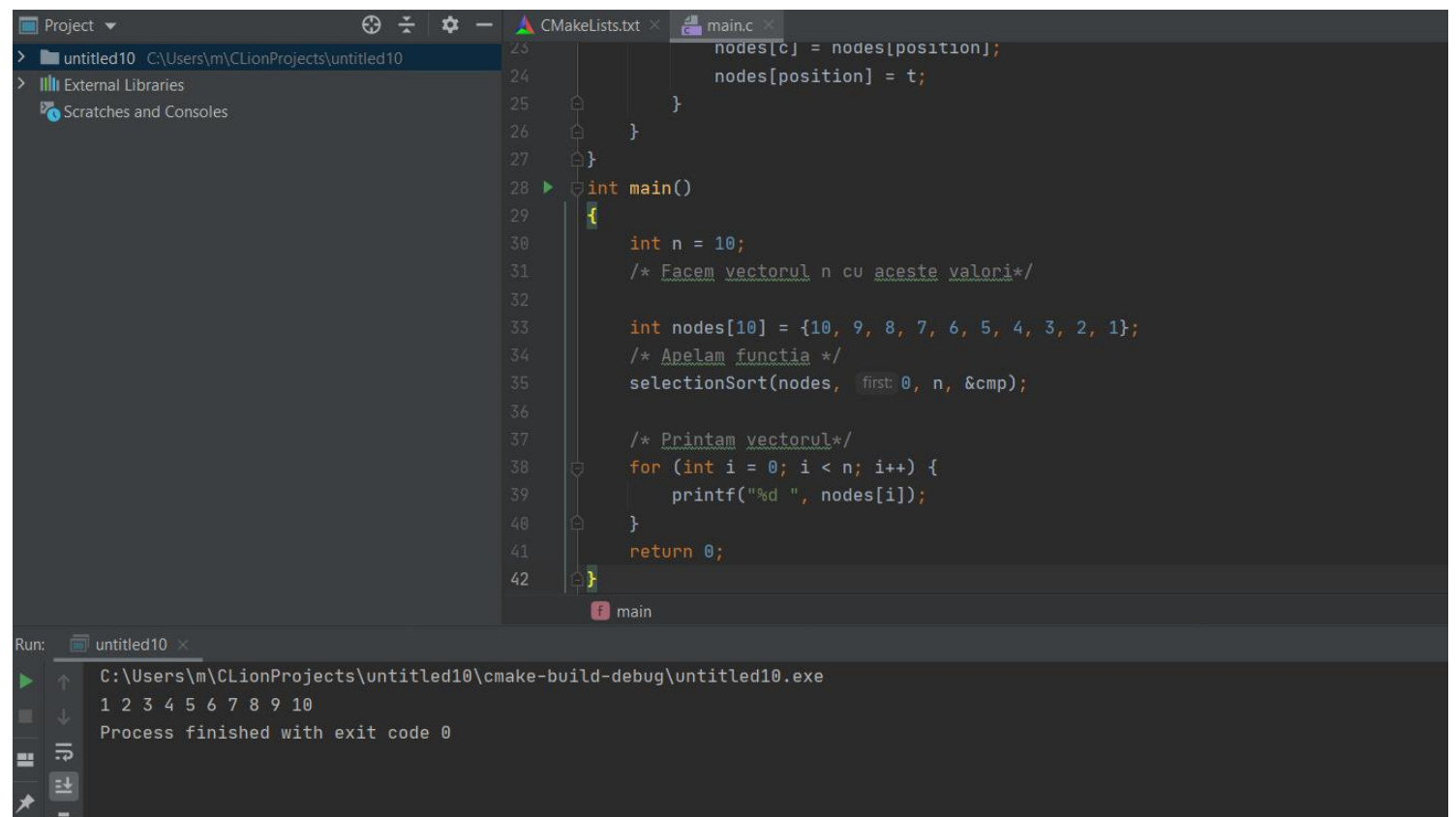
```
}
```

```
return 0;
```

```
}
```



Si tot asa pâna când vectorul este sortat.



The image shows a C++ IDE with a project named 'untitled10'. The main file, 'main.c', contains a selection sort algorithm. The code is as follows:

```
23     nodes[c] = nodes[position];
24     nodes[position] = t;
25 }
26 }
27 }
28 int main()
29 {
30     int n = 10;
31     /* Facem vectorul n cu aceste valori*/
32
33     int nodes[10] = {10, 9, 8, 7, 6, 5, 4, 3, 2, 1};
34     /* Apelam functia */
35     selectionSort(nodes, first: 0, n, &cmp);
36
37     /* Printam vectorul*/
38     for (int i = 0; i < n; i++) {
39         printf("%d ", nodes[i]);
40     }
41     return 0;
42 }
```

The Run console shows the output of the program:

```
Run: untitled10 x
C:\Users\m\CLionProjects\untitled10\cmake-build-debug\untitled10.exe
1 2 3 4 5 6 7 8 9 10
Process finished with exit code 0
```

```
1b). #include <stdio.h>

#include <stdlib.h>

struct node_btree
{
    int data;

    struct node_btree* left, *right;
};

int get_height(struct node_btree* node)
{
    if (node == NULL)
        return 0;
    else {
        /* calculam inaltimea pe fiecare subarbore */
        int lheight = get_height(node->left);
        int rheight = get_height(node->right);

        /* Si trebuie sa o luam pe cel mai mare dintre cele 2 */
        if (lheight > rheight)
            return(lheight + 1);
        else return(rheight + 1);
    }
}

/* Functia aceasta printeaza elementele de pe nivelul curent */
void printLevel(struct node_btree* root, int level)
{
    /* Conditia de oprire */
    if (root == NULL)
```

```

    return;

/* Am ajuns la final */

if (level == 1)

    printf("%d ", root->data);

else if (level > 1)

{

    /* Recursivitate pe bratul stang si pe bratul drept */

    printLevel(root->left, level - 1);

    printLevel(root->right, level - 1);

}

}

void tl_tree(struct node_btree* r)

{

    /* Luam inaltimea arborelui */

    int height = get_height(r);

    for (int i = 1; i <= height; i++)

        printLevel(r, i);

}

/* Functie cu care creem noduri pentru testare */

struct node_btree* newNode(int data)

{

    /* Creem nodul dinamic */

    struct node_btree* node = (struct node_btree*)

        malloc(sizeof(struct node_btree));

    /* Punem informatia in el dar si copii ii setam pe null deoarece

    * nu are copii in acel moment

    */

    node->data = data;

```

```
node->left = NULL;

node->right = NULL;


return(node);
}

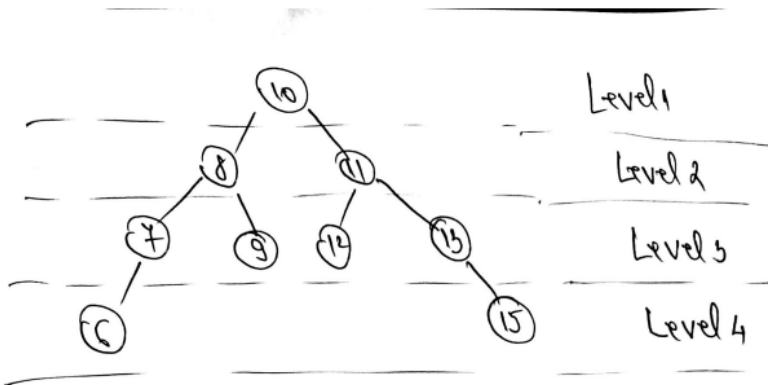
int main()
{
    struct node_btree *root = newNode(1);

    root->left    = newNode(2);
    root->right   = newNode(3);
    root->left->left = newNode(4);
    root->left->right = newNode(5);


    printf("Parcurerea in latime pe arbore este: \n");

    tl_tree(root);


    return 0;
}
```



Încep cu Level 1; Apăsăm: 10

Level 2: 8, 11

Level 3: 7, 9, 12, 13

Level 4: 6, 15

Se folosește de recursivitate, când terminăm un subarbore se întorcem și se duce pe celălalt subarbore.

```

project
untitled11 C:\Users\m\CLionProjects\untitled11
External Libraries
Scratches and Consoles

56  /* Punem informatia in el dar si copii ii setam pe null deoarece
57  * nu are copii in acel moment
58  */
59  node->data = data;
60  node->left = NULL;
61  node->right = NULL;
62
63  return(node);
64
65  int main()
66  {
67      struct node_btree *root = newNode( data: 1);
68      root->left = newNode( data: 2);
69      root->right = newNode( data: 3);
70      root->left->left = newNode( data: 4);
71      root->left->right = newNode( data: 5);
72
73      printf("Parcurea in latime pe arbore este: \n");
74      tl_tree(root);
75
76      return 0;
77  }

main

untitled11
C:\Users\m\CLionProjects\untitled11\cmake-build-debug\untitled11.exe
Parcurea in latime pe arbore este:
1 2 3 4 5
Process finished with exit code 0
  
```

```
2. #include <stdio.h>
```

```
#include <stdlib.h>
```

```
#include <math.h>
```

```
#define len(x) ((int)log10(x)+1)
```

```
/* Nodurile din arborele Huffman */
```

```
struct node{
```

```
    /* Valoarea */
```

```
    int value;
```

```
    /* Letter */
```

```
    char letter;
```

```
    /* Copii din stanga si din dreapta */
```

```
    struct node *left,*right;
```

```
};
```

```
typedef struct node Node;
```

```
/* 81 = 8.1%, 128 = 12.8% si asa mai departe. Cele 26 de litere sunt spatiul de encodare.*/
```

```
int romanianLetterFrequencies [26] = {81, 15, 28, 43, 128, 23, 20, 61, 71, 2, 1, 40, 24, 69, 76, 20, 1, 61, 64, 91, 28, 10, 24, 1, 20, 1};
```

```
/* Comparator pentru selection sort*/
```

```
int cmp(int *a, int *b)
```

```
{
```

```
    return *a > *b;
```

```
}
```

```
void selectionSort(int nodes[], int first, int last, int (*cmp)(int * a, int* b))
```

```
{
```



```

int position = 0, c = 0, d = 0;

/* Cautam minimum n - 1 ori*/
for (c = first; c < (last - 1); c++) {

    position = c;

    /* Luam mereu elementele din fata si le schimbam daca indeplinesc conditia */
    for (d = c + 1; d < last; d++) {

        if (cmp(&nodes[position], &nodes[d]))

            position = d;

    }

    /* Daca nu este elementul insusi, facem swap cu elementul mai mic*/
    if (position != c) {

        int t = nodes[c];

        nodes[c] = nodes[position];

        nodes[position] = t;

    }

}

}

/*Cream arborele huffman si returnam adresa lui */
void buildHuffmanTree(Node **tree){

    Node *temp;

    Node *array[26];

    int i, subTrees = 26;

    int smallOne = 0, smallTwo = 0;

    /* Il folosim pentru sortare*/

    int array2[26] = {0};

    /* Alocam memorie pentru fiecare element din arborele huffman*/

    for (i = 0; i < 26; i++){

```

```

array[i] = malloc(sizeof(Node));

array[i]->value = romanianLetterFrequencies[i];

array[i]->letter = i;

array[i]->left = NULL;

array[i]->right = NULL;

}

```

```

for (int i = 0; i < 26; i++)

```

```

    array2[i] = array[i]->value;

```

```

selectionSort(array2, 0, 26, &cmp);

```

```

while (subTrees > 1){

```

```

    /* Luam primele cele mai mici 2 valori*/

```

```

    smallOne=array2[0];

```

```

    smallTwo=array2[1];

```

```

    /* Retinem primul element*/

```

```

    temp = array[smallOne];

```

```

    /* Alocam memorie pentru elementul cel mai mic ****REINITIALIZARE***** pentru el sii copii lui*/

```

```

    array[smallOne] = malloc(sizeof(Node));

```

```

    array[smallOne]->value=temp->value+array[smallTwo]->value;

```

```

    array[smallOne]->letter=127;

```

```

    array[smallOne]->left=array[smallTwo];

```

```

    array[smallOne]->right=temp;

```

```

    /* Al doilea va deveni copil*/

```

```

    array[smallTwo]->value=-1;

```

```

    subTrees--;

```

```

}

```

```

/* Am creat arborele*/

```

```

*tree = array[smallOne];

```

```

}

```

```
int main()

{

    /* arborele nostru */

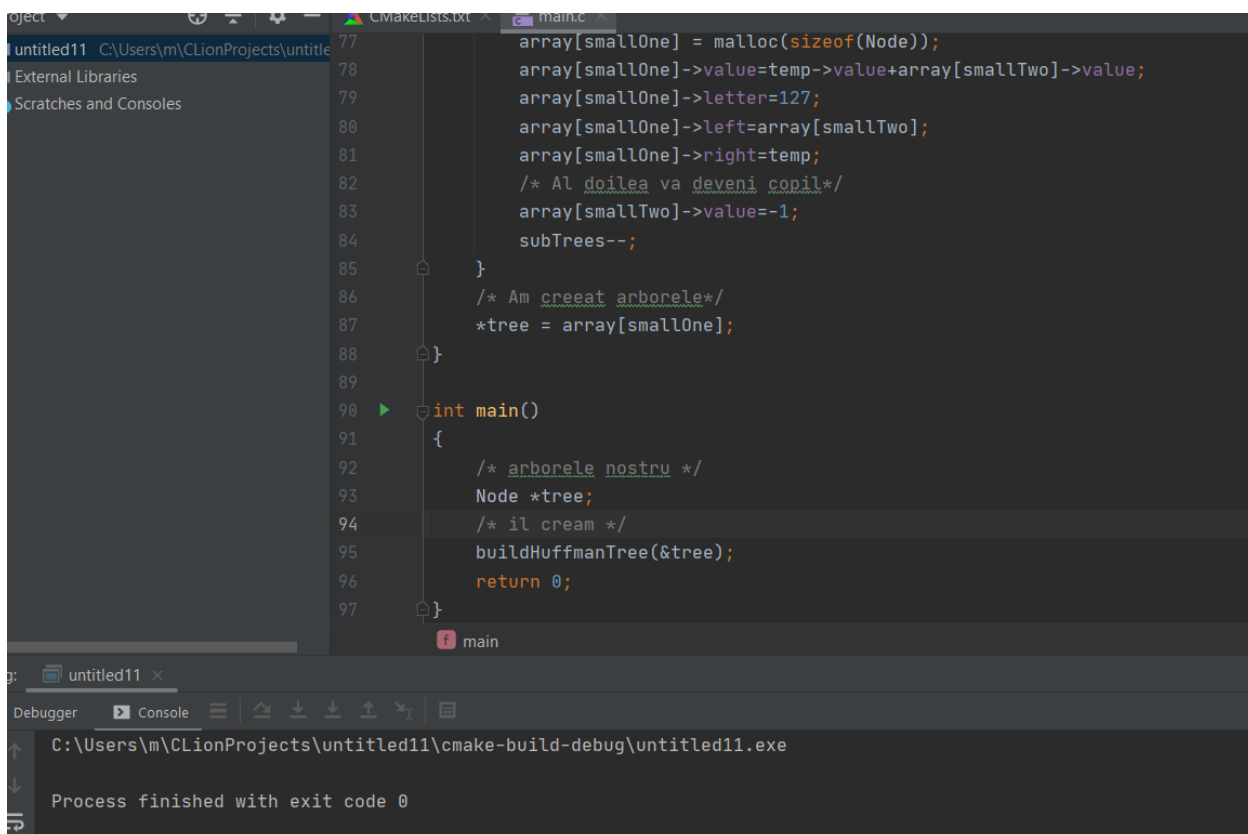
    Node *tree;

    /* il cream */

    buildHuffmanTree(&tree);

    return 0;

}
```



The screenshot displays the CLion IDE interface. The main editor window shows a C++ source file with the following code:

```
77     array[smallOne] = malloc(sizeof(Node));
78     array[smallOne]->value=temp->value+array[smallTwo]->value;
79     array[smallOne]->letter=127;
80     array[smallOne]->left=array[smallTwo];
81     array[smallOne]->right=temp;
82     /* Al doilea va deveni copil */
83     array[smallTwo]->value=-1;
84     subTrees--;
85 }
86 /* Am creat arborele */
87 *tree = array[smallOne];
88 }
89
90 int main()
91 {
92     /* arborele nostru */
93     Node *tree;
94     /* il cream */
95     buildHuffmanTree(&tree);
96     return 0;
97 }
```

The left sidebar shows the project structure with 'untitled11' and 'External Libraries'. The bottom status bar indicates the process finished with exit code 0.