Задача № 1/5

Задание № 1.

Напишите программу, которая считывает два целых числа *a* и *b* и выводит наибольшее значение из них. Числа — целые от 1 до 1000.

При решении задачи можно пользоваться только целочисленными арифметическими операциями +, -, \*, //, %, =. Нельзя пользоваться нелинейными конструкциями: ветвлениями, циклами, функциями вычисления модуля, извлечения квадратного корня.

#include <iostream>

using namespace std;

int main()

{

int a,b,c;

cin >> a>>b;

c = (((a / b) \* a) + ((b / a) \* b)) / ((a / b) + (b / a));

cout << c;

return 0;

}

Задание № 2.

Подсчитать значение числа a в степени b с логарифмической сложностью.

#include <iostream>

using namespace std;

int binpow(int a, int n) {

int res = 1;

while (n) {

if (n & 1)

res \*= a;

a \*= a;

n >>= 1;

}

return res;

}

int main() {

int x, y;

cin >> x >> y;

cout << binpow(x,y);

return 0;

}

Задание № 3.

Написать алгоритм (функцию) определения четности целого числа, который будет аналогичен нижеприведенному по функциональности, но отличен по своей сути.

Объяснить плюсы и минусы обеих реализаций.

C/C++ example:

bool isEven(int value){return value%2==0;}

Аналог на С++:

#include <iostream>

using namespace std ; // используем именное пространство std

int main()

{

setlocale(LC\_ALL, "Russian");

int number ;

cout << "Введите число : " ;

cin >> number;

if (number % 2 == 0)

cout << number << " четное число " << endl;

else

cout << number << " нечетное число " << endl;

system("PAUSE");

return 0 ;

}

На ассемблере:

isEven(int):

push rbp

mov rbp, rsp

mov DWORD PTR [rbp-4], edi

mov eax, DWORD PTR [rbp-4]

and eax, 1

test eax, eax

sete al

pop rbp

ret

isEven(int): # @isEven(int)

push rbp

mov rbp, rsp

mov dword ptr [rbp - 4], edi

mov eax, dword ptr [rbp - 4]

cdq

mov ecx, 2

idiv ecx

cmp edx, 0

sete sil

and sil, 1

movzx ecx, sil

mov eax, ecx

pop rbp

ret

Задание № 4.

Написать минимум по 2 реализации циклических буферов FIFO и LIFO.

Объяснить плюсы и минусы каждой реализации.

**– На практике используются и LIFO, и FIFO – всё зависит от цели и от роли использующего, и от особенностей памяти последовательного доступа(стека)!**

Первая реализация:

#include <cstdlib>

#include <iostream>

using namespace std;

struct struc{

    char a;

    int n;

       };

 void input(struc \*str,int m){

      for(int i=0; i<m; i++  ){

          cout<<i+1<< "element structuri a "<<" " ;

          cin>> str[i].a;

          str[i].n=1+i;

              }

              cout<<endl;

      }

void outFIFO(struc \*str,int m){

     cout<<"FIFO"<<endl;

  for(int i=0; i<m; i++ ){

   cout<<str[i].a<<" "<<str[i].n<<endl;

}}

void outLIFO(struc \*str, int m){

     cout<<"LIFO"<<endl;

      for(int i=1; i<m+1; i++ ){

            cout<<str[m-i].a<<" "<<str[m-i].n<<endl;

              }

     }

main()

{

 cout<<"kolicestvo elementov";

 int m;

cin>>m;

cout<<endl;

struc str[m];

input(str,m);

outFIFO(str,m);

outLIFO(str,m);

    system("PAUSE");

    return EXIT\_SUCCESS;

}

Вторая реализация:

// работа LIFO  
// используя стек в C ++  
#include<bits/stdc++.h>

using namespace std;

// Вставка элемента в верхнюю часть стека

stack<int> stack\_push(stack<int> stack)

{

    for (int i = 0; i < 5; i++)

    {

        stack.push(i);

    }

    return stack;

}

// выталкивающий элемент сверху стека

stack<int> stack\_pop(stack<int> stack)

{

    cout << "Pop :";

    for (int i = 0; i < 5; i++)

    {

        int y = (int)stack.top();

        stack.pop();

        cout << (y) << endl;

    }

    return stack;

}

// Отображение элемента в верхней части стека

void stack\_peek(stack<int> stack)

{

    int element = (int)stack.top();

    cout << "Element on stack top : " << element << endl;

}

// Поиск элемента в стеке

void stack\_search(stack<int> stack, int element)

{

    int pos = -1,co = 0;

    while(stack.size() > 0)

    {

        co++;

        if(stack.top() == element)

        {

            pos = co;

            break;

        }

        stack.pop();

    }

    if (pos == -1)

        cout << "Element not found" << endl;

    else

        cout << "Element is found at position " << pos << endl;

}

// Код драйвера

int main()

{

    stack<int> stack ;

    stack = stack\_push(stack);

    stack = stack\_pop(stack);

    stack = stack\_push(stack);

    stack\_peek(stack);

    stack\_search(stack, 2);

    stack\_search(stack, 6);

    return 0;

}

Задание № 5.

Написать реализацию бинарного поиска.

#include <iostream>

using namespace std;

int binarySearch(int arr[], int size, int k) {

int left = 0;

int right = size - 1;

if (arr[left] == k) {

return left;

}

else if (arr[right] == k) {

return right;

}

else {

int middle;

while ((left <= right)) {

middle = (left + right) / 2;

if (arr[middle] == k) {

return middle;

}

else if (k < arr[middle]) {

right = middle - 1;

}

else {

left = middle + 1;

}

}

}

return -1;

}

int main() {

int array[10] = { 100,90,80,70,60,50,40,30,20,10 };

int n;

cin >> n;

for (int i = 0; i < sizeof(array) / sizeof(int); i++) { //Сортировка

for (int j = 0; j < sizeof(array) / sizeof(int) - 1 - i; j++) {

if (array[j] > array[j + 1]) {

int temp = array[j];

array[j] = array[j + 1];

array[j + 1] = temp;

}

}

}

cout << binarySearch(array, sizeof(array) / sizeof(int), n); //-1 - не найлено

return 0;

}