  
**Колледж космического машиностроения и технологий**

**РАБОЧАЯ ПРОГРАММА УЧЕБНОЙ ПРАКТИКИ УП.01.01 ПО МОДУЛЮ ПМ.01 «РАЗРАБОТКА ПРОГРАММНЫХ МОДУЛЕЙ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ ДЛЯ КОМПЬЮТЕРНЫХ СИСТЕМ»**

**Выполнил:**

**Ухин Константин**

**Студент 3 курса**

**Группы:**

**П3-19**

**Проверил преподаватель**

**Раздел 1. Системное программирование и решение алгоритмических задач.**

Тема 1.1 Работа с ресурсами.

Работа с библиотекой <time.h>. / <stdlib.h>

#include <time.h>

#include <stdlib.h>

#include <stdio.h>

#include <conio.h>

#include <locale.h>

//#define INFILE "C:\Users\Indikator\Desktop\mts1.txt"

//#define OUTFILE "C:\Users\Indikator\Desktop\mts2.txt"

main()

{

setlocale(0,"");

FILE \*inputfile,\*outputfile;

int m,n,i,j,l,temp,max=0;

inputfile = fopen("mts1.txt", "r");

fscanf(inputfile, "%d%d",&m,&n);

printf("%d %d",m,n);

int mas[m][n];

fclose(inputfile);

outputfile=fopen("mts2.txt","w");

srand(time(NULL));

for(i=0; i<m; i++)

{

for(j=0; j<n; j++)

fprintf(outputfile, "%5d", rand()%100);

fprintf(outputfile,"\n");

}

fclose(outputfile);

outputfile=fopen("mts2.txt", "r");

for(i=0; i<m; i++)

{

for(j=0;j<n; j++)

fscanf(outputfile,"%d",&mas[i][j]);

}

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

for(j=0; j<n; j++)

if(mas[i][j]>max){max=mas[i][j];}

}

outputfile=fopen("mts2.txt","a");

fprintf(outputfile,"Максимальное число в массиве=%d",max);

}

Используется библиотека time.h для избегания повторений рандомных элементов массива при повторных запусков программы.

Библиотека stdlib.h используется для создания случайных чисел.

Тема 1.2 Абстрактные типы данных.

Реализация АТД «Стек» Средства отладки программ.

#include <iostream>

#include <locale>

using namespace std;

int main() {

setlocale(LC\_ALL, "RUS");

int steck[20];

int i = -1;

for (int j = 0; j < 6; j++) {

int a;

cin >> a;

i++;

steck[i] = a; // добавляем в стек элемент

}

if (i == -1) cout << "Стек пуст"<<endl; // проверяем пуст ли стек (нет)

cout << steck[i] << " это верхний элемент стека"<<endl;

cout << "Сейчас мы удалим верхний элемент"<<endl;

i--; // уменьшаем i на один

system("pause");

return 0;

}

Реализация АТД «Очередь».

#include <iostream>

#include <queue>

using namespace std;

int main() {

priority\_queue <long long> nums;

long long n;

cin >> n;

long long number = 0;

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

cin >> number;

nums.push(-number);

}

long long min\_sum = 0;

long long a = 0;

long long b = 0;

while ( !nums.empty() )

{

a = nums.top();

nums.pop();

if (nums.empty()) break;

b = nums.top();

nums.pop();

min\_sum -= a+b;

nums.emplace(a+b);

}

cout << " " << min\_sum;

return 0;

}

Реализация АТД «Циклический список».

Реализация АТД «Множество».

#include <iostream>

#include <set>

using namespace std;

set <int> a;

set <int> b;

set <int> c;

int main()

{

a.insert(1);

a.insert(2);

cout << a.size() << endl;

b.insert(2);

b.insert(3);

cout << b.size() << endl;

c.insert(a.begin(), a.end());

c.insert(b.begin(), b.end());

cout << c.size() << endl;

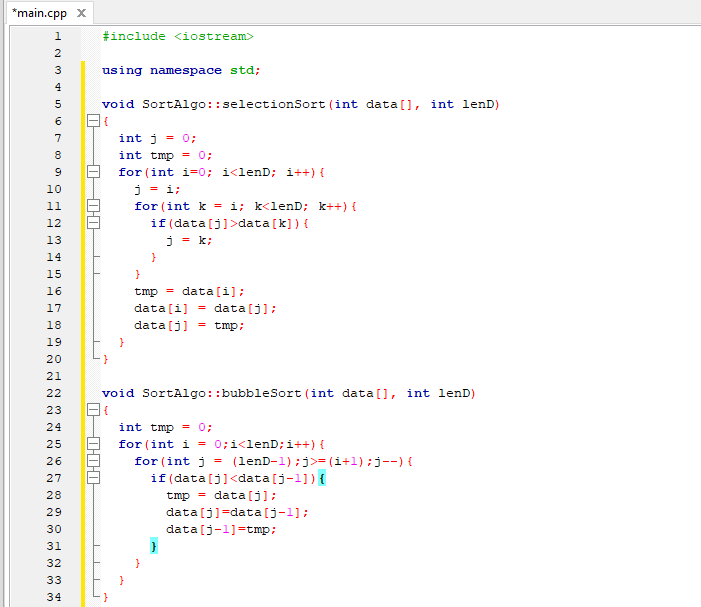
return 0;

}

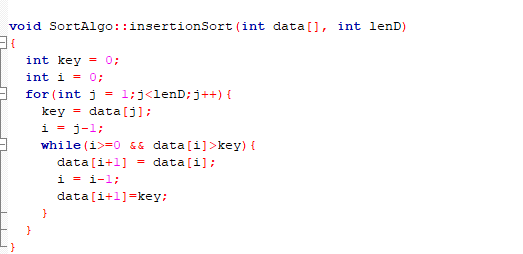
Реализовано в виде объединение множеств

1. Разработка комплекса «Сортировки».

Сортировка выбором

Пузырьковая сортировка.   


Сортировка вставками.



«Разделяй и властвуй».

#include <iostream>

using namespace std;

int sumeven(int\* a, int l, int r)

{

if (r >= 6) return 0;

if (l > r ) return 0;

if (l >= r)

{

if (a[l] % 2 == 0)

{

return a[l];

}

else

{

return 0;

}

}

int mid = (l + r) / 2;

int s1 = sumeven(a, l, mid);

int s2 = sumeven(a, mid + 1, r);

return s1 + s2;

}

int main()

{

int n=6;

int a[6]={1,2,3,48,5,6};

cout<<sumeven(a,0,n-1);

return 0;

}

1. «Жадный» алгоритм.

#include <iostream>

#include <bits/stdc++.h>

using namespace std;

bool comp(int a, int b)

{

return a > b;

}

int main()

{

int m[1000], sum = 0, m\_size;

cin >> m\_size;

for (int i = 0;i < m\_size; i ++)

cin >> m[i];

sort(m, m + m\_size, comp);

for (int i = 0; i < m\_size && max >= sum; i ++)

sum += m[i];

cout << sum << "\n";

return 0;

}

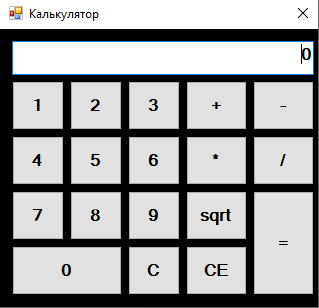
1. Работа с деревьями.
2. Реализация метода динамического программирования.
3. През по сортировки.

**Раздел 2.Работа с текстом.**

Тема 2.2 Разработка Калькулятора.

1. Постановка задачи.

Создание калькулятора с интерфейсом.



#pragma endregion

double a, b;

double res;

String^ znak;

private: System::Void Cler\_Click(System::Object^ sender, System::EventArgs^ e) {

//CE

txtDisplay->Text = "0";

lblShow0p->Text = "";

}

private: System::Void button11\_Click(System::Object^ sender, System::EventArgs^ e) {

//C

txtDisplay->Text = "0";

}

private: System::Void button\_Click(System::Object^ sender, System::EventArgs^ e) {

Button^ Numbers = safe\_cast<Button^>(sender);

if (txtDisplay->Text == "0")

{

txtDisplay->Text = Numbers->Text;

}

else

{

txtDisplay->Text = txtDisplay->Text + Numbers->Text;

}

}

private: System::Void Arithmetic(System::Object^ sender, System::EventArgs^ e) {

Button ^ op = safe\_cast<Button^>(sender);

a = Double::Parse(txtDisplay->Text);

txtDisplay -> Text = "";

znak = op -> Text;

lblShow0p->Text = System::Convert::ToString(znak) + " " + a;

}

private: System::Void btnEquals\_Click(System::Object^ sender, System::EventArgs^ e) {

lblShow0p->Text = "";

b = Double::Parse(txtDisplay->Text);

if (znak == "+")

{

res = a + b;

txtDisplay->Text = System::Convert::ToString(res);

}

else if (znak == "-")

{

res = a - b;

txtDisplay->Text = System::Convert::ToString(res);

}

else if (znak == "/")

{

res = a / b;

txtDisplay->Text = System::Convert::ToString(res);

}

else if (znak == "\*")

{

res = a \* b;

txtDisplay->Text = System::Convert::ToString(res);

}

}

private: System::Void button16\_Click(System::Object^ sender, System::EventArgs^ e) {

a = Double::Parse(txtDisplay->Text);

lblShow0p->Text = System::Convert::ToString("sqrt" + "(" + (txtDisplay->Text) + ")");

a = Math::Sqrt(a);

txtDisplay->Text = System::Convert::ToString(a);

}

};

}

Тема 2.2 Разработка текстового редактора.

1. Постановка задачи.

#include <windows.h> // подключение библиотеки с функциями API

HINSTANCE hb1;

// Глобальные переменные:

HINSTANCE hInst; // Указатель приложения

LPCTSTR szWindowClass = "Kravets";

LPCTSTR szTitle = "Program";

// Предварительное описание функций

ATOM MyRegisterClass(HINSTANCE hInstance);

BOOL InitInstance(HINSTANCE, int);

LRESULT CALLBACK WndProc(HWND, UINT, WPARAM, LPARAM);

// Основная программа

int APIENTRY WinMain(HINSTANCE hInstance,

HINSTANCE hPrevInstance,

LPSTR lpCmdLine,

int nCmdShow)

{

MSG msg;

// Регистрация класса окна

MyRegisterClass(hInstance);

// Создание окна приложения

if (!InitInstance (hInstance, nCmdShow))

{

return FALSE;

}

// Цикл обработки сообщений

while (GetMessage(&msg, NULL, 0, 0))

{

TranslateMessage(&msg);

DispatchMessage(&msg);

}

return msg.wParam;

}

//Регистрация класса

ATOM MyRegisterClass(HINSTANCE hInstance)

{

WNDCLASSEX wcex;

wcex.cbSize = sizeof(WNDCLASSEX);

wcex.style = CS\_HREDRAW | CS\_VREDRAW ; // стиль окна

wcex.lpfnWndProc = (WNDPROC)WndProc; // оконная процедура

wcex.cbClsExtra = 0;

wcex.cbWndExtra = 0;

wcex.hInstance = hInstance; // указатель приложения

wcex.hIcon = LoadIcon(NULL, IDI\_APPLICATION); // опре-деление иконки

wcex.hCursor = LoadCursor(NULL, IDC\_ARROW); // опреде-ление курсора

wcex.hbrBackground = (HBRUSH)GetStockObject(WHITE\_BRUSH); // ус-тановка фона

wcex.lpszMenuName = NULL; // определение меню

wcex.lpszClassName = szWindowClass; // имя класса

wcex.hIconSm = NULL;

return RegisterClassEx(&wcex); // регистрация класса окна

}

//Создание окна приложения

BOOL InitInstance(HINSTANCE hInstance, int nCmdShow)

{

HWND hWnd;

hInst = hInstance; // сохраняет указатель приложения в переменной hInst

hWnd=CreateWindow(szWindowClass, // имя класса окна

szTitle, // имя приложения

WS\_OVERLAPPEDWINDOW, // стиль окна

CW\_USEDEFAULT, // положение по Х

CW\_USEDEFAULT, // положение по Y

1000, // размер по Х

800, // размер по Y

NULL, // описатель родительского окна

NULL, // описатель меню окна

hInstance, // указатель приложения

NULL); // параметры создания.

if (!hWnd) // Если окно не создалось, функция возвращает FALSE

{

return FALSE;

}

ShowWindow(hWnd, nCmdShow); // Показать окно

UpdateWindow(hWnd); // Обновить окно

return TRUE; //Успешное завершение функции

}

LRESULT CALLBACK WndProc(HWND hWnd, UINT message, WPARAM wParam, LPARAM lParam)

{

PAINTSTRUCT ps;

HDC hdc;

RECT rt;

static HINSTANCE hInstance;

static HWND hb1;

switch (message)

{

case WM\_CREATE: // Сообщение приходит при создании окна

hb1 = CreateWindow (TEXT ("edit"), NULL, WS\_CHILD | WS\_VISIBLE |

WS\_HSCROLL | WS\_VSCROLL | WS\_BORDER | ES\_LEFT | ES\_MULTILINE |

ES\_AUTOHSCROLL | ES\_AUTOVSCROLL, 0, 0, 0, 0, hWnd, (HMENU)1,

hInstance, NULL);

break;

case WM\_SETFOCUS:

SetFocus(hb1);

break;

case WM\_SIZE:

MoveWindow(hb1,0,0,LOWORD(lParam),HIWORD(lParam),TRUE);

break;

case WM\_PAINT: // Перерисовать окно

hdc = BeginPaint(hWnd, &ps); // Начать графический вывод

EndPaint(hWnd, &ps); // Закончить графический вывод

break;

case WM\_DESTROY: // Завершение работы

PostQuitMessage(0);

break;

default:

return DefWindowProc(hWnd, message, wParam, lParam);

}

return 0;

}