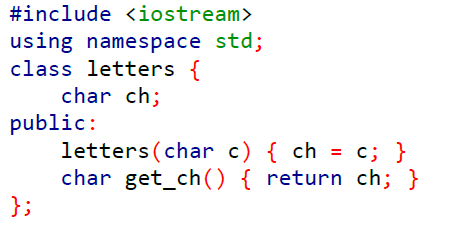
**ПРАКТИЧНЕ ЗАВДАННЯ № 5**

**Завдання №1 :**

Используя следующее объявление класса, создайте массив из 10 элементов и инициализируйте переменную ch значениями от А до J. Покажите, что массив на самом деле содержит эти значения:



**Код програми:**

#include <iostream>

using namespace std;

#define size 10

class letters {

private:

char ch;

public:

letters(char c) {

ch = c;

}

char get\_ch(){

return ch;

}

};

int main() {

letters a[size] = { 'a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j'};

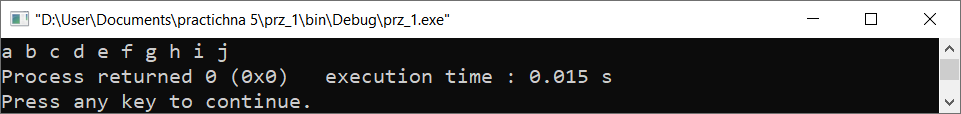
for( int i = 0; i < size; ++i )

cout << a[i].get\_ch() << ' ';

return 0;

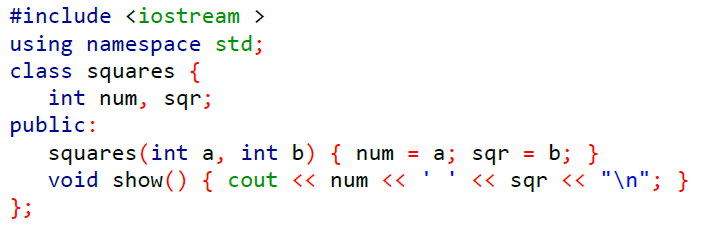
}

**Результати роботи програми (скриншоти)**



**Завдання №2 :**

Используя следующее объявление класса, создайте массив из 10 элементов, инициализируйте переменную num значениями от 1 до 10, а переменную sqr — квадратом num.



**Код програми:**

#include <iostream>

#define size 10

using namespace std;

class squares {

private:

int num, sqr;

public:

squares(int a, int b){ num = a; sqr = b; }

void show() { cout << num << ' ' << sqr << endl;}

};

int main() {

squares a[size] = { squares(1, 1), squares(2, 4),

squares(3, 9), squares(4, 16),

squares(5, 25), squares(6, 36),

squares(7, 49), squares(8, 64),

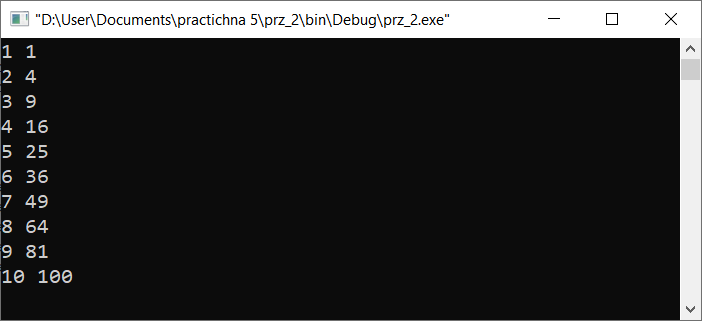
squares(9, 81), squares(10, 100) };

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

a[i].show();

return 0;

}**Результати роботи програми (скриншоти):**



**Завдання №3 :**

Перепишите Example 5.5 (см. Unit 5 "Using Pointers To Objects") так, чтобы на экран выводилось содержимое массива obj в обратном порядке.

**Код програми:**

#include <iostream>

#define size 4

using namespace std;

class samp {

private:

int a, b;

public:

samp(int n, int m) { a = n, b = m; }

int get\_a() { return a; }

int get\_b() { return b; }

};

int main() {

samp obj[size] = { samp(1, 645), samp(6, 6454),

samp(5, 26), samp(7, 8) };

samp \*m = &obj[0];

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

cout << m->get\_a() << ' ';

cout << m->get\_b() << endl;

++m;

}

puts("\n\n");

samp \*p = &obj[size - 1];

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

cout << p->get\_a() << ' ';

cout << p->get\_b() << endl;

--p;

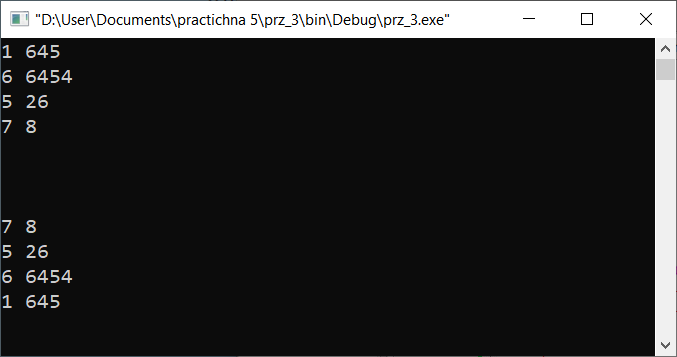
}

cout << endl;

return 0;

}

**Результати роботи програми (скриншоти):**



**Завдання №4 :**

Измените пример Example 5.3 (см. Unit 5 "Arrays of objects") так, чтобы получить доступ к двумерному массиву через указатель. Подсказка: в C++, как и в С, все массивы хранятся непрерывно, слева направо, от младшего элемента к старшему.

**Код програми:**

#include <iostream>

#define size 4

using namespace std;

class samp {

private:

int a, b;

public:

samp(int n) { a = n; }

int get\_a() { return a; }

};

int main() {

samp obj[size][size - 2] = { 1, 2,

3, 4,

5, 6,

7, 8, };

// samp (\*p)[2] = obj;

samp \*ptr = (samp \*) obj;

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

for(int j = 0; j < size - 2; ++j){

cout << ptr->get\_a() << ' ';

++ptr;

}

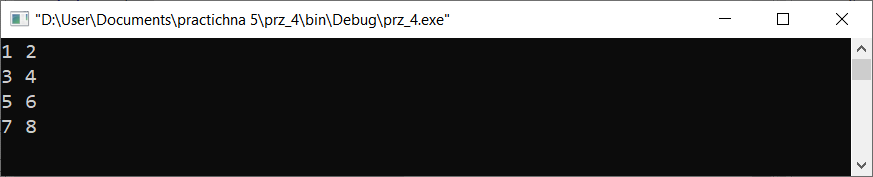
cout << '\n';

}

return 0;

}

**Результати роботи програми (скриншоти):**



**Завдання №5 :**

Напишите программу, в которой оператор new используется для динамического размещения переменных типа float, long и char. Задайте этим динамическим переменным значения и выведите эти значения на экран. В завершение с помощью оператора delete освободите всю динамически выделенную область памяти.

**Код програми:**

#include <iostream>

#include <ctime>

#define SIZE 10

using namespace std;

void generate(long \*a, float \*b, char \*c);

void free\_a(long \*a, float \*b, char \*c);

void show(long \*a, float \*b, char \*c);

int main() {

srand (time(NULL));

long \*a = new long[SIZE];

float \*b = new float[SIZE];

char \*c = new char[SIZE];

generate(a, b, c);

show(a,b,c);

free\_a(a, b, c);

return 0;

}

void free\_a(long \*a, float \*b, char \*c){

delete []a;

delete []b;

delete []c;

}

void generate(long \*a, float \*b, char \*c){

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

a[i] = rand() % 8 \* (rand() % 2 ? 1 : -1);

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

b[i] = rand() % 64 /1.3;

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

c[i] = 'a' + rand() % 26;

}

void show(long \*a, float \*b, char \*c){

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

cout << a[i] << ' ';

cout <<"\n\n";

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

cout << b[i] << ' ';

cout << "\n\n";

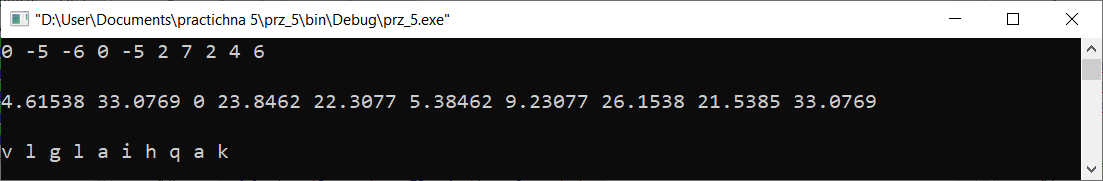
for(int i = 0; i < SIZE; ++i)

cout << c[i] << ' ';

cout <<"\n\n";

}

**Результати роботи програми (скриншоти):**



**Завдання №6 :**

Создайте класс для хранения своего имени и номера телефона. Используя оператор new, динамически выделите память для объекта этого класса и введите имя и телефон в соответствующие поля внутри объекта.

**Код програми:**

#include <iostream>

#include <cstring>

#define SIZE 34

using namespace std;

class mobile {

private:

char name[SIZE];

char number[SIZE];

public:

void save(char \*a, char \*b);

void show();

};

void mobile::save(char \*a, char \*b){

strcpy(name, a);

strcpy(number, b);

}

void mobile::show(){

cout << name<<' '<< number << endl;

}

int main() {

mobile \*ptr = new mobile;

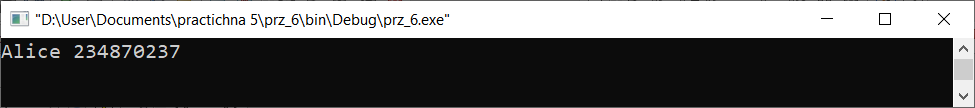
ptr->save((char \*)"Alice",(char \*) "234870237");

ptr->show();

return 0;

}

**Результати роботи програми (скриншоти):**



**Завдання №7 :**

В новую версию программы Example 5.11 (см. Unit 5 "More About new And delete") добавьте деструктор и удостоверьтесь (продемонстрируйте в программе), что при освобождении памяти, обозначенной указателем рtr, для каждого элемента массива (по разу на каждый элемент массива) вызывается деструктор

**Код програми:**

#include <iostream>

using namespace std;

class Products {

private:

int i, j;

public:

~Products() {cout << "del "<< ' ' ;}

void set\_ij(int a, int b){ i = a; j = b; }

int get\_product() { return i + j; }

};

int main() {

Products \*p = new Products[10];

if(!p)

return 1;

for(int i = 0; i < 10; ++i)

p[i].set\_ij(i, i);

for(int i = 0; i < 10; ++i)

cout << p[i].get\_product() << ' ';

puts("\n");

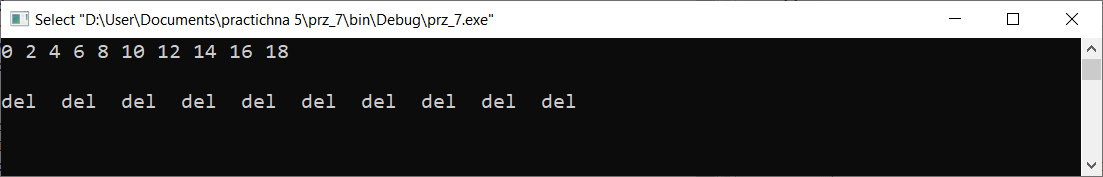
delete []p;

puts("\n");

return 0;

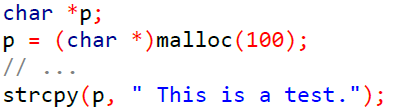
}

**Результати роботи програми (скриншоти):**



**Завдання №8 :**

Переделайте данную программу так, чтобы в ней использовался оператор new:



Подсказка: Строка — это просто массив символов.

**Код програми:**

#include <iostream>

#include <cstring>

using namespace std;

int main(){

char \*p = new char[100];

strcpy(p, "This is a test\n\n");

cout << p;

delete [] p;

return 0;

}

**Результати роботи програми (скриншоти):**



**Завдання №9 :**

Используя оператор new, покажите, как динамически разместить переменную типа double и передать ей начальное значение -123.0987.

**Код програми:**

#include <iostream>

using namespace std;

int main() {

double \*p = new double(-123.87868);

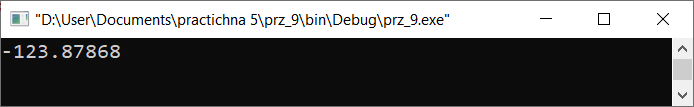
cout.precision(8);

cout << \*p << endl;

return 0;

}

**Результати роботи програми (скриншоти):**



**Завдання №10 :**

Создайте односвязный список (см. Unit 1 "A singly-linked list"), используя new, и реализуйте возможные операции с этим списком, а именно: добавление элемента, удаление элемента, копирование элемента, слияние, разбиение, вставку перед конкретным элементом или после него.

**Код програми:**

#include <string.h>

#include <iostream>

using namespace std;

typedef struct List List;

struct List {

char \*name;

int value;

List \*next;

};

List \*lookup(List \*listp, char \*name){

for ( ; listp != NULL; listp = listp->next)

if (strcmp(name, listp->name) == 0)

return listp;

return NULL;

}

List \*create\_item(char \*name, int value){

List \*newp;

newp = new List;

newp->name = name;

newp->value = value;

newp->next = NULL;

return newp;

}

List \*push\_front(List \*listp, List \*newp){

newp->next = listp;

return newp;

}

List \*push\_end(List \*listp, List \*newp){

List \*p;

if (listp == NULL)

return newp;

for (p = listp; p->next != NULL; p = p->next);

p->next = newp;

return listp;

}

void free\_list(List \*listp){

List \*next;

for ( ; listp != NULL; listp = next) {

next = listp->next;

delete listp;

}

}

List \*del\_item(List \*listp, char \*name){

List \*p, \*prev;

prev = NULL;

for (p = listp; p != NULL; p = p->next) {

if (strcmp(name, p->name) == 0) {

if (prev == NULL)

listp = p->next;

else

prev->next = p->next;

delete p;

return listp;

}

prev = p;

}

return NULL;

}

void print(const List \*listp){

const List \*p;

for (p = listp; p != NULL; p = p->next) {

cout << p->name << ' ' << p->value << " ";

}

cout << endl;

}

List \*create\_item1(char \*name, int value) {

List \*newp;

newp = new List;

newp->name = name;

newp->value = value;

return newp;

}

List \*copy(const List \*listp) {

if(listp == NULL)

return 0;

List \*newp = create\_item1(listp->name, listp->value);

newp->next = copy(listp->next);

return newp;

}

List \*merge(List \*lista, List \*listb){

List \*p;

if (lista == NULL)

return listb;

for (p = lista; p->next != NULL; p = p->next)

continue;

p->next = listb;

return lista;

}

List \*split(List \*listp, char \*name, List \*\*other){

List \*p, \*prev;

prev = NULL;

for (p = listp; p != NULL; p = p->next) {

if (strcmp(p->name, name) == 0) {

\*other = p;

if (prev == NULL)

listp = NULL;

else

prev->next = NULL;

return listp;

}

prev = p;

}

\*other = NULL;

return listp;

}

List \*insert\_before(List \*listp, List \*item, List \*newp){

List \*p, \*prev;

prev = NULL;

for (p = listp; p != NULL; p = p->next) {

if (p == item) {

p = push\_front(item, newp);

if (prev == NULL)

listp = p;

else

prev->next = p;

return listp;

}

prev = p;

}

return NULL;

}

List \*insert\_after(List \*listp, List \*item, List \*newp){

item->next = push\_front(item->next, newp);

return listp;

}

int main(){

List \*listp = NULL, \*listp1 = NULL, \*merged = NULL;

List \*other;

cout << "original\n";

listp = push\_front(listp, create\_item((char \*)"A", 2));

listp = push\_front(listp, create\_item((char \*)"B",8));

listp = push\_front(listp, create\_item((char \*)"C", 5));

listp = push\_front(listp, create\_item((char \*)"D", 4));

print(listp);

cout << "insert\n";

listp = insert\_before(listp, listp, create\_item((char \*)"E", 3));

listp = insert\_before(listp, lookup(listp, (char \*)"B"), create\_item((char \*)"T", 7));

insert\_after(listp, lookup(listp, (char \*)"C"), create\_item((char \*)"G", 6));

del\_item(listp, (char \*)"A");

print(listp);

cout << "copy\n";

listp1 = copy(listp);

print(listp1);

cout << "split\n";

listp = split(listp, (char \*)"C", &other);

print(listp);

cout << "merge\n";

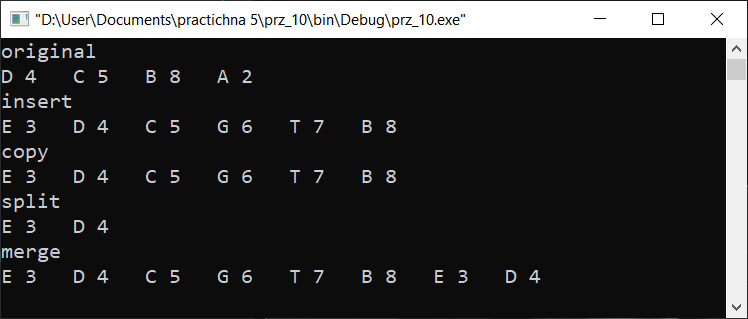
merged = merge(listp1, listp);

print(merged);

return 0;

}

**Результати роботи програми (скриншоти):**



**Завдання №11 :**

Напишите функцию neg(), которая меняет знак своего целого параметра на противоположный. Напишите функцию двумя способами: первый — используя параметр-указатель и второй — параметр-ссылку. Составьте короткую программу для демонстрации обеих функций.

**Код програми:**

#include <iostream>

using namespace std;

int neg(int \*a){

return -\*a;

}

int neg(int &a){

return -a;

}

int main() {

int num;

cin >> num;

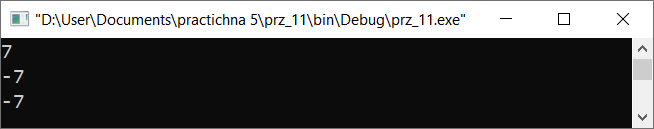
cout << neg(&num) << endl;

cout << neg(num);

return 0;

}

**Результати роботи програми (скриншоти):**



**Завдання №12 :**

Напишите программу, которая создает безопасный (bounded array/safe array) двумерный (2x3) массив целых. Покажите, как она работает.

**Код програми:**

#include <iostream>

#include <cstdlib>

using namespace std;

class Array {

private:

int size1;

int size2;

int \*p;

public:

Array(int i, int j);

~Array(){ delete [] p;}

int &put(int i, int j);

int get(int i, int j);

};

Array::Array(int i, int j){

p = new int[i \* j];

if(!p)

exit(1);

size1 = i;

size2 = j;

}

int &Array::put(int i, int j){

if(i < 0 || j < 0 || i >= size1 || j >= size2)

exit(1);

return p[i \* size2 + j];

}

int Array::get(int i, int j){

if(i < 0 || j < 0 || i >= size1 || j >= size2)

exit(1);

return p[i \* size2 + j];

}

int main() {

int size\_i, size\_j;

cin >> size\_i >> size\_j;

Array a(size\_i, size\_j);

for(int i = 0; i < size\_i; ++i){

for(int j = 0; j < size\_j; ++j){

//a.put(i , j) = i \* j;

a.put(i , j) = rand() % 8;

}

}

for(int i = 0; i < size\_i; ++i){

for(int j = 0; j < size\_j; ++j)

cout << a.get(i , j) << ' ';

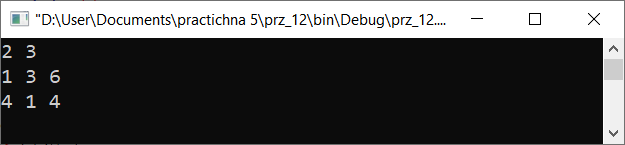
cout << endl;

}

return 0;

}

**Результати роботи програми (скриншоти):**



**Завдання №13 :**

Попытайтесь найти полезное применение для независимой ссылки. Напишите демонстрационную программу.

**Код програми:**

#include <iostream>

using namespace std;

class number {

private:

int x;

public:

number(int n) { n = x; }

void set\_i(int n) { n = x; }

int get(){ return x;}

int &ref = x;

};

void sqr(number m){

m.set\_i(m.get() \* m.get());

}

int main() {

number a(5);

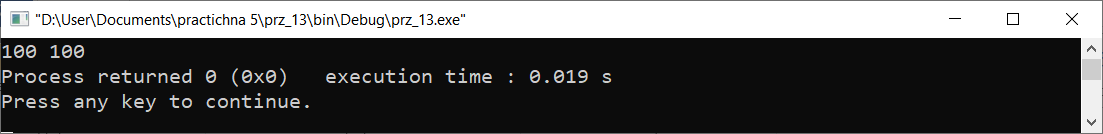
a.ref = 100;

cout << a.ref << ' ' << a.get();

return 0;

}

**Результати роботи програми (скриншоти):**



**Код програми:**

#include <iostream>

using namespace std;

int &max\_el(int \*arr, int size){

int res = arr[0];

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

if(arr[res] < arr[i])

res = i;

}

return arr[res];

}

int main() {

int num;

cin >> num;

int \*arr = new int[num];

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

cin >> arr[i];

max\_el(arr, num) = 0;

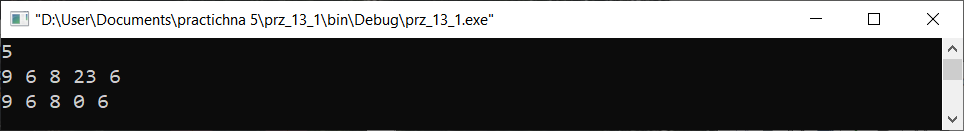
for(int i = 0; i < num; ++i)

cout << arr[i] << ' ';

return 0;

}

**Результати роботи програми (скриншоти):**



**Завдання №14 :**

В Unit2 была создана программа с классом strtype, в которой память для строки выделялась динамически. Переделайте эту программу так, чтобы в ней использовались операторы new и delete.

**Код програми:**

#include <iostream>

#include <cstring>

#include <cstdlib>

using namespace std;

class Strings {

private:

char \*p;

public:

Strings(char const \*ptr);

~Strings();

void show();

};

Strings::Strings(char const \*ptr){

p = new char[strlen(ptr) + 1];

if(!p)

exit(1);

strcpy(p, ptr);

}

Strings::~Strings(){

delete []p;

}

void Strings::show(){

cout << p << endl;

}

int main(){

Strings str1("ABCDEF");

Strings str2("GHJKLM");

str1.show();

str2.show();

return 0;

}

**Результати роботи програми (скриншоти):**

