**ПРАКТИЧНЕ ЗАВДАННЯ № 12**

**Завдання №1 :**

Напишите родовую функцию min(), возвращающую меньший из двух своих аргументов. Например, версия функции min(3, 4) должна возвратить 3, а версия min('c', 'a') – а. Продемонстрируйте работу функции с помощью программы.

**Код програми:**

#include <iostream>

using namespace std;

template <class T> T min(T &a, T &b){

return (a < b) ? a : b;

}

int main() {

cout << "min value: " << min('a', 'z') << endl;

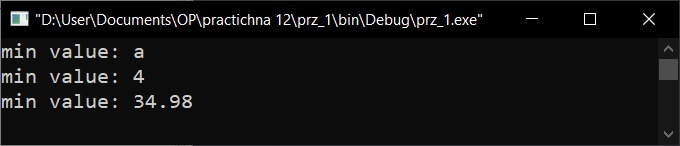
cout << "min value: " << min(9, 4) << endl;

cout << "min value: " << min(34.98, 123.0987) << endl;

return 0;

}

**Результати роботи програми (скриншоти)**



**Завдання №2 :**

Функция find() ищет объект в массиве. Она возвращает либо индекс найденного объекта (если его удалось найти), либо -1, если заданный объект не найден. Ниже представлен прототип конкретной версии функции flnd():

int find(int object, int \*list, int size) {

// Параметр size задает количество элементов массива.

}

Переделайте функцию find() в родовую функцию и продемонстрируйте ваше решение в программе.

**Код програми:**

#include <iostream>

#include <ctime>

using namespace std;

template <class T> int find(T object, T \*list, int size) {

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

if (list[i] == object)

return i;

}

return -1;

}

int main() {

srand(time(NULL));

int arr[10];

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

arr[i] = rand() % 10;

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

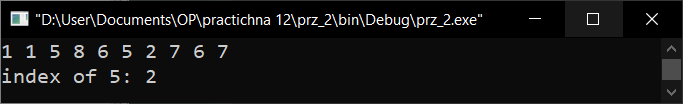
cout << arr[i] << ' ';

cout << endl << "index of 5: " << find(5, arr, 10) << endl;

return 0;

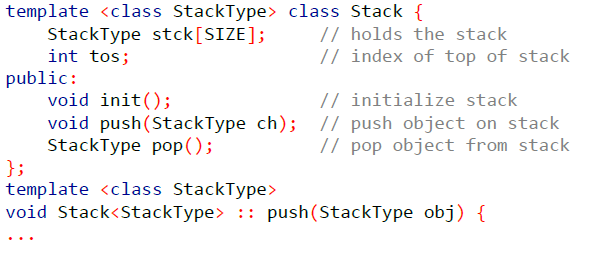
}

**Результати роботи програми (скриншоти)**



**Завдання №3 :**

Создайте и продемонстрируйте родовые классы, реализующие очередь и стек. Подсказка: вот начало родового класса, реализующего стек. Посмотрите Example 2.6 из Unit 2 и закончите написание кода.



**Код програми:**

#include <iostream>

#define SIZE 10

using namespace std;

template <class StackType> class Stack {

StackType stck[SIZE];

int tos;

public:

void init();

void push(StackType a);

StackType pop();

void show();

};

template <class QueueType> class Queue {

QueueType queue[SIZE];

int head, tail;

public:

void init();

void push(QueueType a);

QueueType pop\_q();

void show();

};

template <class StackType> void Stack<StackType>::init() {

tos = 0;

}

template <class StackType> void Stack<StackType>::push(StackType a){

if(tos == SIZE)

return;

stck[tos] = a;

++tos;

}

template <class StackType> StackType Stack<StackType>::pop(){

if(tos ==0)

return 0;

--tos;

return stck[tos];

}

template <class StackType> void Stack<StackType>::show(){

cout << "Stack: ";

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

cout << pop() << ' ';

cout << endl;

}

template <class QueueType> void Queue<QueueType>::init() {

head = tail = 0;

}

template <class QueueType> void Queue<QueueType>::push(QueueType a) {

if(tail + 1 == head || (tail + 1 == SIZE && !head))

return;

++tail;

if(tail == SIZE)

tail = 0;

queue[tail] = a;

}

template <class QueueType> QueueType Queue<QueueType>::pop\_q() {

if(head == tail)

return 0;

++head;

if(head == SIZE)

head = 0;

return queue[head];

}

template <class QueueType> void Queue<QueueType>::show() {

cout << "Queue: ";

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

cout << pop\_q() << ' ';

}

int main() {

Stack<int> a;

Queue<char> b;

a.init();

b.init();

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

a.push(rand() % 20);

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

b.push(rand() % 26 + 'a');

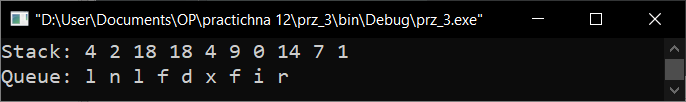
a.show();

b.show();

return 0;

}

**Результати роботи програми (скриншоти)**



**Завдання №4 :**

Создайте родовой класс Input, который при вызове конструктора делает следующее:

• выводит на экран строку-приглашение,

• получает данные от пользователя,

• повторно выводит на экран строку-приглашение, если вводимые данные не соответствуют заданному диапазону. Объекты типа Input должны объявляться следующим образом:

Input obj("prompt message> ", min\_value, max\_value)

Здесь prompt message - это сообщение, появляющееся на экране в качестве приглашения для ввода. Минимальное и максимальное допустимые значения задаются с помощью параметров min\_value и max\_value соответственно. (Тип данных, вводимых пользователем, будет тем же самым, что и тип значений min\_value и max\_value.)

**Код програми:**

#include <iostream>

using namespace std;

template <class T> class Input {

T value;

public:

Input(string str, T min\_value, T max\_value);

void show();

};

template <class T> Input<T>::Input(string str, T min\_value, T max\_value){

do {

cout << str << endl;

cin >> value;

} while(value > max\_value || value < min\_value);

}

template <class T> void Input<T>::show(){

cout << "Entered data: " << value << endl;

}

int main() {

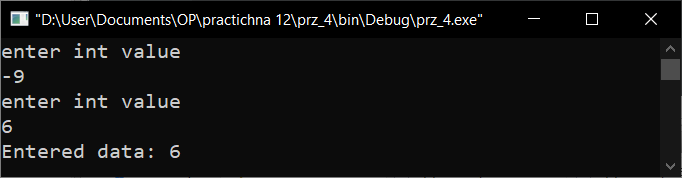
Input<int> a("enter int value", 0, 10);

a.show();

return 0;

}

**Результати роботи програми (скриншоти)**



**Завдання №5 :**

Здесь представлен каркас функции divide():

double divide (double a, double b) {

// add error handling

return a/b;

}

Эта функция возвращает результат деления а на b. Добавьте в функцию код для обработки исключительных ситуаций, а конкретно предусмотрите обработку ошибки деления на ноль. Покажите, что ваша программа работает.

**Код програми:**

#include <iostream>

using namespace std;

double devide(double a, double b){

try {

if(b == 0) throw (b);

} catch (double){

cout << "enter b again: ";

cin >> b;

return devide(a, b);

}

return a/b;

}

int main(){

double a, b;

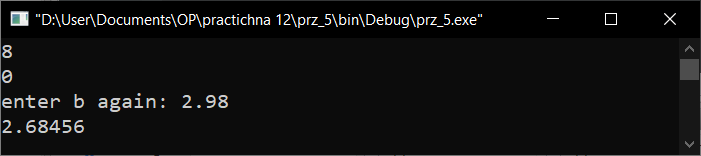
cin >> a >> b;

cout << devide(a, b) << endl;

return 0;

}

**Результати роботи програми (скриншоти)**



**Завдання №6 :**

Создайте родовую функцию, возвращающую значение элемента, который чаще всего встречается в массиве.

**Код програми:**

#include <iostream>

#define SIZE 10

using namespace std;

template <class T> T find(T \*arr, int size = SIZE){

int max = 1; int count = 1; T result = arr[0];

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

int min = i;

for(int j = i + 1; j < size; ++j){

if(arr[j] < arr[min])

min = j;

}

swap(arr[min], arr[i]);

}

for(int i = 1; i < size; ++i){

if(arr[i] == arr[i - 1])

++count;

else {

if(count > max){

max = count;

result = arr[i - 1];

}

count = 1;

}

}

if(count > max){

max = count;

result = arr[size - 1];

}

return result;

}

int main() {

int arr[SIZE]; char str[SIZE];

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

arr[i] = rand() % 4;

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

}

cout << endl << "the most frequent value -- " << find(arr) << endl;

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

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

cout << str[i] << ' ';

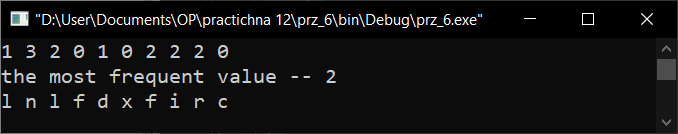
}

cout << endl << "the most frequent value -- " << find(str) << endl;

return 0;

}

**Результати роботи програми (скриншоти)**



**Завдання №7 :**

Создайте родовую функцию, возвращающую сумму значений элементов массива.

**Код програми:**

#include <iostream>

#define SIZE 10

using namespace std;

template <class T> T sum(T \*arr, int size = SIZE) {

T result = 0;

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

result += arr[i];

return result;

}

int main() {

int array[SIZE]; double arr[SIZE]; char str[SIZE];

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

array[i] = rand() % 10;

cout << array[i] << ' ';

}

cout << endl << "sum of elements: " << sum(array) << endl;

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

arr[i] = rand() % 40/10.35 ;

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

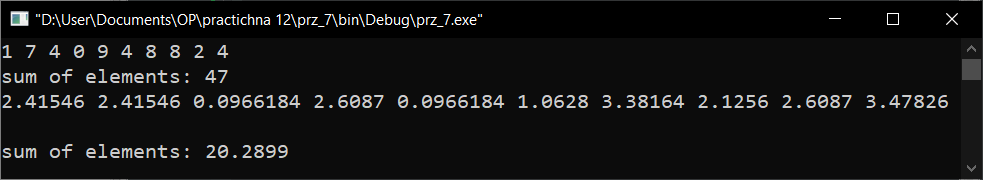
}

cout << endl << "sum of elements: " << sum(arr) << endl;

return 0;

}

**Результати роботи програми (скриншоти)**



**Завдання №8 :**

Создайте родовой класс для "пузырьковой" сортировки (или используйте любой другой известный вам алгоритм сортировки).

**Код програми:**

#include <iostream>

#define SIZE 10

using namespace std;

template <class T> void sort(T \*arr, int size = SIZE) { //selection sort

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

int min = i;

for(int j = i + 1; j < size; ++j) {

if(arr[j] < arr[min])

min = j;

}

swap(arr[min], arr[i]);

}

}

template <class T> void show(T \*arr, int size = SIZE) {

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

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

cout << endl;

}

int main() {

int arr[SIZE]; char str[SIZE];

cout << "Array of int" << endl;

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

arr[i] = rand() % 45;

show(arr);

sort(arr);

show(arr);

cout << endl << "Array of char" << endl;

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

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

show(str);

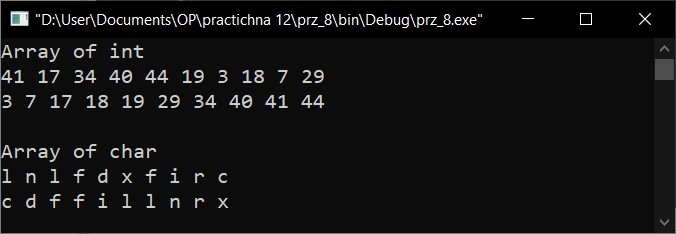
sort(str);

show(str);

return 0;

}

**Результати роботи програми (скриншоти)**



**Завдання №9 :**

Измените класс Stack так, чтобы в стеке можно было хранить пары объектов разных типов.

**Код програми:**

#include <iostream>

#define SIZE 10

using namespace std;

template <class Type1, class Type2> class Stack {

Type1 stack1[SIZE];

Type2 stack2[SIZE];

int tos;

public:

void init();

void push(Type1 a, Type2 b);

void pop();

void show();

};

template <class Type1, class Type2> void Stack<Type1, Type2>::init() {

tos = 0;

}

template <class Type1, class Type2> void Stack<Type1, Type2>::push(Type1 a, Type2 b){

if(tos == SIZE)

return;

stack1[tos] = a;

stack2[tos] = b;

++tos;

}

template <class Type1, class Type2> void Stack<Type1, Type2>::pop(){

if(tos ==0)

return;

--tos;

cout << '{' << stack1[tos] << ' ' << stack2[tos] << "} ";

}

template <class Type1, class Type2> void Stack<Type1, Type2>::show(){

cout << "Stack: ";

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

pop();

cout << endl;

}

int main() {

Stack<int, char> a;

a.init();

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

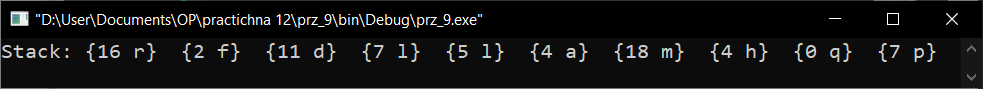
a.push(rand() % 20, rand() % 26 + 'a');

a.show();

return 0;

}

**Результати роботи програми (скриншоти)**



**Завдання №10 :**

Еще раз измените класс Stack так, чтобы переполнение и, наоборот, опустошение стека обрабатывались как исключительные ситуации.

**Код програми:**

#include <iostream>

#define SIZE 10

using namespace std;

template <class StackType> class Stack {

StackType stck[SIZE];

int tos;

public:

void init();

void push(StackType a);

StackType pop();

void show();

};

template <class StackType> void Stack<StackType>::init() {

tos = 0;

}

template <class StackType> void Stack<StackType>::push(StackType a) {

try {

if(tos == SIZE)

throw SIZE;

} catch(int) {

return;

}

stck[tos] = a;

++tos;

}

template <class StackType> StackType Stack<StackType>::pop() {

try {

if(tos == 0)

throw 0;

} catch(int) {

return 0;

}

--tos;

return stck[tos];

}

template <class StackType> void Stack<StackType>::show() {

cout << "Stack: ";

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

cout << pop() << ' ';

cout << endl;

}

int main() {

Stack<int> a;

a.init();

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

a.push(rand() % 20);

a.show();

return 0;

}

**Результати роботи програми (скриншоти)**



**Завдання №11 :**

Просмотрите документацию на ваш компилятор. Проверьте, поддерживает ли он функции terminate() и unexpected(). Как правило, эти функции можно конфигурировать так, чтобы из них вы могли вызвать любую необходимую вам функцию. Если в случае с вашим компилятором это так, постарайтесь создать собственный набор функций завершения программы, который обеспечил бы возможность обработки необрабатываемых до этого исключительных ситуаций.

**Код програми:**

**Результати роботи програми (скриншоти):**

**Завдання №12 :**

Вопрос для размышления: в чем, по вашему мнению, при неудачной попытке выделения памяти преимущество возбуждения исключительной ситуации оператором new по сравнению с возвращением нуля?

**Код програми:**

**Результати роботи програми (скриншоти):**

**Завдання №13 :**

В Unit 2 "Introducing Function Overloading" были созданы перегруженные версии функции abs(). Усовершенствуйте решение, создав родовую функцию abs(), которая возвращала бы абсолютную величину любого численного объекта.

**Код програми:**

#include <iostream>

using namespace std;

template <class T> T abs(T &a) {

return a < 0 ? -a : a;

}

int main() {

cout << abs(-56) << endl;

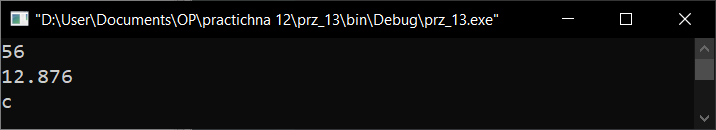
cout << abs(-12.876) << endl;

cout << char(abs(-'c'))<< endl;

return 0;

}

**Результати роботи програми (скриншоти)**



**Завдання №14 :**

В Unit5 "Bounded Array" рассматривался простейший способ реализации безопасного массива, в котором для доступа к элементам массива использовались функции get() и put(). Перегрузка оператора [] позволяет создать такой массив гораздо проще. В практическом задании для Unit7 вам поручалось для создания безопасного массива реализовать в функции operator[]() контроль границ. Кроме этого, функция operator[]() должна возвращать ссылку на индексируемый элемент. В представленном ниже фрагменте сделана попытка добавить контроль границ массива, что позволяет при нарушении границ генерировать соответствующую ошибку.

**Код програми:**

#include <iostream>

using namespace std;

const int SIZE = 10;

template <class T> class Array {

private:

T arr[SIZE];

public:

Array() {

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

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

}

~Array() {

delete[] arr;

}

T &operator[](int i) {

if (i < 0 || i > SIZE - 1) {

cout << "error" << endl;

exit(1);

}

return arr[i];

}

};

int main() {

Array<char> obj1;

Array<int> obj2;

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

cout << char(obj1[i]) << ' ';

cout << endl;

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

cout << obj2[i] << ' ';

cout << endl;

obj1[SIZE + 5] = 10;

return 0;

}

**Результати роботи програми (скриншоти)**

