Министерство Образования Республики Беларусь

Учреждение образования

«БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет Информационных Технологий

Кафедра Информационных систем и технологий

**СТРУКТУРЫ ДАННЫХ**

**Схема Бд. Меню программы**

**Лабораторная работа №3, 4.**

Студент:

1курса 2 группы 1 подгруппы

Марушко Тимофей Фёдорович

Проверяющий:

Белодед Николай Иванович

**ОГЛАВЛЕНИЕ**

[**Теоритические сведения 3**](#_Toc192074976)

[**1. Схема БД (Базы Данных) 3**](#_Toc192074977)

[**2. Меню программы 3**](#_Toc192074978)

[**3. Графическое меню программы 8**](#_Toc192074979)

[**Решение практических задач 11**](#_Toc192074980)

[**Решение Дополнительных задач 13**](#_Toc192074981)

Теоритические сведения

Порой, чтобы продемонстрировать работу консольной программы бывает удобно воспользоваться меню — элементом пользовательского интерфейса, позволяющим выбрать одну из нескольких перечисленных опций программы. Иногда же наличие меню является обязательным условием задания по программированию.

1. Схема БД (Базы Данных)

Базы Данных — это структурное представление данных, которое определяет, как данные будут храниться, организовываться и связываться между собой в базе данных. Схема БД определяет таблицы, поля, связи между таблицами, ограничения и индексы.

Создание схемы БД — это важный этап проектирования базы данных, который позволяет определить основную структуру и связи между объектами данных. При проектировании схемы БД следует учитывать требования к хранению и обработке данных, а также предусмотреть возможность расширения и изменения схемы в будущем.

Но так как изучение баз данных будет подробно изучается как отдельная дисциплина в следующем семестре, то бд будет заменяться файловой системой. Однако составление даже самой простой схемы бд значительно упростит понимание дальнейшей файловой системы (из каких файлов и какие данные получать, записывать и тд). Файлы не будут связаны между собой как в случае с таблицами в базе данных.

1. Меню программы

Меню программы — это пользовательский интерфейс, который позволяет пользователю выбирать определенные функции или операции, которые необходимо выполнить в программе. Меню программы может содержать различные элементы, такие как кнопки, выпадающие списки, чекбоксы и т.д., которые позволяют пользователю выбирать и настраивать определенные параметры программы.

Кроме того, при проектировании меню программы следует учитывать различные возможности для настройки и персонализации пользовательского интерфейса, такие как изменение цветовой схемы, выбор языка и т.д. Это позволяет пользователю настроить программу под свои потребности и предпочтения.

Составление меню программы в C++ включает в себя несколько шагов:

1. Определение опций меню: определите список опций, которые вы хотите включить в меню программы. Каждая опция должна иметь уникальный идентификатор и краткое описание.
2. Отображение меню: напишите код, который отображает меню на экране. Обычно это включает в себя использование цикла для вывода списка опций на экран и ожидание ввода пользователя.
3. Обработка выбора пользователя: когда пользователь выбирает опцию из меню, программа должна выполнить соответствующую функцию или выполнять определенные действия. Напишите код для обработки выбора пользователя и выполнения соответствующих функций.
4. Повторение цикла: после выполнения выбранной опции программа должна вернуться к началу цикла и отобразить меню снова, чтобы пользователь мог выбрать другую опцию или выйти из программы.

Существуют разные способы реализации консольного меню, один из самых простых и популярных использовать **switch-конструкцию.**

*#include* <iostream>

*#include* <windows.h>

using namespace std;

int \_stateMenu;

void Menu() {

cout << "Выберите действие:" << endl

<< "(1) Ввод данных" << endl

<< "(2) Вывод" << endl

<< "(3) Удаление" << endl

<< "Ваш выбор: ";

cin >> \_stateMenu;

}

int main()

{

setlocale(LC\_ALL, "rus");

Menu();

*while* (\_stateMenu != 0) {

*switch* (\_stateMenu)

{

*case* 1:

system("cls"); *// очистка консоли*

cout << "Данные введены!" << endl;

system("pause"); *// задержка консоли*

system("cls");

Menu();

*break*;

*case* 2:

system("cls");

cout << "Данные выведены!" << endl;

system("pause");

system("cls");

Menu();

*break*;

*case* 3:

system("cls");

cout << "Данные удалены!" << endl;

system("pause");

system("cls");

Menu();

*break*;

*default*:

system("cls");

cout << "Неверно введен номер действия!" << endl;

system("pause");

system("cls");

Menu();

*break*;

}

}

}

Есть различные способы реализации меню в C++, помимо использования конструкции **switch-case**. Рассмотрим несколько из них:

1. Использование массива функций: создайте массив указателей на функции, которые будут выполнять определенные действия, и используйте цикл для вывода меню и вызова соответствующей функции при выборе пользователем опции.

Пример:

*#include* <iostream>

using namespace std;

void option1() {

cout << "выбрано 1" << endl;

}

void option2() {

cout << "выбрано 2" << endl;

}

void option3() {

cout << "выбрано 3" << endl;

}

int main() {

setlocale(LC\_ALL, "rus");

void (\*options[3])() = { option1, option2, option3 };

int choice;

*do* {

cout << "Menu:" << endl;

cout << "1. Option 1" << endl;

cout << "2. Option 2" << endl;

cout << "3. Option 3" << endl;

cout << "4. Exit" << endl;

cout << "Ваш выбор: ";

cin >> choice;

*if* (choice >= 1 && choice <= 3) {

(\*options[choice - 1])();

}

*else* *if* (choice == 4) {

cout << "выходим из программы..." << endl;

}

*else* {

cout << "неправильный выбор" << endl;

}

} *while* (choice != 4);

*return* 0;

1. Использование функциональных объектов: вместо массива указателей на функции можно использовать объекты. (способ может показаться тяжёлым, т.к. в нём используются лямбда выражения) Пример:

*#include* <iostream>

*#include* <functional>

using namespace std;

class Option1 {

public:

void operator()() {

cout << "выбрано 1" << endl;

}

};

class Option2 {

public:

void operator()() {

cout << "выбрано 2" << endl;

}

};

class Option3 {

public:

void operator()() {

cout << "выбрано 3" << endl;

}

};

int main() {

setlocale(LC\_ALL, "rus");

Option1 option1;

Option2 option2;

Option3 option3;

std::function<void()> options[3] = { [&]() { option1(); }, [&]() { option2(); }, [&]() { option3(); } };

int choice;

*do* {

cout << "Menu:" << endl;

cout << "1. Option 1" << endl;

cout << "2. Option 2" << endl;

cout << "3. Option 3" << endl;

cout << "4. Exit" << endl;

cout << "Enter your choice: ";

cin >> choice;

*if* (choice >= 1 && choice <= 3) {

options[choice - 1]();

}

*else* *if* (choice == 4) {

cout << "выходим из программы..." << endl;

}

*else* {

cout << "неправильный выбор" << endl;

}

} *while* (choice != 4);

*return* 0;

}

Однако использование конструкции switch-case интуитивно понятно и упрощает чтение кода, поэтому рекомендуем использовать её.

1. Графическое меню программы

Так же есть несколько способов реализации **графического меню** в консоли на C++. Рассмотрим некоторые из них:

1. Использование библиотеки ncurses:

Библиотека ncurses является стандартной библиотекой для создания текстовых пользовательских интерфейсов в UNIX-подобных операционных системах. Она предоставляет набор функций для работы с окнами, кнопками, полями ввода, списками и другими элементами интерфейса. На C++ для работы с ncurses можно использовать, например, библиотеку PDCurses.

1. Использование ANSI Escape Sequences:

ANSI Escape Sequences — это последовательности символов, которые можно использовать для управления цветом текста, перемещения курсора и других действий в консоли. С помощью этих последовательностей можно реализовать простые графические элементы, такие как рамки, кнопки и т.д. Однако, для создания более сложных интерфейсов, потребуется написание большого количества кода.

1. Использование графических библиотек:

Если требуется создать более сложный графический интерфейс, можно использовать графические библиотеки, такие как SDL или OpenGL. Однако, для работы с такими библиотеками потребуется использовать оконный менеджер, такой как X Windows System.

Как мы видим, в основном надо устанавливать дополнительные библиотеки или компиляторы, однако можно использовать ANSI Escape Sequences, которая не является библиотекой, а является последовательностью символов, т.е. не требует дополнительных установок и манипуляций.

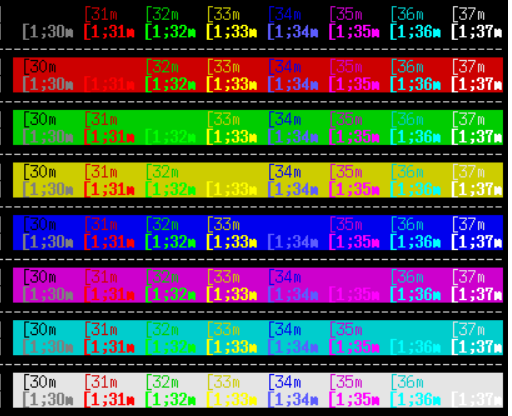
Основные последовательности символов для манипуляций с цветом текста:

* "\033[0m" - сброс стилей и цвета текста;
* "\033[30m" - установка цвета текста на чёрный;
* "\033[31m" - установка цвета текста на красный;
* "\033[32m" - установка цвета текста на зелёный;
* "\033[33m" - установка цвета текста на жёлтый;
* "\033[34m" - установка цвета текста на синий;
* "\033[35m" - установка цвета текста на фиолетовый;
* "\033[36m" - установка цвета текста на голубой;
* "\033[37m" - установка цвета текста на белый;
* "\033[41m" - установка цвета фона на красный;
* "\033[42m" - установка цвета фона на зелёный;
* "\033[43m" - установка цвета фона на жёлтый;
* "\033[44m" - установка цвета фона на синий;
* "\033[45m" - установка цвета фона на фиолетовый;
* "\033[46m" - установка цвета фона на голубой;
* "\033[47m" - установка цвета фона на белый.

Для использования ANSI Escape Sequences в коде на языке C++ можно использовать специальные символы-экранирующие последовательности, такие как \033. Например, чтобы установить цвет текста на красный, можно использовать следующий код:

std::cout << "\033[31m" << "Red text" << "\033[0m" << std::endl;

Этот код устанавливает цвет текста на красный, выводит строку "Red text" и сбрасывает цвет и стили текста до значений по умолчанию.



Теперь добавим ANSI Escape Sequences в наше меню:

*#include* <iostream>

*#include* <windows.h>

using namespace std;

int \_stateMenu;

void Menu() {

system("cls"); *// очистка консоли*

std::cout << "\033[34m+--------------------------+\033[0m" << std::endl;

std::cout << "\033[34m|\033[0m" << " " << "\033[31mЦветное меню\033[0m" << " " << "\033[34m|\033[0m" << std::endl;

std::cout << "\033[34m+--------------------------+\033[0m" << std::endl;

std::cout << "\033[34m|\033[0m" << " " << "\033[32m1.\033[0m" << " Ввод данных " << "\033[34m|\033[0m" << std::endl;

std::cout << "\033[34m|\033[0m" << " " << "\033[32m2.\033[0m" << " Вывод " << "\033[34m|\033[0m" << std::endl;

std::cout << "\033[34m|\033[0m" << " " << "\033[32m3.\033[0m" << " Удаление " << "\033[34m|\033[0m" << std::endl;

std::cout << "\033[34m+--------------------------+\033[0m" << std::endl;

std::cout << "\033[34m|\033[0m" << " Выберите пункт меню:"; cin >> \_stateMenu; cout << "\033[34m|\033[0m" << std::endl;

*/\*cin >> \_stateMenu;\*/*

std::cout << "\033[34m+--------------------------+\033[0m" << std::endl;

}

int main()

{

setlocale(LC\_ALL, "rus");

Menu();

*while* (\_stateMenu != 0) {

*switch* (\_stateMenu)

{

*case* 1:

system("cls"); *// очистка консоли*

cout << "Данные введены!" << endl;

system("pause"); *// задержка консоли*

Menu();

*break*;

*case* 2:

system("cls");

cout << "Данные выведены!" << endl;

system("pause");

Menu();

*break*;

*case* 3:

system("cls");

cout << "Данные удалены!" << endl;

system("pause");

Menu();

*break*;

*default*:

system("cls");

cout << "Неверно введен номер действия!" << endl;

system("pause");

Menu();

*break*;

}

}

*return* 0;

}

Решение практических задач

**Условие:**

Вы работаете в учебном заведении и организуете курсы для улучшения знаний студентов. Вам известна информация *о группах студентов*, которые сформированы в зависимости от их *специальности и отделения*. Каждая группа имеет определенное *количество студентов*, а проведение занятий обеспечивается штатом преподавателей. У каждого преподавателя есть анкетные данные (*фамилия, имя, отчество, телефон*) и *стаж работы*. Вы получаете информацию о том, *сколько часов* каждый преподаватель проводит занятий с соответствующими группами, а также какие виды занятий он проводит (*лекции, практики*), какой *предмет* и сколько он получает *оплату за 1 час*.

**Решение:**

*#include* <iostream>

using namespace std;

*// Функция для отображения меню*

void showMenu() {

cout << "\n--- University Course Management ---\n"

<< "1. Add a new group\n"

<< "2. Show all groups\n"

<< "3. Add a student to a group\n"

<< "4. Remove a student from a group\n"

<< "5. Add a teacher\n"

<< "6. Show all teachers\n"

<< "7. Assign a teacher to a course\n"

<< "8. Show all courses\n"

<< "9. Exit\n"

<< "Choose an option: ";

}

int main() {

int choice;

*while* (true) {

showMenu();

cin >> choice;

*switch* (choice) {

*case* 1:

cout << "Adding a new group...\n";

*// Здесь должен быть ввод данных о группе*

*break*;

*case* 2:

cout << "Showing all groups...\n";

*// Здесь должен быть вывод всех групп*

*break*;

*case* 3:

cout << "Adding a student to a group...\n";

*// Здесь добавление студента в выбранную группу*

*break*;

*case* 4:

cout << "Removing a student from a group...\n";

*// Здесь удаление студента из группы*

*break*;

*case* 5:

cout << "Adding a new teacher...\n";

*// Здесь ввод данных о преподавателе*

*break*;

*case* 6:

cout << "Showing all teachers...\n";

*// Здесь должен быть вывод всех преподавателей*

*break*;

*case* 7:

cout << "Assigning a teacher to a course...\n";

*// Здесь назначение преподавателя на курс*

*break*;

*case* 8:

cout << "Showing all courses...\n";

*// Здесь должен быть вывод всех курсов*

*break*;

*case* 9:

cout << "Exiting program...\n";

*return* 0;

*default*:

cout << "Invalid choice! Please try again.\n";

*break*;

}

}

*return* 0;

}

Решение Дополнительных задач

**Условие:**

Напишите программу, которая будет находить наибольший общий делитель двух чисел. (3 балла)

**Решение:**

*#include* <iostream>

using namespace std;

*// Функция для вычисления НОД (алгоритм Евклида)*

int gcd(int *a*, int *b*) {

*while* (*b* != 0) {

int temp = *b*;

*b* = *a* % *b*;

*a* = temp;

}

*return* *a*;

}

int main() {

int a, b;

cout << "Enter two numbers: ";

cin >> a >> b;

cout << "GCD of " << a << " and " << b << " is " << gcd(a, b) << endl;

*return* 0;

}

**Условие:**

Напишите программу, которая будет находить медиану вектора чисел. (3 балла)

**Решение:**

*#include* <iostream>

*#include* <vector>

using namespace std;

*// Функция для нахождения медианы*

double findMedian(vector<int>& *nums*) {

int n = *nums*.size();

*if* (n % 2 == 1) {

*return* *nums*[n / 2]; *// Нечётное количество элементов*

}

*else* {

*return* (*nums*[n / 2 - 1] + *nums*[n / 2]) / 2.0; *// Чётное количество элементов*

}

}

int main() {

vector<int> numbers;

int num;

cout << "Enter numbers (type -1 to finish): ";

*while* (cin >> num && num != -1) {

numbers.push\_back(num);

}

*if* (numbers.empty()) {

cout << "No numbers entered!" << endl;

}

*else* {

cout << "Median: " << findMedian(numbers) << endl;

}

*return* 0;

}

**Условие:**

Напишите программу, которая будет находить количество путей в ориентированном графе от одной вершины до другой. (3 балла)

**Решение:**

*#include* <iostream>

*#include* <map>

*#include* <vector>

using namespace std;

*// Ориентированный граф (заданный вручную)*

map<int, vector<int>> graph = {

{1, {2, 4}}, *// 1 → 2, 1 → 4*

{2, {3}}, *// 2 → 3*

{4, {5}}, *// 4 → 5*

{5, {3}}, *// 5 → 3*

{3, {}} *// 3 → (нет исходящих рёбер)*

};

*// Функция для поиска пути с использованием DFS*

bool findPath(int *current*, int *end*, vector<int>& *path*, vector<bool>& *visited*) {

*path*.push\_back(*current*);

*if* (*current* == *end*) *return* true; *// Если дошли до цели, путь найден*

*visited*[*current*] = true;

*for* (int neighbor : graph[*current*]) {

*if* (!*visited*[neighbor]) {

*if* (findPath(neighbor, *end*, *path*, *visited*)) *return* true;

}

}

*path*.pop\_back(); *// Откатываемся назад, если путь не найден*

*return* false;

}

int main() {

int start = 1, end = 3; *// Задаём стартовую и конечную вершины*

vector<int> path;

vector<bool> visited(6, false); *// Массив посещённых вершин (макс. вершина = 5)*

*if* (findPath(start, end, path, visited)) {

cout << "Path found: ";

*for* (int node : path) cout << node << " ";

cout << endl;

}

*else* {

cout << "No path exists from " << start << " to " << end << endl;

}

*return* 0;

}