Министерство образования Республики Беларусь

Учреждение образования

“Белорусский Государственный Технологический Университет”

Отчёт по лабораторной работе №6

Минск 2024

Выполнил студент 1-го курса , факультета ФИТ, специальности «Программная инженерия», 10 группы: Езерский Никита Витальевич

#include <windows.h>

#include <iostream>

using namespace std;

void main() {

SetConsoleOutputCP(1251);

SetConsoleCP(1251);

char upper\_case, lower\_case, number;

int n;

cout << "Введите число от 1 до 4: ";

cin >> n;

switch (n) {

case 1: {

cout << "Ввод продолжится до тех пор пока вы не введёте точку(.)\n";

while (true) {

cout << "Прописное написание: ";

cin >> upper\_case;

if (upper\_case == '.') break;

cout << "Строчное написание: ";

cin >> lower\_case;

if (lower\_case == '.') break;

if ((upper\_case >= 0x41 && upper\_case <= 0x5A) && (lower\_case >= 0x61 && lower\_case <= 0x7A)) {

cout << "Разница значений составляет: " << (int)upper\_case - (int)lower\_case << endl;

}

else {

cout << "Ошибка: неверный ввод.\n";

}

}

break;

}

case 2: {

cout << "Ввод продолжится до тех пор пока вы не введёте точку(.)\n";

while (true) {

cout << "Прописное написание: ";

cin >> upper\_case;

if (upper\_case == '.') break;

cout << "Строчное написание: ";

cin >> lower\_case;

if (lower\_case == '.') break;

if ((unsigned char)upper\_case >= 0xC0 && (unsigned char)upper\_case <= 0xDF &&

(unsigned char)lower\_case >= 0xE0 && (unsigned char)lower\_case <= 0xFF) {

cout << "Разница значений составляет: " << (int)upper\_case - (int)lower\_case << endl;

}

else {

cout << "Ошибка: неверный ввод.\n";

}

}

break;

}

case 3: {

cout << "Ввод продолжится до тех пор пока вы не введёте точку(.)\n";

while (true) {

cout << "Число: ";

cin >> number;

if (number == '.') break;

if (number >= '0' && number <= '9') {

cout << "Код символа: " << (int)number << endl;

}

else {

cout << "Ошибка: неверный ввод.\n";

}

}

break;

}

case 4: {

cout << "Выход из программы.\n";

break;

}

default:

cout << "Ошибка: неверный выбор.\n";

break;

}

}

**Задание 1**

**Входные данные:** ввод числа от 1 до 4; ввод символа в прописном(строчном) исполнении ; ввод числа ; ввод точки(.).

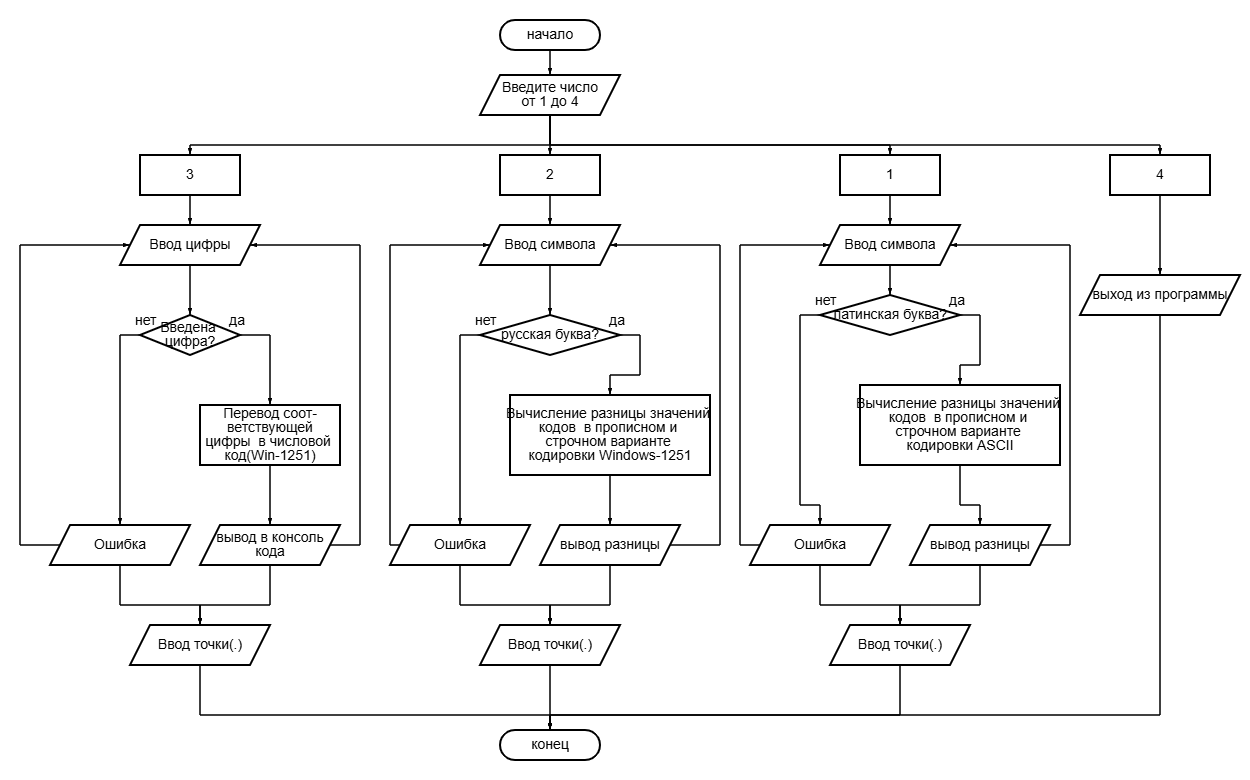
**Выходные данные:**

– определение разницы значений кодов в ASCII буквы в прописном и строчном написании, если введен символ латинского алфавита, иначе вывод сообщения об ошибке;

– определение разницы значений кодов в Windows-1251 буквы в прописном и строчном написании, если введен символ русского алфавита, иначе вывод сообщения об ошибке;

– вывод в консоль кода символа, соответствующего введенной цифре, иначе вывод сообщения об ошибке;

– выход из программы.



**Разбить программу на модули:**

МОДУЛЬ 1(начальный модуль) отвечающий за выбранную пользователем цифру от 1 до 4;

МОДУЛЬ 2 отвечает за выбранную цифру 1;

МОДУЛЬ 3 отвечает за выбранную цифру 2;

МОДУЛЬ 4 отвечает за выбранную цифру 3;

МОДУЛЬ 5 отвечает за выбранную цифру 4;

M1

M4

M3

M2

M5

**ПСЕВДОКОД:**

НАЧАЛО

МОДУЛЬ 1:

ВВОД пользователем цифры от 1 до 4 , ИНАЧЕ “Ошибка: неверный выбор” (КОНЕЦ)

МОДУЛЬ 2:

НАЧАЛО ЦИКЛА

ВВОД символа в прописном , а также строчном исполнении , ИНАЧЕ "Ошибка: неверный ввод."

УСЛОВИЕ “латинская буква?” ЕСЛИ да , то переходим к следующему пункту , ИНАЧЕ “ Ошибка”

ВЫВОД результата

ВВОД (.)

КОНЕЦ ЦИКЛА

МОДУЛЬ 3:

НАЧАЛО ЦИКЛА

ВВОД символа в прописном , а также строчном исполнении , ИНАЧЕ "Ошибка: неверный ввод."

УСЛОВИЕ “русская буква?” ЕСЛИ да , то переходим к следующему пункту , ИНАЧЕ “ Ошибка”

ВЫВОД результата

ВВОД (.)

КОНЕЦ ЦИКЛА

МОДУЛЬ 4:

НАЧАЛО ЦИКЛА

ВВОД цифра , ИНАЧЕ "Ошибка: неверный ввод."

ВЫВОД результата

ВВОД (.)

КОНЕЦ ЦИКЛА

МОДУЛЬ 5:

КОНЕЦ

1. **Парадигма программирования-**подход к написанию программного кода, который определяет, как программист должен решать задачи и структурировать программу.

Императивное **программирование(последовательность)**

Декларативное **программирование(**что нужно сделать, а не как это сделать**)**

Структурное **программирование(необходима структура)**

1. — Методология разработки программного обеспечения совокупность методов, применяемых на различных стадиях жизненного цикла программного обеспечения
2. Императивное **программирование(п**рограмма явно описывает алгоритм решения конкретной задачи**)**

Декларативное **программирование(**предварительная реализация «решателя» для целого класса задач**)**

1. **Цель структурного программирования** — повысить надежность программ (улучшить структуру программы); − создание понятной, читаемой программы, которая исполняется, тестируется, конфигурируется, сопровождается и модифицируется без участия автора.

Основные принципы:

следование; − ветвление; − цикл.

1. программирование «сверху-вниз» (нисходящее программирование); − модульное программирование с иерархическим упорядочением связей между модулями/подпрограммами «От общего к частному»

1. **нисходящее** проектирование (формализация алгоритма «сверху вниз»: движение от общего к частному); − **пошаговое** проектирование (нисходящая пошаговая детализация программы); − **структурное** проектирование (замена формулировки алгоритма на одну из синтаксических конструкций – **последовательность, условие или цикл**; программирование без goto); − одновременное проектирование алгоритма и данных (процессы детализации алгоритма и введение данных, необходимых для работы, идут параллельно); − **модульное проектирование**; − модульное, нисходящее, пошаговое тестирование.
2. Модульное программирование – это организация программы как совокупности небольших независимых блоков, называемых модулями.
3. Функциональная декомпозиция задачи – разбиение большой задачи на ряд более мелких, функционально самостоятельных подзадач – модулей.
4. **один вход и один выход** (на вход программный модуль получает набор исходных данных, выполняет их обработку и возвращает набор выходных данных); − **функциональная завершенность** (модуль выполняет набор определенных операций для реализации каждой отдельной функции, достаточных для завершения начатой обработки данных); − **логическая независимость** (результат работы данного фрагмента программы не зависит от работы других модулей); − **слабые информационные связи с другими программными модулями** (обмен информацией между отдельными модулями должен быть минимален); − **размер и сложность программного элемента должна быть в разумных рамках**.
5. Методы модульного программирования:

**Модульная декомпозиция**

**Повторное использование**

**Иерархическая структура**

1. Стандарт оформления кода – набор правил и соглашений, используемых при написании исходного кода на некотором языке программирования.
2. Следует делать как можно меньше комментариев, делая код самодокументируемым путем выбора правильных имен и создания ясной логической структуры.
3. Имена переменных (включая параметры функций) пишутся **строчными буквами**, **возможно с подчёркиванием** между словами (в одном стиле)
4. Имена пользовательских типов начинаются с **прописной буквы**, **каждое новое слово** также начинается с прописной буквы. **Подчёркивания не используются**.
5. Названия методов и функций **должны быть глаголами**, быть записанными **в смешанном регистре**, **начинаться с прописной буквы** (в нижнем регистре) и каждое слово в имени пишется с прописной буквы
6. Объекты объявляются как const или constexpr, чтобы значение не менялось в процессе выполнения. Имена констант константы пишутся в **верхнем регистре. Подчёркивание** может быть использовано **в качестве разделителя**
7. Названия методов и функций должны быть глаголами, быть записанными в **смешанном регистре**, начинаться с **прописной буквы** (в нижнем регистре) и каждое слово в имени пишется с прописной буквы
8. Хорошо спроектированная функция имеет следующие характеристики: • полностью выполняет четко поставленную задачу; • не берет на себя слишком много работы; • не связана с другими функциями бесцельно; • хранит данные максимально сжато; • помогает распознать и разделить структуру программы; • помогает избавиться от лишней работы, которая иначе присутствовала бы в программе.