**МИНИСТЕРСТВО ЦИФРОВОГО РАЗВИТИЯ, СВЯЗИ И МАССОВЫХ КОММУНИКАЦИЙ РОССИЙСКОЙ ФЕДЕРАЦИИ**

Ордена Трудового Красного Знамени федеральное государственное бюджетное образовательное учреждение высшего образования

**«Московский технический университет связи и информатики»**

Кафедра «Информационная безопасность»

ПРАКТИКУМ

по дисциплине

**РАЗРАБОТКА БЕЗОПОСНОГО ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ**

**Программирование на языках C/С++**

(для студентов направлений подготовки 10.03.01)

Москва 2023

План УМД на 20\_\_/\_\_ уч. г.

Практикум

по дисциплине

РАЗРАБОТКА БЕЗОПАСНОГО ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

Программирование на языках C/С++

(для студентов направлений подготовки 10.03.01)

Составители: Симонян А.Г., к.т.н., доцент (МТУСИ)

Барков В.В., старший преподаватель (МТУСИ)

Рецензент: Шелухин О.И., д.т.н., профессор (МТУСИ)

Издание утверждено на заседании совета факультета ИТ

Протокол № \_ от \_\_\_\_\_\_\_\_\_ г.

Рекомендовано к изданию кафедрой «ИБ»

Протокол № \_ от \_\_\_\_\_\_\_\_\_ г.

## Практикум №1 Создание однофайловых и многофайловых проектов. Объявление и определение переменных и функций.

### Цель практикума

Овладеть навыками создания однофайловых и многофайловых проектов на языке C++ (ISO/IEC 14882:2020). Научиться объявлять и определять переменные и функции в одной и нескольких единице трансляции. Познакомиться со способами передачи данных между функциями. Научиться определять и вызывать перегруженные функции и функции с параметрами по умолчанию.

### Краткие теоретические сведения

#### Основные понятия

Программа на языке C++ состоит из одного или нескольких файлов исходного кода обычно с расширением \*.cpp. Общие для группы файлов исходного кода объявления функций, глобальных переменных, структур, объединений и перечислений могут быть вынесены в отдельный файл, называемый заголовочным файлом. Такие файлы обычно имеют расширение \*.h. Файл исходного кода со всеми включёнными в него заголовочными файлами называется единицей трансляции.

Каждый файл исходного кода независимо от других файлов преобразуется компилятором в файл с объектным кодом \*.obj. Далее множество объектных файлов связываются компоновщиком в исполняемый файл \*.exe или динамическую библиотеку компоновки \*.dll.

Множество объектных файлов также может быть упаковано специальной программой-архиватором в статическую библиотеку. Статические библиотеки вместе с заголовочными файлами могут поставляться другим разработчикам. Такие библиотеки подаются на вход компоновщику вместе с другими объектными файлами.

Объектные модули, исполняемые файлы и динамические библиотеки обычно содержат сегмент глобальных инициализированных данных, размер сегмента глобальных неинициализированных данных и сегмент кода.

Время жизни динамически выделенных объектов определяется вызовами функций выделения и освобождения памяти. Такие объекты хранятся в области динамической памяти.

Сущность (entity) – это значение (value), объект (object), ссылка (reference), структурированная привязка (structured binding), функция (function), перечислитель (enumerator), тип (type), член класса (class member), битовое поле (bit-field), шаблон (template), специализация шаблона (template specialization), пространство имен (namespace) или пакет параметров (pack).

Имя (name) – это такое использование идентификатора, идентификатора функции-операции, идентификатора литеральной операции, идентификатора функции преобразования, или идентификатора шаблона, при котором они обозначают сущность (entity) или метку (label).

Каждое имя, обозначающее сущность вводится при помощи объявления (declaration).

Каждое имя, обозначающее метку, вводится либо при использовании инструкции goto, либо инструкции с меткой.

Переменная вводится путем объявления ссылки (за исключением объявления нестатических данных-членов) или объекта. Имя переменной обозначает ссылку или объект.

Локальная сущность – переменная с автоматической продолжительностью хранения, структурированная привязка, если соответствующая переменная является локальной сущностью, или объект \*this.

Некоторые имена могут обозначать типы или шаблоны. В общем случае перед тем, как продолжать анализ программы, содержащей некоторое имя, необходимо определить, обозначает ли это имя одну из перечисленных сущностей. Такой процесс определения называется поиском имени (name lookup).

Имя, используемое более чем в одной единице трансляции потенциально может ссылаться в этих единицах трансляции на одну и ту же сущность. Это зависит от связывания этого имени, заданного в каждой единице трансляции.

#### Типы данных

Каждый идентификатор имеет тип данных, который задаётся при его объявлении.

Язык C++ поддерживает типы данных: signed char, signed short int, signed int, signed long int, signed long long int, unsigned char, unsigned short int, unsigned int, unsigned long int, unsigned long long int, bool, float, double, long double, char, char16\_t, char32\_t.

Тип данных переменных описывает размер, формат данных и допустимые операции.

Тип данных функции описывает количество и типы аргументов и тип возвращаемого значения.

#### Объявления и определения идентификторов

Перед использованием идентификатор должен быть объявлен.

Объявления лишь указывают компилятору, какой тип имеет переменная или функция.

Объявление (declaration) может вводить одно или несколько имен в единицу трансляции. Объявление может повторно объявлять (redeclare) имена, введенные предыдущими объявлениями. В таком случае оно задает интерпретацию и атрибуты этих имен.

Для того чтобы компилятор разместил код функции в сегменте кода или разместил глобальную переменную в сегменте данных необходимо их определить.

В большинстве случаев объявление также является определением (definition).

#### Область видимости

Каждое имя вводится в некоторой части текста программы, называемом областью объявлений (declarative region). Область объявлений представляет собой наибольшую часть текста программы, в которой вводимое имя имеет смысл (is valid), то есть может использоваться как неквалифицированное имя для ссылки на одну и ту же сущность.

В общем, вводимое имя имеет смысл только в некоторых частях текста программы, которые могут быть несмежными. Такие части называются областью видимости. Для того чтобы определить область видимости определения, удобно ввести понятие потенциальной области видимости. Область видимости определения совпадает с потенциальной областью определения, если потенциальная область видимости не содержит другого определения с тем же именем. Если это условие не выполняется, потенциальная область видимости внутреннего объявления исключается из потенциальной области видимости внешнего объявления и получается область видимости внешнего объявления.

|  |  |  |  |
| --- | --- | --- | --- |
| Переменная | Область объявления | Потенциальная область видимости | Область видимости |
| Первая переменная j | int j = 24;  int main() {  int i = j, j;  j = 42;  } | int j = 24;  int main() {  int i = j, j;  j = 42;  } | int j = 24;  int main() {  int i = j, j;  j = 42;  } |
| Вторая переменная j | int j = 24;  int main() {  int i = j, j;  j = 42;  } | int j = 24;  int main() {  int i = j, j;  j = 42;  } | int j = 24;  int main() {  int i = j, j;  j = 42;  } |

В языке программирования C++ 7 областей видимости:

1. Область видимости блока
2. Область видимости параметров функции
3. Область видимости функции
4. Область видимости пространства имен
5. Область видимости класса
6. Область видимости перечисления
7. Область видимости параметров шаблона

#### Связывание

Программа состоит из одной или более единиц трансляции, объединённых вместе. Единица трансляции состоит из поледовательности объявлений.

Говорят, что имя имеет связывание, когда оно (имя) может означать тот же самый объект, ссылку, функцию, шаблон, пространство имён или значение, что и имя, введённое при помощи объявления в другой области видимости.

Если имя имеет внешнее связывание, то на сущность, которую оно обозначает, можно ссылаться по именам из областей видимости других единиц трансляции или из других областей видимости той же самой единицы трансляции.

Если имя имеет связывание модуля, то на сущность, которую оно обозначает, можно ссылаться по именам из областей видимоти той же самой единицы трансляции модуля (module unit) или из областей видимости других единиц трансляции модуля (того же самого модуля).

Если имя имеет внутреннее связывание, то на сущность, которую оно обозначает, можно ссылаться по именам из других областей видимости в той же единице трансляции.

Если имя не имеет связывания, на сущность, которое оно обозначает, нельзя ссылаться по именам из других областей видимости.

Имя с областью видимости пространства имет имеет внутренне связывание, кроме случаев, когда:

1. это имя переменной, шаблона переменной, функции, шаблона функции, объявленные явно с ключевым словом static.
2. это имя нешаблонной переменной, определённой без квалификатора volatile, с квалификатором const, если:
   1. явно опущено ключевое слово extern
   2. является встроенной (inline) или экспортируемой (export)
   3. объявлена ранее с внутреннем связыанием
3. это переменная-член анониного объединения

Безымянное пространство имён или пространство имён, объявленное прямо или транзитивно внутри безымянного пространства имён имеет внутреннее связывание. Все остальные простанства имён имеют внешнее связывание.

Связывание для имени, имеющего область видимости простанства имён, которому не было установлено внутреннее связывание, обозначающее переменную, функццию, шаблон определяется следующим образом:

1. Если включающее простанство имён имеет внутреннее связывание, то устанавливается внутреннее связвание.
2. Если объвление имени присоединено к именнованному модулю и не экспортировано, имя имеет связывание модуля.
3. В противном случае имя имеет внешнее связывание

#### Примеры объявления и определения идентификаторов

Ниже представлены примеры объявления переменной и функций.

|  |  |
| --- | --- |
| Объявление глобальной переменной | Объявление функции: |
| extern short y; | double sin(double x);  void execute(void); |

Ниже представлены примеры определения глобальных переменных и функций:

|  |  |
| --- | --- |
| Определения глобальных переменных | Определение функции |
| int x1;  int x2 = 5;  static int x3 = 10;  static int x4;  extern int x5 = 15; | void execute(void)  {  } |

Для сборки исполняемого модуля в одном из файлов объектного кода должно быть определение функции с одним из следующих типов:

void main(void);

int main(void);

void main(int argc, char \*\*argv);

int main(int argc, char \*\*argv);

#### Указатели

Переменные в языке C++ могут хранить не только значения базовых типов данных. Переменные могут хранить адреса памяти, назначенные другим переменным. Такие переменные имеют производный тип данных, называемый указателем.

Для объявления переменной-указателя необходимо добавить символ \* к идентификатору. Для того чтобы запретить изменение указателя необходимо объявить его константным. Для того чтобы запретить изменение значения, хранящееся по указателю, необходимо объявить указатель на константный объект.

|  |  |  |
| --- | --- | --- |
| Определение указателя | Определение константного указателя | Определение указателя на константный объект |
| int \*x, \*y;  double \*z; | int \* const x, \* const y;  double \* const z; | const int \*x, \*y;  const double \*z; |

Для получения адреса существующей переменной необходимо воспользоваться операцией взятия адреса (&). Для того чтобы получить значение переменной, хранящейся по адресу, записанному в переменной-указателе, необходимо воспользоваться операцией взятия объекта (\*) (разыменовывание указателя)

|  |  |  |
| --- | --- | --- |
| Определение константного указателя на константный объект | Операция взятия адреса | Разыменовывание указателя |
| const int \* const x, \* const y;  const double \* const z; | int x = 5;  int \*y = &x; | int x = 1, \*y = &x;  \*y = 5; |

Над указателями определены операции сложения, вычитания, инкремента и декремента.

*Операция инкремента* увеличивает адрес, хранимый в указателе таким образом, чтобы он указывал на элемент, следующий за текущим. *Операция декремента* уменьшает адрес, хранимый в указателе таким образом, чтобы он указывал на предыдущий элемент. *Операция сложения* изменяет адрес, хранимый в указателе таким образом, чтобы он указывал на n-ый элемент после текущего. *Операция вычитания* изменяет адрес, хранимый в указателе таким образом, чтобы он указывал на n-ый элемент до текущего. Вычитание указателей показывает, как далеко в памяти находятся два элемента. Если разность равна 1, то элементы являются соседними. Если разность равна 0, то указатели указывают на один и тот же элемент памяти.

Значением указателя может быть любое число. Однако операционные системы накладывают ограничения на адреса памяти, к которым можно обращаться через указатель. Нулевой адрес обычно обозначает, что указатель не указывает ни на какие данные. Обычно первая страница памяти резервируется операционной системой, поэтому обращение к адресам памяти в окресности 0 приводит к ошибке доступа.

#### Передача указателей в функцию

Указатели передаются в функцию также, как и обычные переменные. Значение указателя копируется в функцию. Значение, на которое указывает указатель не копируется. Поэтому изменение указателя не имеет эффекта на вызывающую функцию, а изменение значения, на которое указывает указатель, влияет на вызывающую функцию. Передача параметра по указателю может быть использована для возвращения из функции более одндого параметра, а также для уменьшения времени копирования больших объектов.

#### Возврат указателей из функции

Функции могут возвращать указатели. При этом на возвращаемое значение накладываются ограничения: нельзя возвращать адрес локальной переменной или параметра функции, так как после выхода из функции эта память будет освобождена

#### Ссылки

В языке C++ существуют типы-ссылки: lvalue и rvalue. Ссылки подобны указателям, однако они не могут работать с динамической памятью и не могут быть изменены после инициализации. Можно считать, что ссылки – это псевдонимы. Ссылки lvalue ссылаются на объекты, а rvalue обычно на временные объекты. Ссылки можно использовать при передаче параметров функции. В таком случае функция сможет изменить значения передаваемых переменных. Ниже представлен пример определения и использования ссылок.

|  |  |
| --- | --- |
| Определение ссылок | Использование ссылок |
| int x = 5;  int &a = x; //lvalue ссылка  int &&b = x + 5; //rvalue ссылка | void swap(int &x, int &y)  {  int temp = x;  x = y;  y = temp;  } |

Однако в таком случае, передать временные объекты в функцию не получится. Для того чтобы это исправить, необходимо сообщить компилятору, что мы не будем изменять передаваемые параметры с использованием ключевого слова const либо передавать параметры с использованием rvalue-ссылок

|  |  |
| --- | --- |
| Передача параметров по ссылке на константный объект | Передача параметров по rvalue-ссылке |
| void f(const int &a, const int &b) { } | void f(int &&a, int &&b) { } |

#### Перегрузка функций

В языке C++ в отличие от C разрешено давать функциям одинаковые имена, если они имеют разные типы аргументов. Такое определение функций называется перегрузкой функции:

void f(void);

void f(int);

void f(double);

void f(double, int);

### Задание

В ходе работы необходимо разработать функции, описание которых представлены в таблицах 1-4, а также функцию main.

Консольный вывод осуществлять с помощью глобального объекта cout типа ostream, объявленных в пространстве имен std.

Консольный ввод осуществлять с помощью глобального объекта cin типа istream, объявленных в пространстве имен std.

Объявления типов std::ostream и std::istream, а также объявления глобальных объетков std::cout и std::cin расположены в заголовочном файле iostream.

Работу со строками осуществлять с помощью типа string, объявленного в пространстве имен std.

Преобразование простых типов в строке при необходимости осуществлять с помощью функции to\_string, объявленной в пространстве имен std.

Объявления типа std::string и перегруженной фукнции std::to\_string расположены в заголовочном файле string.

#### Задание 1

Все разработанные функции необходимо записать в файл main.cpp в следующем порядке:

1. ReadPersonAge
2. ReadPersonName
3. ReadPersonHeight
4. ReadPersonWeight
5. ReadPersonSalary
6. ReadPersonData (вариант 1)
7. ReadPersonData (вариант 2)
8. WritePersonData
9. Main

#### Задание 2

Каждую функцию разместить в отдельном файле. Глобальные переменные поместить вместе с определением той функции, для которой они предназначены.

Таблица 1 – Описание функций ReadPersonAge и ReadPersonName

|  |  |  |
| --- | --- | --- |
|  | **ReadPersonAge** | **ReadPersonName** |
| **Назначение функции** | Осуществляет ввод с клавиатуры возраста человека | Осуществляет ввод с клавиатуры имени человека |
| **Входные данные:** | | |
| через параметры по значению |  |  |
| через параметры по ссылке |  |  |
| через параметры с использованием указателей |  |  |
| через глобальные объекты |  |  |
| **Выходные данные:** | | |
| через возвращаемое значение | возраст  (unsigned short) | имя  (std::string) |
| через параметры по ссылке |  |  |
| через параметры с помощью указателей |  |  |
| через глобальные объекты |  |  |

Таблица 2 – Описание функций ReadPersonHeight и ReadPersonWeight

|  |  |  |
| --- | --- | --- |
|  | **ReadPersonHeight** | **ReadPersonWeight** |
| **Назначение функции** | Осуществляет ввод с клавиатуры роста человека | Осуществляет ввод с клавиатуры веса человека |
| **Входные данные:** | | |
| через параметры по значению |  |  |
| через параметры по ссылке |  |  |
| через параметры с использованием указателей |  |  |
| через глобальные объекты |  |  |
| **Выходные данные:** | | |
| через возвращаемое значение |  |  |
| через параметры по ссылке |  | вес  (unsigned short) |
| через параметры с помощью указателя |  |  |
| через глобальные объекты | рост: height  (unsigned short) |  |

Таблица 3 – Описание функций ReadPersonSalary и WritePersonData

|  |  |  |
| --- | --- | --- |
|  | **ReadPersonSalary** | **WritePersonData** |
| **Назначение функции** | Осуществляет ввод с клавиатуры заработной платы человека | Осуществляет вывод на экран имени, возраста, роста, веса, заработной платы человека. |
| **Входные данные:** | | |
| через параметры по значению |  |  |
| через параметры по ссылке |  | имя  (const std::string &) |
| рост  [по умолчанию “”]  (const std::string &) |
| вес  [по умолчанию “”]  (const std::string &) |
| через параметры с использованием указателей |  | Возраст  (const unsigned short \*) |
| через глобальные объекты |  | заработная плата: salary  (std::string) |
| **Выходные данные:** | | |
| через возвращаемое значение |  |  |
| через параметры по ссылке |  |  |
| через параметры с помощью указателя | заработная плата  (double \*) |  |
| через глобальные объекты |  |  |

Таблица 4 – Описание перегруженных функций ReadPersonData

|  |  |  |
| --- | --- | --- |
|  | **ReadPersonData (перегруженная функция)** | |
| **Назначение функции** | Осуществляет ввод с клавиатуры имя, возраст и заработную плату человека. Для ввода использовать разработанные ранее функции | Осуществляет ввод с клавиатуры имя, возраст, рост и вес человека. Для ввода использовать разработанные ранее функции. |
| **Входные данные:** | | |
| через параметры по значению |  |  |
| через параметры по ссылке |  |  |
| через параметры с использованием указателей |  |  |
| через глобальные объекты |  |  |
| **Выходные данные:** | | |
| через возвращаемое значение |  |  |
| через параметры по ссылке | имя  (std::string &) | имя  (std::string &) |
| возраст  (unsigned short &) | возраст  (unsigned short &) |
| заработная плата  (double &) | рост  (unsigned short &) |
|  | вес  (unsigned short &) |
| через параметры с помощью указателя |  |  |
| через глобальные объекты |  |  |

### Контрольные вопросы

1. Что такое машинный код?
2. Что такое объектный файл?
3. Что такое исходный файл?
4. Какую работу выполняет компилятор?
5. Какую работу выполняет компоновщик?
6. Что такое заголовочный файл?
7. Что такое статическая библиотека?
8. Что такое динамическая библиотека?
9. Что такое исполняемый файл?
10. Что такое область видимости переменной?
11. На какие группы можно разделить все переменные по области видимости?
12. Что такое связывание?
13. Чем отличаются глобальные переменные с внутренним, модульным и внешним связыванием?
14. Как определить переменную с внутренним связыванием?
15. Как определить переменную с внешним связыванием?
16. Как определить переменную без связывания?
17. Что такое продолжительность хранения?
18. Как определить переменную с автоматической продолжительностью хранения?
19. Как определить переменную со статической продолжительностью хранения?
20. В какой части памяти хранятся переменные со статической продолжительностью хранения?
21. В какой части памяти хранятся переменные с автоматической продолжительностью хранения?
22. Перечислите классы памяти переменных и укажите для каждого класса область видимости, связывание и продолжительность хранения. Как определяется переменная каждого класса?
23. Какие классы памяти могут иметь функции?
24. Что нужно добавить в код, чтобы можно было вызвать функцию, определённую в другом файле?
25. Что нужно добавить в код, чтобы можно было изменить значение глобальной переменной, определённой в другом файле?
26. Как сделать глобальную переменную или функцию недоступной из других файлов?
27. Какие действия выполняются при вызове функции?
28. Какие действия выполняются при возврате значения?
29. Что такое прототип функции?
30. Что такое указатель?
31. В чем отличие константных указателей и указателей на константный объект?
32. Привидите примеры определения указателей, указателей на константные объекты, константных указателей и константных указателей на константные объекты
33. Влияет ли изменение значения параметра функции на фактически переданную в функцию переменную, если передача происходит по значению?
34. Влияет ли изменение значения параметра функции на фактически переданную в функцию переменную, если передача происходит через указатель?
35. Можно ли возвращать указатель на локальный объект? Если нет, почему?