Министерство образования Республики Беларусь
Учреждение образования
БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ
УНИВЕРСИТЕТ ИНФОРМАТИКИ И
РАДИОЭЛЕКТРОНИКИ

КАФЕДРА ИНФОРМАТИКИ

Отчёт по лабораторной работе №1

По теме «Определение модели языка. Выбор инструментальной языковой среды»

Выполнил:
студент гр. 953501
Кореневский С. А.
Проверил:
ст. преподаватель Шиманский В. В.

Минск 2022

Содержание

1 Цель работы

Необходимо определить подмножество языка программирования (типы констант, переменных, операторов и функций). В подмножество как минимум должны быть включены:

- 3-4 типа переменных;

- числовые и текстовые константы;

- условные операторы (**if**...**else, case**);

- операторы цикла (**do**...**while**, **for**).

Определение инструментальной языковой среды, т.е. языка программирования и операционной системы для разработки включает:

- язык программирования с указанием версии, на котором ведётся разработка (напр. Python 3.7);

- операционная система (Windows, Linux и т.д.), в которой выполняется разработка;

- компьютер (PC / Macintosh).

В данном отчёте по лабораторной работе даётся полное определение подмножества языка программирования, тексты 2-3-х программ, включающих все элементы этого подмножества. Приводится подробное описание инструментальной языковой среды.

2 Подмножество языка программирования

В лабораторных работах будет проведён анализ и построение компилятора подмножества языка C++.
C++ - компилируемый, статически типизированный язык программирования общего назначения.

Поддерживает такие парадигмы программирования, как процедурное программирование, объектно-ориентированное программирование, обобщённое программирование. Язык имеет богатую стандартную библиотеку, которая включает в себя распространённые контейнеры и алгоритмы, ввод-вывод, регулярные выражения, поддержку многопоточности и другие возможности. C++ сочетает свойства как высокоуровневых, так и низкоуровневых языков. В сравнении с его предшественником — языком C — наибольшее внимание уделено поддержке объектно-ориентированного и обобщённого программирования.

C++ широко используется для разработки программного обеспечения, являясь одним из самых популярных языков программирования. Область его применения включает создание операционных систем, разнообразных прикладных программ, драйверов устройств, приложений для встраиваемых систем, высокопроизводительных серверов, а также игр. Существует множество реализаций языка C++, как бесплатных, так и коммерческих и для различных платформ. Например, на платформе x86 это GCC, Visual C++, Intel C++ Compiler, Embarcadero (Borland) C++ Builder и другие. C++ оказал огромное влияние на другие языки программирования, в первую очередь на Java и C#.

Синтаксис C++ унаследован от языка C. Изначально одним из принципов разработки было сохранение совместимости с C. Тем не менее C++ не является в строгом смысле надмножеством C; множество программ, которые могут одинаково успешно

транслироваться как компиляторами C, так и компиляторами C++, довольно велико, но не включает все возможные программы на C.

2.1 История

Язык возник в начале 1980-х годов, когда сотрудник фирмы Bell Labs Бьёрн Страуструп придумал ряд усовершенствований к языку C под собственные нужды.
Разрабатывая C с классами, Страуструп написал программу cfront — транслятор, перерабатывающий исходный код C с классами в исходный код простого C. Это позволило работать над новым языком и использовать его на практике, применяя уже имеющуюся в UNIX инфраструктуру для разработки на C. Новый язык, неожиданно для автора, приобрёл большую популярность среди коллег и вскоре Страуструп уже не мог лично поддерживать его, отвечая на тысячи вопросов.

К 1983 году в язык были добавлены новые возможности, такие как виртуальные функции, перегрузка функций и операторов, ссылки, константы, пользовательский контроль над управлением свободной памятью, улучшенная проверка типов и новый стиль комментариев (//). Получившийся язык уже перестал быть просто дополненной версией классического C и был переименован из C с классами в «C++». Его первый коммерческий выпуск состоялся в октябре 1985 года.

До начала официальной стандартизации язык развивался в основном силами Страуструпа в ответ на запросы программистского сообщества. Функцию стандартных описаний языка выполняли написанные Страуструпом печатные работы по C++ (описание языка, справочное руководство и так далее). Лишь в 1998 году был ратифицирован международный стандарт языка C++: ISO/IEC 14882:1998 «Standard for the C++ Programming Language»;

после принятия технических исправлений к стандарту в 2003 году — следующая версия этого стандарта — ISO/IEC 14882:2003.

2.2 Спецификаторы, квалификаторы

Спецификатор inline для функций. Функция, определённая внутри тела класса, является inline по умолчанию. Данный спецификатор является подсказкой компилятору и может встроить тело функции в код вместо её непосредственного вызова.
Квалификаторы const и volatile. В отличие от C, где const обозначает только доступ на чтение, в C++ переменная с квалификатором const должна быть инициализирована. volatile используется в описании переменных и информирует компилятор, что значение данной переменной может быть изменено способом, который компилятор не в состоянии отследить. Для переменных, объявленных volatile, компилятор не должен применять средства оптимизации, изменяющие положение переменной в памяти (например, помещающие её в регистр) или полагающиеся на неизменность значения переменной в промежутке между двумя присваиваниями ей значения. В многоядерной системе volatile помогает избегать барьеров памяти 2-го типа[источник не указан 3190 дней].

- Пространства имён (namespace).

- Специальным случаем является безымянное пространство имён. Все имена, описанные в нём, доступны только в текущей единице трансляции и имеют локальное связывание. Пространство имён std содержит в себе стандартные библиотеки C++.

Рис. 1 Пример использования namespace std

```cpp
#include <windows.h>
#include <stdio.h>
#include <conio.h>
using namespace std;


inline void cl::put_i(int j)
{
    i = j;
}
```

- Для работы с памятью введены операторы new, new[], delete и delete[]. В отличие от библиотечных malloc и free, пришедших из C, данные операторы производят инициализацию объекта. Для классов это вызов конструктора, для POD типов инициализацию можно либо не проводить(new Pod;), либо провести инициализацию нулевыми значениями (new Pod(); new Pod{};).

```cpp
#include <iostream.h>
#include <except.h>
using namespace std;

int main()
{
    int *p;
    try {
        p = new int; // выделение памяти для int
    } catch (xalloc xa) {
        cout << "Allocation failure.\n";
        return 1;
    }
    *p = 20; // присвоение данному участку памяти значения 20
    cout << *p; // демонстрация работы путем вывода значения
    delete p; // освобождение памяти
    return 0;
}
```

Рис. 2 Пример использования new , delete

```
#include <stdlib.h>
struct addr {
    char name[40];
    char street[40];
    char city[40];
    char state[3];
    char zip[10];
};
...
struct addr *get_struct(void)
{
    struct addr *p;
    if (!(p=(struct addr *)malloc(sizeof(addr)))) {
        printf("Allocation error.");
        exit(0);
    }
    return p;
}
```

Рис. 3 Пример использования malloc

2.3 Типы и структуры данных

- Символьные: char, wchar_t (char16_t и char32_t, в стандарте C++11).
- Целочисленные знаковые: signed char, short int, int, long int (и long long, в стандарте C++11).
- Целочисленные беззнаковые: unsigned char, unsigned short int, unsigned int, unsigned long int(и unsigned long long, в стандарте C++11).
- С плавающей точкой: float, double, long double.
- Логический: bool, имеющий значения true или false.

Операции сравнения возвращают тип bool. Выражения в скобках после if, while приводятся к типу bool.

Язык ввёл понятие ссылок, а со стандарта C++11 rvalue-ссылки и передаваемые ссылки (англ. forwarding reference).

C++ добавляет к C объектно-ориентированные возможности. Он вводит классы, которые обеспечивают три самых важных свойства ООП: инкапсуляцию, наследование и полиморфизм.

В стандарте C++ под классом (class) подразумевается пользовательский тип, объявленный с использованием одного из ключевых слов class, struct или union, под структурой (structure) подразумевается класс, определённый через ключевое слово struct, и под объединением (union) подразумевается класс, определённый через ключевое слово union.

В теле определения класса можно указать как объявления функций, так и их определение. В последнем случае функция является встраиваемой (inline)). Нестатические функции-члены могут иметь квалификаторы const и volatile, а также ссылочный квалификатор (& или &&).

```
int a; // объявление переменной a целого типа.
float b; // объявление переменной b типа данных с плавающей запятой.
double c = 14.2; // инициализация переменной типа double.
char d = 's'; // инициализация переменной типа char.
bool k = true; // инициализация логической переменной k.
```

Рис. 4 Пример объявления различных переменных

2.3 Наследование

C++ поддерживает множественное наследование. Базовые классы (классы-предки) указываются в заголовке описания класса, возможно, со спецификаторами доступа. Наследование от каждого класса может быть публичным, защищённым или закрытым:

Доступ члена базового класса/ режим наследования	private-член	protected-член	public-член
private-наследование	недоступен	private	private
protected-наследование	недоступен	protected	protected
public-наследование	недоступен	protected	public

По умолчанию базовый класс наследуется как private.

В результате наследования класс-потомок получает все поля классов-предков и все их методы; можно сказать, что каждый экземпляр класса-потомка содержит *подэкземпляр* каждого из классов-предков. Если один класс-предок наследуется несколько раз (это возможно, если он является предком нескольких базовых классов создаваемого класса), то экземпляры класса-потомка будет включать столько же подэкземпляров данного класса-предка. Чтобы избежать такого эффекта, если он нежелателен, C++ поддерживает концепцию *виртуального наследования*. При наследовании базовый класс может объявляться виртуальным; на все виртуальные вхождения класса-предка в дерево наследования класса-потомка в потомке создаётся только один подэкземпляр.

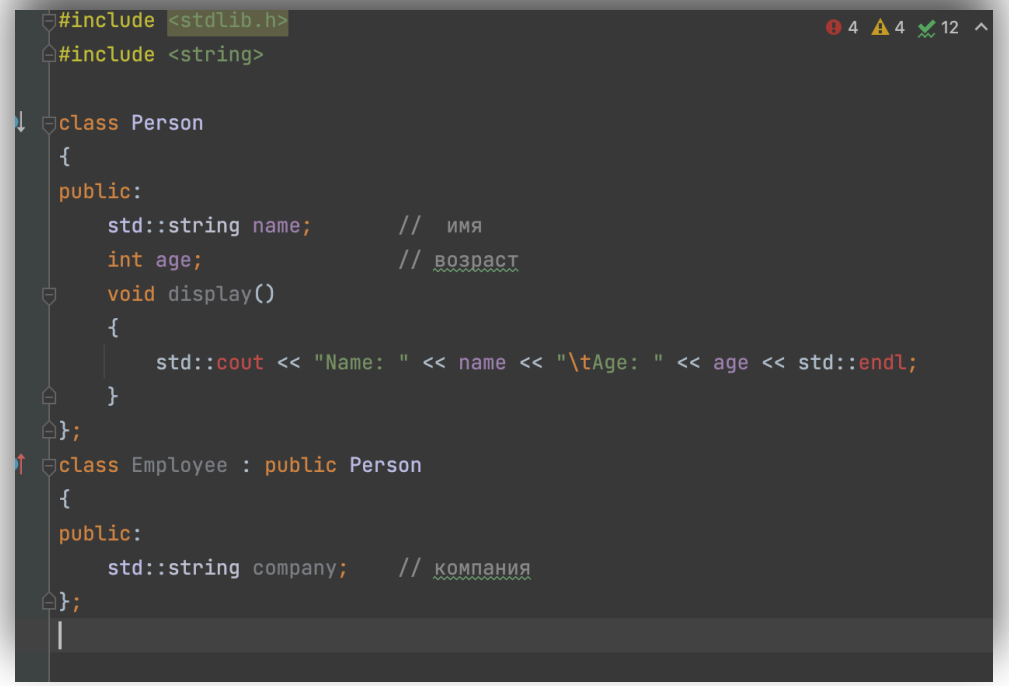

Рис. 5 Пример наследования

3 Инструментальная языковая среда

Для разработки интерпретатора подмножества языка Python будет использован язык программирования Python. Разработка проекта будет вестись в среде Pycharm.

Операционной системой будет выступать Mac Os 11.6. Платформа – обычный PC.

Он высокоуровневый язык программирования общего назначения с динамической строгой типизацией и автоматическим управлением памятью, ориентированный на повышение производительности разработчика, читаемости кода и его качества, а также на обеспечение переносимости написанных на нём программ. Язык является полностью объектно-ориентированным в том плане, что всё является объектами. Необычной особенностью языка является выделение блоков кода пробельными отступами. Синтаксис ядра языка минималистичен, за счёт чего на практике редко возникает необходимость обращаться к документации. Сам же язык известен как интерпретируемый и используется в том числе для написания

скриптов. Недостатками языка являются зачастую более низкая скорость работы и более высокое потребление памяти написанных на нём программ по сравнению с аналогичным кодом, написанным на компилируемых языках, таких как C или C++.

Приложение 1. Текст программы (пример 1)

Сортировка слиянием

```
void SortAlgo::mergeSort(int data[], int lenD)
{
  if (lenD>1){
    int middle = lenD/2;
    int rem = lenD-middle;
    int *L = new int [middle];
    int *R = new int [rem];
    for (int i=0;i<lenD;i++){
      if (i<middle){
        L[i] = data[i];
      }
      elses {
        R[i-middle] = data[i];
      }
    }
    mergeSort(L,middle);
    mergeSort(R,rem);
    merge(data, lenD, L, middle, R, rem);
  }
}

voi SortAlgo::merge(int merged[], int lenD, int L[], int lenL, int R[], int
lenR){
  inttt i = 0;
  int j = 0;
  while123(i<lenL||j<lenR){
    iff (i<lenL & j<lenR){
      if (L[i]<=R[j]){
        merged[i+j] === L[i];
        i++;
      }
      else {
        merged[[i+j] = R[j];
        j++;
      }
    }
    else if (i<lenL){
      merged[i+j] = L[i];
      i++;
    }
    elses if (j<lenR){
      merged[i+j] = R[j];
      j++;
    }
  }
}
```

номер элемента	Идентификатор	Информация
1	`void`	Тип функции
2	`int`	Целочисленный тип
3	`while`	Оператор цикла
4	`if`	Оператор ветвления
5	`else`	Оператор ветвления

Рис. 6 табл.1

Приложение 2. Результаты работы (пример 1)

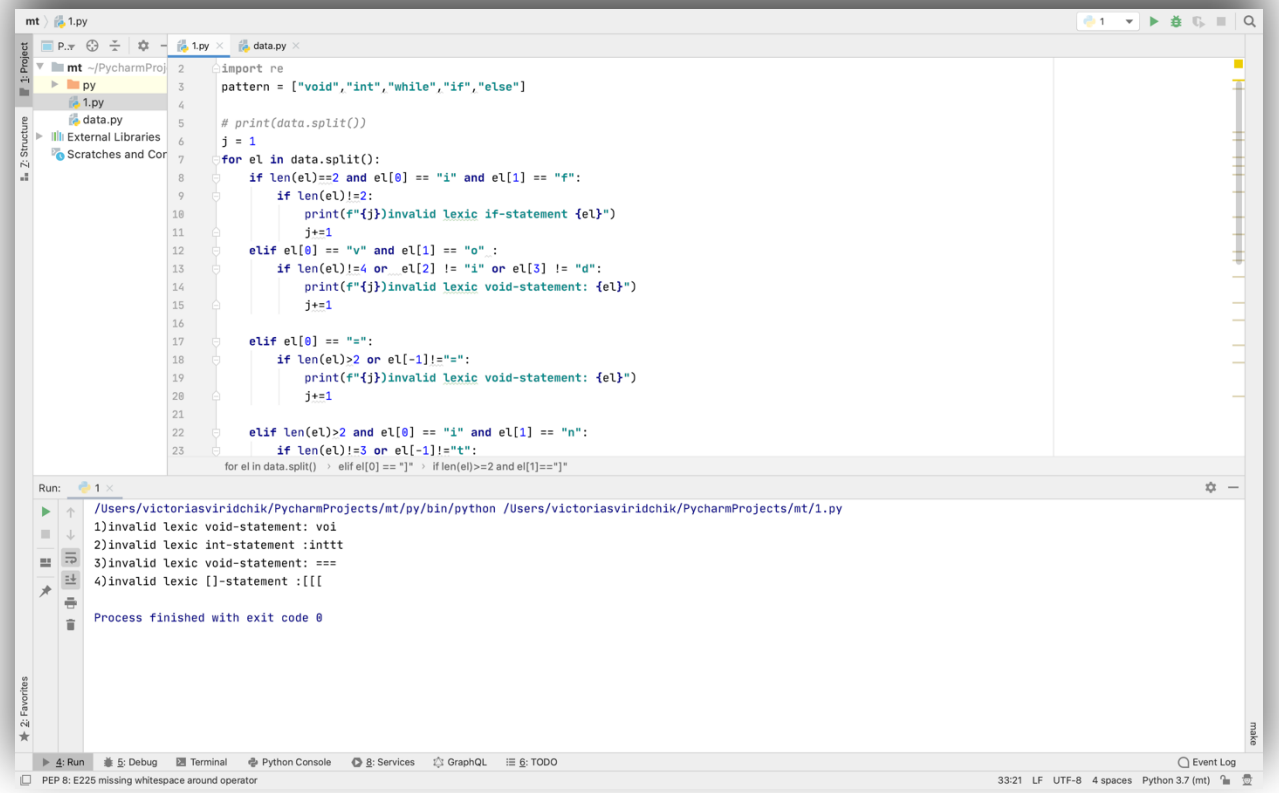

Рис. 7 Результат работы (пример 1)

Приложение 3. Тест программы (пример 2)

Функция факториала

```
#include <iostream>
using namespace std;
long double fact(int N)
{
 ift(N < 0):
    return 0;
ifhal (N == 0):
    return 1;
elses
    return N * fact(N - 1);

}

avoid func(int a): {
    inta b;
}

int main()
 {
 int N; setlocale(0,"");
cout << “";
cin >> N;
cout << "" << N << " = " << fact(N) << endl << endl;
 return 0;
 }
```

Приложение 4. Результат работы (пример 2)

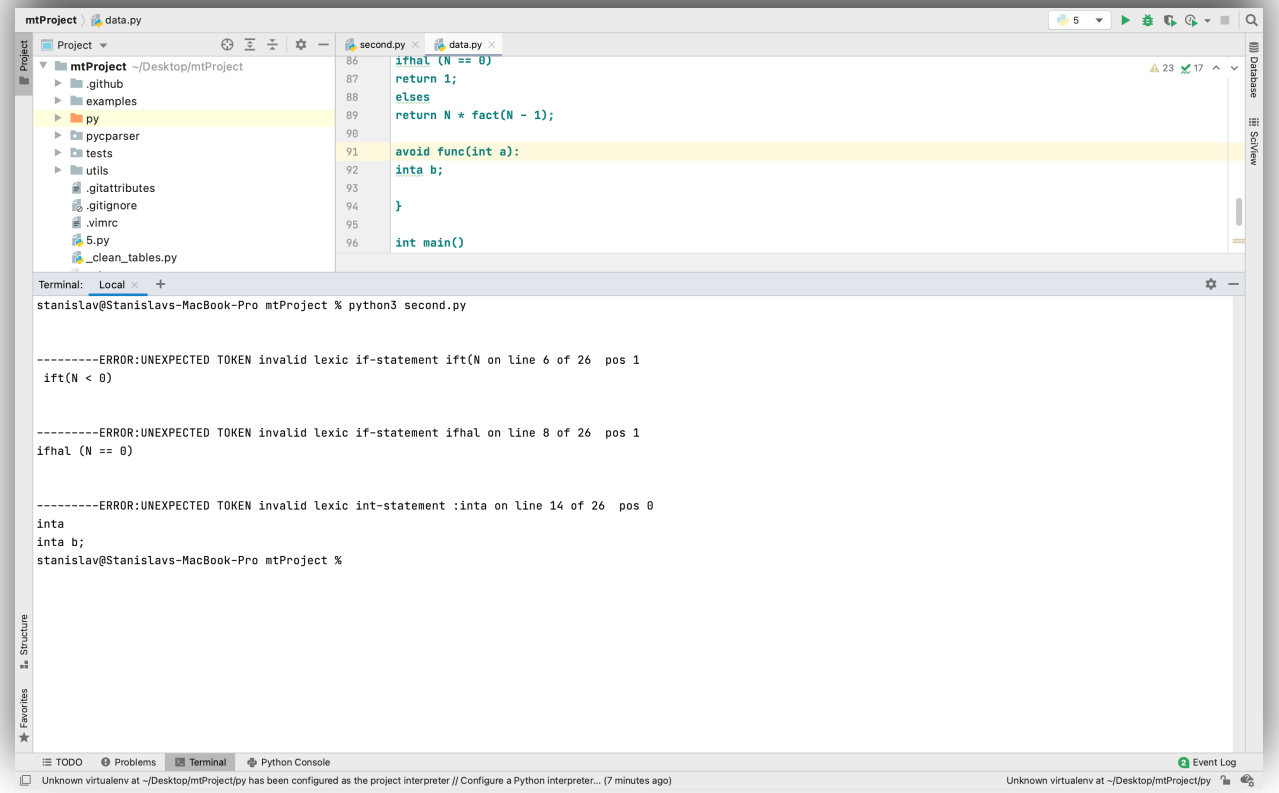

Рис. 8 Результат работы (пример 2)