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. 853504
Шевченя И.В.
Проверил:
Ст. преподаватель КИ Шиманский В. В.

Минск 2021

Содержание

1. Цель работы

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

- числовые и текстовые константы;

- 3-4 типа переменных;

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

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

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 3.6);

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

- компьютер;

2. Подмножество языка программирования

В качестве подмножества языка программирования выбран язык C++.

C++ — компилируемый, статически типизированный язык программирования общего назначения. Поддерживает как процедурное программирование, объектно-ориентированное программирование, обобщенное программирование. Язык имеет богатую стандартную библиотеку, которая включает в себя распространённые контейнеры и алгоритмы, ввод-вывод, поддержку многопоточности и другие возможности. C++ сочетает свойства как высокоуровневых, так и низкоуровневых языков.

2.1 Числовые и текстовые константы

- -5, -4, -3, -2, -1, 0, 1, 2, 3, 4, 5 (int).
- 8.1, 4.3, -2.9, -11.7 (float литералы)
- `a`, `b`, `c` (char литералы)
- "", "hello" (char[] литералы)
- true, false (bool литералы)

2.2 Типы переменных

C++ поддерживает статическую типизацию. В C++ имеются встроенные типы, которые практически полностью повторяют типы данных в C: логический, символьные, целочисленные знаковые, целочисленные беззнаковые, число с плавающей точкой и некоторые другие. Из коллекций в C++ встроены: массивы статические и динамические.

2.3 Операторы цикла

- while - выполняет тело цикла до тех пор, пока условие цикла истинно.

```
int collection_sum(int* array) {
    int i = 0;
    int sum = 0;
    while (i < len(array)) {
        sum += array[i];
        i++;
    }

    return sum;
}
```

- for - выполняет тело цикла, итерируясь из переданного условия

```
int collection_sum(int* array) {
    sum = 0;
    for (int i = 0; i < length(array); i++)
        sum += element;

    return sum;
}
```

- break - прерывает исполнение цикла

```
void example() {
    while (i <= 10) {
        if (i % 7 == 0)
            break;
        print(i);
    }

    print(“The end“);
}
```

- continue — начинает следующий проход цикла, не исполняя оставшееся тело цикла

```
void example() {
    while (i <= 10) {
        if (i % 2 == 1)
            continue;
        print(i);
    }

    print(“The end“);
}
```

2.3 Условные операторы

- Оператор *if*

```
if выражение {
    инструкция_1
```

```
    инструкция_2
    ...
    инструкция_n


}
```

Пример:

```
void example() {
    num = 3;
    if (num > 0)
        cout <<  num << “is a positive number.“;

    cout << “This is always printed.“;

}
```

- Конструкция if - else

```
if выражение {
  инструкция_1
  инструкция_2
  ...
  инструкция_n

}
else {
  инструкция_a
  инструкция_b
  ...
  инструкция_x


}
```

Пример:

```
void example(int* array){
    for (element: array) {
        if (element % 2 == 0){
            cout << element << “ even“;
        }
        else {
            cout << element << “ odd“;
        }
    }
}
```

3. Инструментальная языковая среда

В качестве языковой среды выбран язык программирования Python 3.8. Разработка основана на работе с операционной системой Linux на PC.

Python — высокоуровневый язык программирования общего назначения, ориентированный на повышение производительности разработчика и читаемости кода. Синтаксис ядра Python минималистичен. В то же время стандартная библиотека включает большой объём полезных функций.

Python поддерживает структурное, объектно-ориентированное, функциональное, императивное и аспектно-ориентированное программирование. Основные архитектурные черты — динамическая типизация, автоматическое управление памятью, полная интроспекция, механизм обработки исключений, поддержка многопоточных вычислений, высокоуровневые структуры данных. Поддерживается разбиение программ на модули, которые, в свою очередь, могут объединяться в пакеты.

Эталонной реализацией Python является интерпретатор CPython, поддерживающий большинство активно используемых платформ. Он распространяется под свободной лицензией Python Software Foundation License, позволяющей использовать его без ограничений в любых приложениях, включая проприетарные. Есть реализация интерпретатора для JVM с возможностью компиляции, CLR, LLVM, другие независимые реализации. Проект PyPy использует JIT-компиляцию, которая значительно увеличивает скорость выполнения Python-программ.

Приложение. Текст программ

1 Сортировка слиянием

```
void merge(int *a, int n)
{
  int mid = n / 2;
  if (n % 2 == 1)
    mid++;
  int h = 1;
  int *c = (int*)malloc(n * sizeof(int));
  int step;
  while (h < n)
  {
    step = h;
    int i = 0;
    int j = mid;
    int k = 0;
    while (step <= mid)
    {
      while ((i < step) && (j < n) && (j < (mid + step))) {
        if (a[i] < a[j]) {
          c[k] = a[i];
          i++; k++;
        }
        else {
          c[k] = a[j];
          j++; k++;
        }
      }
      while (i < step) {
        c[k] = a[i];
        i++; k++;
      }
      while ((j < (mid + step)) && (j<n)) {
        c[k] = a[j];
        j++; k++;
      }
      step = step + h;
    }
    h = h * 2;

    for (i = 0; i<n; i++)
      a[i] = c[i];
  }
}

int main()
{
  int a[8];
```

```
  for (int i = 0; i < 8; i++)
    a[i] = rand() % 20 – 10;

  for (int i = 0; i < 8; i++)
    printf("%d ", a[i]);

  printf("\n");
  merge(a, 8);
  for (int i = 0; i < 8; i++)
    printf("%d ", a[i]);

  printf("\n");
  getchar();
  return 0;
}
```

2 Перестановка массива в обратном порядке

```
void reverseArray(int arr[], int start, int end)
{
    while (start < end)
    {
        int temp = arr[start];
        arr[start] = arr[end];
        arr[end] = temp;
        start++;
        end--;
    }
}

void printArray(int arr[], int size)
{
    for (int i = 0; i < size; i++)
    cout << arr[i] << " ";

    cout << endl;
}


int main()
{
    int arr[] = {1, 2, 3, 4, 5, 6};

    int n = sizeof(arr) / sizeof(arr[0]);

    printArray(arr, n);
    rvereseArray(arr, 0, n-1);

    cout << "Reversed array is" << endl;
    printArray(arr, n);

    return 0;
}
```