МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

**«БЕЛГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ
ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ им. В. Г. ШУХОВА»
(БГТУ им. В.Г. Шухова)**

Кафедра программного обеспечения вычислительной техники и автоматизированных систем

Лабораторная работа №6а

по дисциплине: Основы программирования
тема: «Потоки. Ссылки»

Выполнил: ст. группы ПВ-211
Чувилко Илья Романович

Проверили:
Притчин Иван Сергеевич
Черников Сергей Викторович

Белгород 2022 г.

Цель работы: получение навыком работы с потоками, ссылками, управляющими конструкциями; осознание необходимости появления данных языковых средств.

Содержание отчета:
Тема лабораторной работы.
Цель лабораторной работы.
Тексты задания с набранными фрагментами кода к каждому из пунктов и ответами на вопросы по ходу выполнения работы.

Задания к лабораторной работе:

1. Создайте функцию `infinitivePause`, которая в своём теле будет воспроизводить бесконечный цикл.

```
void infinitivePause() {
  while (true) {
    while (true) {
      while (true) {
        while (true) {
          while (true) {
            // Вечный цикл для того, чтобы я мог находиться в вечном цикле пока
            // нахожусь в вечном цикле, пока нахожусь в вечном цикле, пока
нахожусь...
          }
        }
      }
    }
  }
}
```

№2. Отправьте в объект cout, связанный со стандартным потоком вывода, произвольное сообщение без использования endl. После чего сделайте вызов infinitivePause. Запустите программу. Опишите наблюдаемое поведение.

Ответ: Если не вносить никаких изменений в программу, то выведется текст, после чего программа впадет в вечный цикл. Произойдет это из-за того, что по стандарту размер буфера составляет 0 байт, следовательно после вызова функции cout случится переполнение и вывод сообщения в консоль. Но если добавить функцию setvbuf(stdout, nullptr, _IOFBF, 1024), то размер буфера станет равным 1024 байтам, и при запуске программы, программа не выводя ничего в консоль впадет в вечный цикл;

№3. Выполните аналогичные пункту 2 действия только с использованием endl. Опишите наблюдаемое поведение.

Ответ: Вне зависимости от того, какого размера будет буфер, в консоль выведется сообщение. Связано это с тем, что endl очищает буфер после вызова;

№4. Для случая 2, после вывода сообщения вставьте инструкцию ввода какой-нибудь переменной. Запустите приложение. Опишите наблюдаемое поведение.

Ответ: Вне зависимости от того, какого размера будет буфер, в консоль выведется сообщение. Связано это с тем, что cin так же очищает буфер после вызова;

№5. Выполните создание строки, состоящей из 10000 символов ’a’;
Осуществите её вывод перед вызовом infinitivePause. Опишите наблюдаемое поведение.

Ответ: При выводе 10000 символов, происходит переполнение буфера и в консоль выводится число символов, кратное размеру буфера. Например, если выделить 512 байт под буфер, то:
1) В буфере поместится 512 char-символов;
2) Произойдет переполнение 19 раз, а следовательно в консоль выведется 9728 символов
3) Оставшиеся 272 символа останутся в буфере, а программа уйдет в вечный цикл

№6. Таким образом, выделите 3 случая, когда очищается буфер вывода.

Ответ:

1. При переполнении буфера;
2. При принудительной очистке буфера (endl);
3. При вводе переменной (cin)

№7. Говорят, что частый сброс буфера способен повлиять на производительность приложения. Проведите эксперимент, доказывающий это.

Ответ:

```
"D:\BGTU\Programming Basics\6a\cmake-build-debug\6a.exe"
Often buffer reset: 3.025
Buffer opt: 0.051

Process finished with exit code 0
```

Вывод: частый сброс буфера замедлил скорость выполнения в 59 раз;

№8. Рассмотрим случай, когда частый сброс буфера вывода из-за чередования ввода / вывода оказывает влияние на производительность. Выполните решение задачи Минимальная OR сумма (1635A) на codeforces и приложите вердикт тестирующей системы.
Код программы:

```
void number1_minORSum_Slow() {
  std::setvbuf(stdout, nullptr, _IOFBF, 2048);

  int t;
  std::cin >> t;
  for(int k = 0; k < t; k++) {
    int n;
    std::cin >> n;

    int minSum = 0;
    for (int i = 0; i < n; i++) {
      int a;
      std::cin >> a;
      minSum |= a;
    }

    std::cout << minSum << '\n';
  }
}
```

Ответ: В консоль не выводится сообщение, так как вызов cin больше не очищает буфер, связанный с выводом в консоль;

№10. В решение вашей задачи для пункта 8 добавьте строки

```
cin.tie(nullptr);
ios_base::sync_with_stdio(false);
```

Вердикт тестирующей системы:

151256680	28.03.2022 18:52	IlyaChuvilko	1635A - Минимальная OR сумма	GNU C++14	**Полное решение**	30 мс	0 КБ

0.1.2 Некоторые особенности вывода

№1. При помощи cout можно выводить в поток вывода значения переменных и адреса:

```
int i = 10;
cout << i << ' ' << &i;
```

Создайте переменную s типа const char* и попробуйте вывести её в поток вывода. Получившееся поведение опишите.

Ответ: Выведется адрес строки;

№3. Выведите литералы true и false в поток вывода. Какие значения были отображены на экране?

Ответ: 1 и 0;

№4. Добавьте манипулятор std::boolalpha.
Опишите отображаемый вывод. Проверьте, распространяется ли действие манипулятора на последующие выводы.

Ответ: true и false. Действие манипулятора распространилось на последующие выводы.

0.1.3 Ввод

№1. Создайте переменную целочисленного типа, но при осуществлении ввода введите строку. Опишите наблюдаемое поведение.

Ответ: Выделим 4 случая:

1. Вводится строка, состоящая только из чисел, при этом не выходя за область значений int. **В таком случае, переменной присваивается введенное значение.**
2. Вводится строка, состоящая только из чисел, но значение выходит за область значений int. **В таком случае, переменной присваивается крайнее значение области (2 147 483 647 или -2 147 483 648)**
3. Вводится строка, состоящая как из чисел, так и из символов, но сначала будут записаны числа. **В таком случае, переменной присваивается значение, которое идет до первого символа.**
4. Вводится строка, состоящая как из чисел, так и из символов, но сначала будут записаны символы. **В таком случае, переменной присваивается значение 0.**

№2. Переменная, связанная с потоком ввода будет возвращать значение "истина", если поток ввода не находится в состоянии ошибки. Попробуйте запустить пример из прошлого пункта, добавив обработку:

```
int a;
std::cin >> a;

if (std::cin)
    std::cout << "Success";
else
    std::cout << "Error";
```

Поэксперементируйте с разными вариантами ввода. Приведите пример такого ввода, который выдаёт "Success", но содержит символы, отличные от цифр.

Ответ: Пример 3 из прошлой задачи;

№3. Создайте переменную типа char и попробуйте выполнить ввод пробельного символа. Опишите наблюдаемое поведение.

Ответ: Ничего не происходит.

№4. Проделайте то же самое но при считывании осуществляйте ввод через cin.get():

Ответ: Теперь переменной присваивается введенный символ

0.1.4 Файловый ввод / вывод

№3. Код описанный выше несколько небезопасен. Файл с таким именем мог и не существовать. Попробуйте запустить прошлый фрагмент для файла, который не был создан до этого и опишите наблюдаемое поведение.

Ответ: Программа ничего не делает с фактом отсутствия файла с таким именем и просто продолжает выполнение.

5. В примере выше мы осуществляли ввод в строку, но с таким же успехом вы можете считывать данные в переменные, например, целочисленного типа.

```
int x;
while (inputFile >> x) {

}
```

Имея данные сведения, реализуйте функцию `long long getSum(const std::string &filename)` для вычисления суммы чисел, записанных в файл с именем `filename`. Если файла с таким именем нет, функция должна возвращать -1.

```
#include <iostream>
#include <fstream>

int main() {
  std::ifstream inputFile("input.txt");
  if (!inputFile) {
    std::cout << "File don't exist!";
    return -1;
  }

  int a;
  int s = 0;
  while (inputFile >> a)
    s += a;

  return 0;
}
```

6. История с возвратом значения -1, если файл не был найден, смотрится несколько нелепо. В C++ появились другие средства для сигнализирования об ошибках, называемых исключениями. Мы не будем вдаваться в подробности сейчас. Если вы хотите сигнализировать, что возникла некоторая ошибка времени исполнения, можно использовать такой подход:

```cpp
#include <stdexcept>

long long getSum(const std::string &filename) {
    std::ifstream inputFile(filename);

    if (!inputFile)
        throw std::runtime_error("File doesn't exist");

    // работа с файлом

}
```

Модифицируйте программу из пункта 5, запустите её для несуществующего файла. Вставьте в отчёт полученное сообщение.

```
"D:\BGTU\Programming Basics\6a\cmake-build-debug\6a.exe"
terminate called after throwing an instance of 'std::runtime_error'
  what():  File doesnt exist

Process finished with exit code 3
```

№7. Функция long long getSumOfMaxesInRows(const std::string &filename) должна выполнять поиск суммы максимальных элементов строк.

```cpp
long long getSumOfMaxesInRows(const std::string &filename) {
  std::ifstream input_file(filename);

  if (!input_file.is_open())
    throw std::runtime_error("file does not exist");
  long long sum = 0;

  size_t height, width;
  input_file >> height >> width;
  for (size_t h = 0; h < height; h++) {
    int max;
    input_file >> max;
    for (size_t w = 1; w < width; w++) {
      int cur;
      input_file >> cur;
      if (max < cur)
        max = cur;
    }
    sum += max;
  }

  return sum;
}
```

0.1.6 Определение операций ввода / вывода для произвольных типов

№1. Опишите произвольную (но отличную от примера) структуру.

```
struct Product {
  std::string name;
  int price;
  size_t quantity;
};
```

№2. Реализуйте функции ввода и вывода структуры:

```
void inputProduct(Product &p) {
  std::cout << "Enter product name: ";
  std::cin >> p.name;
  std::cout << "Enter price: ";
  std::cin >> p.price;
  std::cout << "Enter quantity product in stock: ";
  std::cin >> p.quantity;
}

void outputProduct(Product &p) {
  std::cout << "Product name: " << p.name;
  std::cout << "\nEnter price: " << p.price;
  std::cout << "\nEnter quantity product in stock: " << p.quantity;
}
```

5. Попробуйте написать цепочку вида:

```
Person p1, p2;
std::cin >> p1 >> p2;
```

Опишите наблюдаемое поведение.

Ответ: Ошибка

0.1.7 Ссылки

№1. Реализуйте функцию sort2 с использованием указателей. Введите два значения в main и выполните вызов функции sort2.

```
void sort2(int *a, int *b) {
  if (*a > *b) {
    int t = *a;
    *a = *b;
    *b = t;
  }
}

int main() {
  int a, b;
  std::cin >> a >> b;

  sort2(&a, &b);
  std::cout << a << ' ' << b;

  return 0;
}
```

№2. Реализуйте функцию sort2 с использованием ссылок. Введите два значения в main и выполните вызов функции sort2.

```
void sort2(int &a, int &b) {
  if (a > b) {
    int t = a;
    a = b;
    b = t;
  }
}

int main() {
  int a, b;
  std::cin >> a >> b;

  sort2(a, b);
  std::cout << a << ' ' << b;

  return 0;
}
```

№3. Создайте в функции main вектор из 1000000 целых значений.
Напишите функции которые в своём теле ничего не делают, но принимают вектор следующими способами:

- По значению
- По значению указателя
- По ссылке

Замерьте время вызова каждой из функций и приложите к отчёту вместе с кодом эксперимента. Сделайте выводы.

```cpp
void fByValue(std::vector<int> v) {

}

void fByCursor(std::vector<int> *v) {

}

void fByLink(std::vector<int> &v) {

}

int main() {
  std::vector<int> v(1000000);

  TIME_TEST({
    fByValue(v);
  }, "Time by value");

  TIME_TEST({
    fByCursor(&v);
  }, "Time by cursor");

  TIME_TEST({
    fByLink(v);
    }, "Time by link");

  return 0;
}
```

```
"D:\BGTU\Programming Basics\6a\cmake-build-debug\6a.exe"
Time by value: 0.003
Time by cursor: 0
Time by cursor: 0

Process finished with exit code 0
```

№4. Напишите следующие функции:

- Ввод вектора целых чисел
- Вывод вектора целых чисел
- Поиск минимального значения среди элементов вектора
- Вводится последовательность с клавиатуры, признак конца ввода – 0. Изменить порядок следования элементов в векторе на обратный5 . Индексы в функции изменения порядка должны иметь тип size_t. Убедитесь, что функция корректно работает для пустого вектора (проще обработать данный случай до тела цикла). Убедитесь, что используете оптимальный способ передачи вектора в функцию (по ссылке или по ссылке на константу).

```cpp
void inputVector(std::vector<int> &v) {
  for (size_t i = 0; i < v.size(); i++)
    std::cin >> v.data()[i];
}

void outputVector(std::vector<int> &v) {
  for (size_t i = 0; i < v.size(); i++) {
    std::cout << v.data()[i] << ' ';
  }
}

int getMinInVector(std::vector<int> &v) {
  int min = v.data()[0];
  for (size_t i = 1; i < v.size(); i++)
   if (min > v.data()[i])
     min = v.data()[i];
  return min;
}

void inputFromKeyboard(std::vector<int>(&v)) {
  while (true) {
    int x;
    std::cin >> x;
    if (x == 0)
      return;
    v.push_back(x);
  }
}

void reverseVector(std::vector<int>(&v)) {
  for (int i = 0; i < v.size() / 2; ++i)
    swap(v[i], v[v.size() - i - 1]);
}
```

№5. Решите следующую задачу, используя указатели: даны коэффициенты a, b, c квадратного уравнения. Гарантируется, что количество корней равняется двум. Вывести полученные корни. Корни должны выводиться в функции main, а их вычисление – происходить в void getRoots(int a, int b, int c, double *x1, double *x2).

```cpp
void getRoots(int a, int b, int c, double *x1, double *x2) {
  double d = sqrt(pow(b, 2) - 4 * a * c);
  *x1 = (-b + d) / 2 * a;
  *x2 = (-b - d) / 2 * a;
}
```

№6. Для условия задачи 1, выполните переход на ссылки: void getRoots(int a, int b, int c, double &x1, double &x2)

```cpp
void getRoots(int a, int b, int c, double &x1, double &x2) {
  double d = sqrt(pow(b, 2) - 4 * a * c);
  x1 = (-b + d) / 2 * a;
  x2 = (-b - d) / 2 * a;
}
```

Аспект	Указатель	Ссылка
Обязан быть инициализирован	-	+
Особенности обращения к элементам	Необходимо ставить знак указателя, чтобы обратиться к значению	Нет необходимости ставить что-либо, чтобы обратиться к значению
Возможность перенацеливания	Есть	Нет
Возможность получения адреса переменной (адреса указателя или адреса ссылки)	Есть	Нет
Возможность непосредственной работы с динамической памятью	Есть	Нет
Возможность создавать массивы	Есть	Нет