Интернет-технологии в профессиональной деятельности

Преподаватель:
к.ф.-м.н., заведующий кафедрой ВТиКГ ДВГУПС,
преподаватель программы IT-школа Samsung,
Пономарчук Юлия Викторовна
E-mail: yulia.ponomarchuk@gmail.com

Организация обучения

- Еженедельно проводится лекция 90 мин.
- На неделю выдается пакет заданий
- Отводится неделя на решение заданий, приветствуется обсуждение на занятии; можно также задавать вопросы по электронной почте
- Выполненные задания просим высылать по электронной почте, преподаватель их проверяет и отправляет вам ответ с указаниями ошибок, рекомендациями
- Через 1-2 недели после выдачи заданий на форуме выкладывается подробный разбор решения заданий

Что нужно для успешного выступления на олимпиаде

1. Прекрасное владение языком программирования
2. Уверенное знание большого количества алгоритмов (уверенно знать – значит уметь быстро, без подготовки реализовать алгоритм)
3. Математическая подготовка
4. Большое количество прорешенных задач
5. Опыт участия в тренировочных и реальных олимпиадах
6. Психологическая подготовка

Литература для дополнительного чтения

- Окулов С.М. «Программирование в алгоритмах», 2004
- Порублев И.Н., Ставройский А.Б. «Алгоритмы и программы. Решение олимпиадных задач», 2007
- Меньшиков Ф.В. «Олимпиадные задачи по программированию», 2006
- Андреева Е.В. «Математические основы информатики. Элективный курс», 2012
- Шень А. «Программирование. Теоремы и задачи», 2004

Интернет-ресурсы

- Учебные курсы www.intuit.ru
- Коллекция алгоритмов http://e-maxx.ru/algo
- Международные и всероссийские олимпиады по информатике http://info.rusolymp.ru
- Сайт школьных олимпиад, проводимых в Приморском крае http://imcs.dvgu.ru/works/school.html
- Площадка соревнований по программированию http://codeforces.ru/
- Дистанционная подготовка школьников по информатике http://informatics.mccme.ru
- Сайт «Школа программиста» Красноярского края http://acmp.ru

Материалы

Повторить и обобщить знания, которыми вы должны владеть для успешного прохождения курса

Используемый язык программирования – Паскаль.
Среда программирования – Free Pascal

Рекомендуемая литература для повторения основ языка Паскаль:

- Окулов С.М. «Основы программирования», 2008;
- http://www.intuit.ru/studies/courses/41/41/info.

1. Представление о программе как о наборе инструкций, ввод с консоли и вывод на консоль.

Pascal

Ввод с клавиатуры:

```
readln(a);
```

Вывод на экран:

```
write(a);
```

Вывод на экран с переносом на новую строку:

```
writeln(a);
```

C++

Ввод с клавиатуры:

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

Вывод на экран:

```
std::cout << a;
```

Вывод на экран с переносом на новую строку:

```
std::cout << a << endl;
```

Pascal

2. Основные типы данных

`Byte` – целое в диапазоне 0…255 (1 байт)

`Integer` – целое в диапазоне -32768…32767 (2 байта)

`Longint` – целое в диапазоне -2147483648… 2147483647 (4 байта)

C++

short int	2	-32 768 / 32 767
unsigned short int	2	0 / 65 535
int	4	-2 147 483 648 / 2 147 483 647
unsigned int	4	0 / 4 294 967 295
long int	4	-2 147 483 648 / 2 147 483 647
unsigned long int	4	0 / 4 294 967 295

Pascal

`Real`, `Double`, `Extended` – вещественные числа

C++

типы данных с плавающей точкой		
float	4	-2 147 483 648.0 / 2 147 483 647.0
long float	8	-9 223 372 036 854 775 808 .0 / 9 223 372 036 854 775 807.0
double	8	-9 223 372 036 854 775 808 .0 / 9 223 372 036 854 775 807.0

Pascal

`Char` – символьный тип данных
`String` – строковый тип данных
`Boolean` – логический тип данных (`True` или `False`)

C++

целочисленный (логический) тип данных		
bool	1	0 / 255
целочисленный (символьный) тип данных		
char	1	0 / 255

3. Оператор присваивания (:=), выражения, арифметические операции (+, -, *, /, mod, div)

3.1 Оператор присваивания (=), выражения, арифметические операции (+, -, *, /, %)

---

4. Условные операторы (**if, case**), операторные скобки **begin** – **end**, логические операции **and, or, not**

4.1 Условные операторы (**if, case**), операторные скобки **{ - }**, логические операции **&&, ||, !**

Pascal

```
if <условие> then
    <оператор1>
else
    <оператор2>;
```

```
case <переменная> of
    <знач1>: <опер1>;
    <знач2>: <опер2>;
    …
    else <оперN>;
```

C++

```
if (<условие>) {
<тело условия>;
}
else {
<тело условия>;
}
```

```
switch (<переменная>){
    case <знач1>: {
    <опер1>;
    }
    case <знач2>: {
    <опер2>;
    }
    default: {
    <опер2>;
    }
```

5. Циклические операторы

Pascal

Цикл с параметром:

```
for <пер> := <начало> to <конец> do
   оператор;
```

C++

```
for (счетчик = значение; счетчик < значение;
шаг цикла) {
   тело цикла;
}
```

Pascal

Цикл с предусловием:

```
while <условие> do
    оператор;
```

Цикл с постусловием:

```
repeat
    <операторы>
until <условие>;
```

C++

Цикл с предусловием:

```
while (Условие) {
Тело цикла;
}
```

Цикл с постусловием:

```
do {
Тело цикла;
} while (Условие);
```

7. Одномерные и многомерные массивы. Основные действия с массивами:

- печать,
- поиск максимума (минимума),
- поиск суммы элементов, линейный поиск,
- сортировка простым методом (например, пузырьковая сортировка)

```
A: array [1..10] of integer;
B: array [1..5,1..3] of real;

A[4]:=5;
B[2,1]:=10.5;
```

C++

```
int a[5]
int a[5][3]

int a[3] = {3, 4, 6}
int a[2][3] = {{2,3,4}, {2,7,6}}
```

8. Записи – пользовательский тип данных для описания объектов реального мира

```
X: record
   a: integer;
   b: real;
   c: char;
end;

X.a := 4;
X.b := 5.4;
X.c := ‘f’;
```

8.1 Структура — это , некое объединение различных переменных (даже с разными типами данных), которому можно присвоить имя.

C++

```
struct building   //Создаем структуру!
{
  string owner;    //здесь будет храниться имя владельца
  string city;     //название города
  int amountRooms;  //количество комнат
  float price;    //цена
};


building.city = “Khv”
building.price = “10000”
```

9. Процедуры и функции. Параметры-значения и параметры-переменные. Локальные и глобальные переменные. Область видимости переменных

```
procedure <имя>(<список параметров>);
var …
begin
   …
end;

function <имя>(<список параметров>): <тип>;
var …
begin
   …
end;
```

9.1 Функции — это блоки кода, выполняющие определенные операции. Если требуется, функция может определять входные параметры, позволяющие вызывающим объектам передавать ей аргументы. При необходимости функция также может возвращать значение как выходное.

C++

```
void /*имя функции*/(/*параметры функции*/)
{
// тело функции
}

int /*имя функции*/(/*параметры функции*/)
{
// тело функции
// return <переменная>
}
```

10. Строки. Строковые функции. Работа со строкой как с массивом символов

`copy` – копирование части строки
`delete` – удаление части строки
`insert` – вставка в строку другой строки
`+` (`concat`) – сцепление строк
`pos` – поиск первого вхождения подстроки в строку
`ord` – возвращение кода по символу
`chr` – возвращение символа по его коду

11. Навыки тестирования и отладки. Использование встроенного отладчика

C++

size_t size() const
Возвращает текущее количество символов в строке
size_t length() const
Возвращает текущее количество символов в строке

void resize(size_t n);
Изменяет размер длины строки

void clear() noexcept
Очищает строку

bool empty() const noexcept
Возвращает булев флаг о том, пуста ли строка

at	получение указанного символа с проверкой выхода индекса за границы
find	поиск символов в строке
push_back	добавление символа в конец строки
pop_back	удаляет последний символ
append	добавляет символы в конец строки
compare	сравнивает две строки
replace	заменяет каждое вхождение указанного символа
substr	возвращает подстроку
copy	копирует символы
resize	изменяет количество хранимых символов
swap	обменивает содержимое
find	поиск символов в строке

12. Работа с текстовыми файлами в языке Паскаль

Объявление файловой переменной:

```
f: text;
```

Связывание файловой переменной с физическим файлом на диске:

```
assign(f, <путь к файлу>);
```

Если файл находится в том же каталоге, что и программа (а на олимпиадах так и есть) вместо всего пути можно ограничиться только именем файла.

Открытие файла в режиме чтения:

```
reset(f);
```

Открытие файла в режиме записи:

```
rewrite(f);
```

Закрытие файла:

```
close(f);
```

Чтение из файла значения в переменную `a`:

```
read(f, a);
```

Чтение из файла значения в переменную `a` и переход на новую строчку в файле:

```
readln(f, a);
```

Проверка, достигнут ли конец файла:

```
eof(f)
```

Проверка, достигнут ли конец строки:

```
eoln(f)
```

Запись в файл переменной `a`:

```
write(f, a);
```

Запись в файл переменной `a` и переход на новую строчку в файле:

```
writeln(f, a);
```

C++

```
ofstream fout("cppstudio.txt");
fout.close();

std::ofstream out;
out.open("D:\\hello.txt");
```

Пример

```
fin >> buff; // считали первое слово из файла
cout << buff << endl; // напечатали это слово

fin.getline(buff, 50); // считали строку из файла
fin.close(); // закрываем файл
cout << buff << endl; // напечатали эту строку
```

…В программировании содержатся эстетические и практические ценности. Хорошую программу можно читать как стихи, и она может привлекать как песня, музыка или прекрасная картина. Каждый программист на своем опыте знает, что он ощущает каждый раз, когда программа начинает работать. Еще большее интеллектуальное удовлетворение можно получить, найдя хорошее решение трудной задачи. Наилучшие образцы программирования представляют собой творческую науку и искусство.

Э. Хювёнен, Й. Сеппянен

- Не забывайте о проектировании программ сверху вниз:
  - прежде чем приступить к кодированию, вы должны спроектировать программу на достаточном уровне детализации на бумаге.
  - Не приступайте к кодированию до тех пор, пока не сможете ясно, понятно для любого слушателя рассказать идею решения

- Разбивайте программу на отдельные подпрограммы (процедуры и функции). Старайтесь отлаживать каждую функцию по отдельности

- Старайтесь использовать как можно меньше глобальных переменных: процедуры и функции должны быть максимально независимыми
- Всегда программируйте «с отступами»
- Выбирайте осмысленные имена для переменных, функций и т.д.
- Одна строка – один оператор
- Не забывайте присваивать переменным начальные значения, даже если компилятор сделает это за вас
- Добавляйте комментарии по ходу написания программы
- Не пренебрегайте тестированием программы. Помните, что каждая последняя ошибка – есть предпоследняя.

Особенности программирования задачи

- Программа представляет собой консольное приложение
- Как правило, исходные данные должны считываться из исходного файла и записываться в выходной файл. Все файлы текстовые.
- **Проверять корректность данных в исходном файле не требуется!**
- **Необходимо тщательно следить за корректностью данных, которые записываются в выходной файл**

Особенности олимпиадного задания

- Заданы ограничения на время и на ресурсы памяти (программа должна выполняться не дольше предъявленного лимита и не превышать требований к допустимому объему памяти), т.е. необходимо работать над эффективностью программы

- Решения проверяются автоматизированной системой по заранее заготовленному большому набору тестов (порядка 30 – 40).