УДК 004.021

ДИНАМИЧЕСКОЕ ВЫДЕЛЕНИЕ ПАМЯТИ ДЛЯ СТРОКОВОЙ ПЕРЕМЕННОЙ С ИСПОЛЬЗОВАНИЕМ СТАНДАРТНОЙ БИБЛИОТЕКИ C++

Русак Е.О.
Научный руководитель – Воюш Н.В., ст. преподаватель

Часто при создании проекта возникает необходимость в реализации ввода пользователем некоторой текстовой информации, при этом разработчик сталкивается с неопределенностью, связанной с тем, что невозможно предугадать, какой именно размер строковой переменной выделить на ввод информации. Такая проблема зачастую решается либо выделением слишком большого объема памяти, т.е. «с запасом», либо подключением библиотеки «string», которая автоматически выделяет память для переменной после ее ввода в программу.

Использование библиотеки «string» для решения этой проблемы является наиболее популярным методом, так как имеет следующие преимущества:

- возможность обработки строк стандартными операторами C++ (=, +, ==, <, > и т.п.);
- обеспечение лучшей надежности (безопасности) программного кода. Например, при копировании строк, автоматически осуществляются соответствующие действия, которые могут возникнуть в случае, если строка-источник имеет больший размер чем строка-приемник;
- объявление строки, как самостоятельного типа данных, что обеспечивает непротиворечивость данных.

Однако оба метода имеют свои недостатки. Выделение заведомо слишком большого объема памяти является нерациональным использованием памяти и имеет ряд недостатков:

- теряется память, если переменные фактически не используются или используются, но не на полную;
- может произойти переполнение стека, что приведет к автоматическому завершению выполнения программы;
- такой подход может привести к искусственным ограничениям или переполнению массива;
- в некоторых случаях такой подход и вовсе невозможен в силу очень маленького объема памяти компьютера, либо другого вычислительного устройства, на котором эта программа будет функционировать.

Работа с библиотекой «string» тоже имеет свой недостаток – это замедленная скорость обработки данных. Это связано с тем, что тип «string» – это, фактически, контейнерный класс. А работа с классом требует дополнительной реализации программного кода, который, в свою очередь занимает лишнее время.

Наиболее рациональным будет динамическое выделение памяти, т.е. отправка с помощью оператора «new» в операционную систему запроса зарезервировать некоторую часть памяти.

Для решения этой проблемы с использованием только стандартной библиотеки ввода-вывода C++ «iostream» и оператора «new» предлагается следующий алгоритм:

- инициализация переменной типа char размером один байт;
- ограничение потока ввода до одного байта;
- ввод текста и запись первого его символа в выделенную ранее переменную;
- определение размера текста, оставшегося в буфере, методами библиотеки «iostream»;
- выделение памяти при помощи оператора «new» для массива типа char по размеру текста, установленному ранее;
- в качестве первого элемента получившегося массива устанавливается переменная с первым символом текста;
- все остальные элементы массива заполняются данными из буфера, используя методы библиотеки «iostream».

Таким образом в результате работы данного алгоритма образуется массив символов, который содержит в себе информацию, введенную пользователем. Такой метод не требует подключения дополнительных библиотек, либо выделения заведомо большего объема памяти. Также при использовании такого алгоритма в отличие от статического или автоматического выделения памяти, сама программа отвечает за запрос и обратный возврат динамически выделенной памяти.

Тем не менее, у такого подхода существуют следующие недостатки:

- необходимо постоянно контролировать размер информации, которая считывается из буфера, так как в нем помимо введенных пользователем данных может содержаться ненужная информация;
- различные операции с полученной строкой необходимо производить побайтно, контролируя ее размер, так как полученная таким образом строка не содержит символа конца строки (/0);
- отсутствует возможность обработки строк стандартными операторами C++, таким образом даже наиболее простые операции со строками выглядят сложно и требуют написания чрезмерного программного кода;
- при запросе памяти из операционной системы в редких случаях она может быть недоступной, что вызовет сбой и завершение работы программы.